

# Sicherheit von Betriebssystemen – VO 08: Einführung in iOS

Rafael Vrecar

26S

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

- Allgemeines über iOS
- Secure Enclave (Prozessor)
- Keychain
- Maßnahmen zur Sicherstellung der Integrität
  - Secure Boot
  - Maßnahmen bzgl. Betriebssystemintegrität
  - Sichere Software Updates
- iOS Apps – Entwicklung, Verbreitung, Aufbau
- Ausblick
- Zusammenfassung & Abschluss

# Allgemeines über iOS

# 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
- größtenteils „Closed Source“, nur auf Apple-Geräten unterstützt
- basiert auf macOS (bzw. OS X)
- aktuelle Version (Herbst 2025): iOS 26

# Secure Enclave (Prozessor)

# Secure Enclave (Prozessor)

- „Enklave“  $\hat{=}$  isolierter Bereich, vom Hauptprozessor abgegrenzt
- seit A7 auf Apples Chips enthalten (iPhone 5S, 2013)
- Secure Enclave Prozessor (SEP) wird ausschließlich(!) von Secure Enclave (SE) benutzt
- niedrige Taktrate soll **Clock- & Power-Attacken** verhindern
- Ziel der Abgrenzung: Verhinderung von **Side-Channel-Attacken**
- Speicher der SE vom Anwendungsprozessor isoliert
- zuständig für Verschlüsselung & Schutz von User:innen-Daten...
- Touch ID & Face ID (Biometrie-Informationen) werden mithilfe der SE verschlüsselt aufbewahrt

# Keychain

# Keychain (Schlüsselbund) – User:innen-Perspektive

- Passwortmanager mit 2FA-Token-Speicher seit 2021
- verwaltet z.B. Anmeldedaten, Zertifikate, Schlüssel, gespeicherte WLANs
- Inhalte der Keychain können über Apple ID mit anderen (Apple) Geräten synchronisiert werden



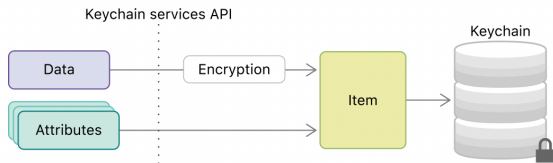
(macOS Icon)

# Keychain (Schlüsselb.) – technische Perspektive: Überblick

- sichere Methode zum verschlüsselten Speichern von vertraulichen Datenfragmenten, ermöglicht Speicherung kurzer Datenbits
- als **verschlüsselte SQLite-Datenbank** implementiert, liegt im Dateisystem
- Schlüsselbundobjekte mithilfe zweier AES-256-GCM-Keys encrypted:
  - **Tabellenschlüssel** zur Ent- & Verschlüsselung von Schlüsselbund-Metadaten ⇒ schnelle Suche möglich
  - 1 pro Zeile erzeugter Schlüssel (**SecretKey**) für sensible Daten
- *für Entwickler:innen*: nur ein Schlüsselbund für alle Apps unter iOS; auf Datenbank kann nur über die API zugegriffen werden
- Schlüsselbundeinträge nur zwischen Apps *desselben* Entwicklungsteams gemeinsam nutzbar

# Keychain (Schlüsselbund) – technische Perspektive: Attribute

- müssen zusammen mit Daten übergeben werden
- sind nicht verschlüsselt
- werden für Zugriffskontrolle & Suche verwendet



© Apple

(Vergleiche [https://developer.apple.com/documentation/security/keychain\\_services/keychain\\_items](https://developer.apple.com/documentation/security/keychain_services/keychain_items))

[//developer.apple.com/documentation/security/keychain\\_services/keychain\\_items](https://developer.apple.com/documentation/security/keychain_services/keychain_items))

# Sicherheit von Keychain-Daten

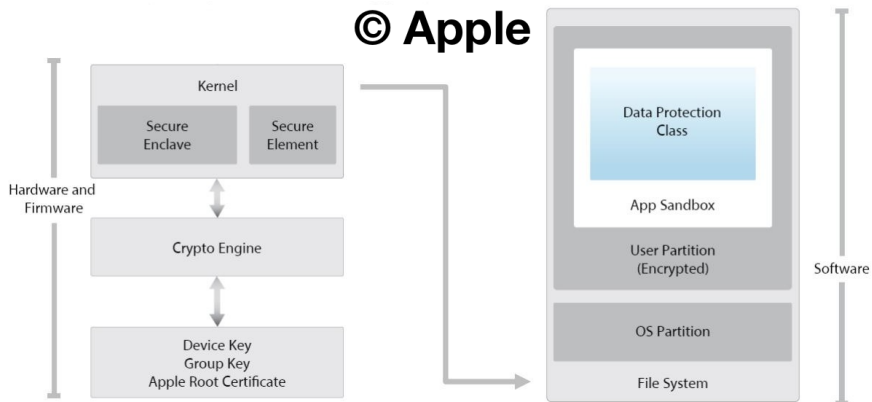
Verfügbarkeit	Sicherheit von Dateidaten	Sicherheit von Schlüsselbunddaten
Wenn entsperrt	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
Wenn gesperrt	NSFileProtectionCompleteUnless Open	Nicht verfügbar
Nach erstem Entsperren	NSFileProtectionCompleteUntil FirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Immer	NSFileProtectionNone	kSecAttrAccessibleAlways
Code aktiviert	Nicht verfügbar	kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly

© Apple

(Vergleiche <https://support.apple.com/de-at/guide/security/secb0694df1a/web>)

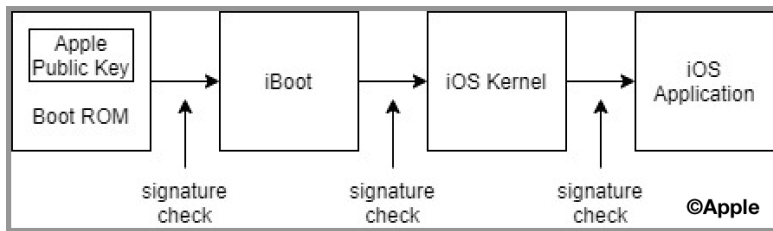
# Maßnahmen zur Sicherstellung der Integrität

# 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



# Secure Boot: Schritt für Schritt

(Einschalten ...)

1. Anwendungsprozessor führt Code vom *Boot ROM* aus
  - unveränderlicher Code
  - „Hardware Root of Trust“
  - wird bei Herstellung eingebaut ⇒ implizit vertrauenswürdig
  - Boot ROM enthält Apples öffentlichen Schlüssel
2. mithilfe dieses Schlüssels wird *iBoot Bootloader* auf Signatur geprüft
3. iBoot führt iOS Kernel aus (A9 & älter haben Low Level Bootloader, der von Boot ROM geladen wird & iBoot verifiziert ⇒ unterschiedliche Art der Verifikation)

(...OS booten)

# Secure Boot: Fehlerfall

- mit iTunes/Finder verbinden via USB ⇒ Factory Reset
- hierbei gibt es zwei verschiedene Modi
  1. *DFU Mode* (von Boot ROM gesetzt) – A12 & später
  2. *Recovery Mode* (von iBoot, also später, gesetzt) – A10 & später
- auf Geräten mit Mobilfunk: Base Band Prozessor führt zusätzliche Checks hinsichtlich der Integrität der Modem-Firmware aus
- außerdem: Secure Enclave führt „Selbstcheck“ durch (Secure Boot) & überprüft, dass sepOS von Apple signiert

# Secure Boot: iBoot & Speichersicherheit

- seit iOS 14: C Compiler Toolchain verändert, die für iBoot Build verwendet wird
- zusätzliche Sicherheitsmaßnahmen zum Speicherschutz
- **Buffer Overflows:** Pointer haben Bereichsinformationen gespeichert ⇒ beim Speicherzugriff verifiziert
- **Heap Exploits:** Heap Data von Meta Data getrennt, *Double Free Errors* sollen erkannt werden
- **Type Confusion:** Pointer haben Typinformation gespeichert (beim Casting verifiziert), dynamische Speicherreservierung nach statischem Typ aufgetrennt (*Free Errors* verhindern)

# Betriebssystemintegrität – Maßnahmen 1/2

- **Kernel Integrity Protection:**

Memory Controller ⇒ geschützte Speicherregion für iBoot, um Kernel + Erweiterungen zu laden  
nach Start: Read-Only

- **Fast Permission Restrictions:**

CPU Register, das Berechtigungen per Thread einschränkt (APRR Register)  
⇒ Execute Berechtigungen vom Speicher ohne System Call entfernbar  
⇒ JIT Compiled Code Attacken schwieriger, da Ausführung von Befehl im Speicher effektiv nicht möglich ist, während Bereich gelesen/beschrieben wird

# Betriebssystemintegrität – Maßnahmen 2/2

- **System Coprocessor Integrity Protection:**  
soll Modifikation der Coprocessor Firmware verhindern (bspw. Secure Enclave), äquivalent zu Kernel Integrity Protection durch geschützte Speicherbereiche
- **Pointer Authentication Codes:**  
soll Exploiting von Memory Corruption Bugs verhindern  
Pointer werden signiert, um bspw. Änderung der Rücksprung-Adresse zu verhindern (und so zu schadhaftem Code springen zu können)
- **Page Protection Layer:**  
soll verhindern, dass Code im user space nach Codesignatur modifiziert wird  
Permission Management für Page Tables

# OS-Integrität – unterstützte Chips

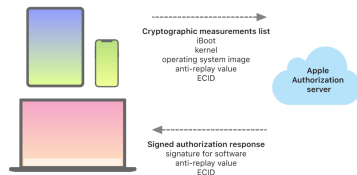
Feature	A10	A11, S3	A12, S4	A13, S5	A14, S6	M1
Kernel Integrity Protection	✓	✓	✓	✓	✓	✓
Fast Permission Restrictions		✓	✓	✓	✓	✓
System Coprocessor Integrity Protection			✓	✓	✓	✓
Pointer Authentication Codes			✓	✓	✓	✓
Page Protection Layer		✓	✓	✓	✓	See Note below.

© Apple

re: M1: bei macOS nicht in dieser Form möglich, weil dort auch nicht-signierter Code ausgeführt werden darf.

# 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)
    - ⇒ von Apple signiert
  - Apps von Dritten
    - ⇒ von Entwickler:in signiert
- „Chain of Trust“-Konzept auf Apps ⇒ 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 ⇒ nur Lesezugriff

# iOS Apps – Berechtigungen (Entitlements): Allgemeines

- = Schlüssel-Wert-Paare, die eine Genehmigung zur Nutzung eines Dienstes oder einer Technologie erteilen
- werden zusammen mit App signiert
- Xcode fügt Berechtigungen in die Eigenschaftslisten-Datei mit Endung `.entitlements` ein

# iOS Apps – Berechtigungen (Entitlements): Beispiele

- „Mit Apple anmelden“-Berechtigung: `com.apple.developer.applesignin`
- Authentifizierung: „AutoFill Credential Provider“-Berechtigung (Boolean; gibt an, ob Anwendung mit User:innen-Genehmigung User:innennamen & Kennwörter für „AutoFill“ in Safari & anderen Anwendungen bereitstellen darf): `com.apple.developer.authentication-services  
.autofill-credential-provider`
- Kontakte (Boolean: gibt an, ob Anwendung auf in Kontakten gespeicherte Notizen zugreifen darf): `com.apple.developer.contacts.notes`

# iOS Apps – Speicherverwürfelung

- **Address Space Layout Randomization (ASLR):** schützt vor Exploits, welche Speicher modifizieren
- integrierte Apps stellen mithilfe von ASLR sicher, dass beim Start alle Speicherbereiche nach Zufallsprinzip vergeben werden
- zufällige Anordnung der Speicheradressen von ausführbarem Code, Systembibliotheken (System Libraries) & zugehöriger Programmelemente verringert Wahrscheinlichkeit vieler Exploits
- *Beispiel:* Return-to-libc Angriff

# iOS Apps – Execute Never (XN)

- ARM-Funktion kennzeichnet Speicherseiten als „nicht ausführbar“
- beschreibbar & ausführbar gekennzeichnete Speicherseiten von Apps nur unter streng kontrollierten Bedingungen verwendbar
- Kernel überprüft, ob die Apple vorbehaltene Berechtigung „dynamic code signing“ vorhanden ist
  - ⇒ selbst dann kann nur einziger mmap-Aufruf genutzt werden, um so eine Seite anzufordern, die eine zufällige Adresse erhält
- Safari verwendet Funktion für seinen *JavaScript JIT Compiler*

# 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.tigissoftware.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:

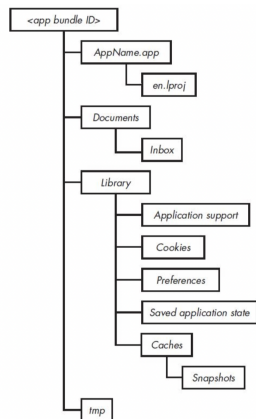
- enthält Verzeichnis für jede auf Gerät installierte App
- *Pfad*: `/private/var/containers/Bundle/Application/<UUID>`
- jedes App-Verzeichnis enthält Ordner `<App-Name>.app`, in dem Ressourcen, `Info.plist` & Binär-Dateien enthalten

## Shared:

- für App-Gruppen: Apps eines Entwickler:innenaccounts können Inhalte, Schlüsselbundobjekte, Einstellungen gemeinsam verwenden
- 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  $\Rightarrow$  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  $\Rightarrow$  Versuch schlägt fehl, App kann abstürzen
  - mehr als 20 geschützte Ressourcen, z.B. Kamera, Mikrofon, Dateien, Ordner, Netzwerk, NFC, Standort
- Info.plist: Beispiel (XML) auf der nächsten Folie

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>NSLocationWhenInUseUsageDescription</key>
6   <string>Your location is used to help you find health services near you.</string>
7   <key>CFBundleDevelopmentRegion</key>
8   <string>en</string>
9   <key>CFBundleURLTypes</key>
10  <array>
11    <dict>
12      <key>CFBundleURLSchemes</key>
13      <array>
14        <string>adahealth</string>
15        <string>ada</string>
16      </array>
17      <key>CFBundleURLName</key>
18      <string>com.ada.app</string>
19    </dict>
20  </array>
21
22  <!-- [...] -->
23 </dict>
24 </plist>
```

# 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
  - ⇒ 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  $\Rightarrow$  z.B. in SQLite & XML serialisierbar
- verwaltet Objekt-Instanzen zur Laufzeit
- High-Level-Abstraktionsdarstellung ermöglicht direkte Kommunikation mit SQLite-Datenbanken (siehe Bild)
- 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
- hilfreiche Befehle:
  - `sqlite3 CoreData.sqlite .schema`
  - `sqlite3 CoreData.sqlite .dump`
  - `strings CoreData.sqlite`
  - mit `strings` oder `grep` erhält man schnell Teile aus Datei, ohne zusätzliche Dekodierung, jedoch ohne Kontext

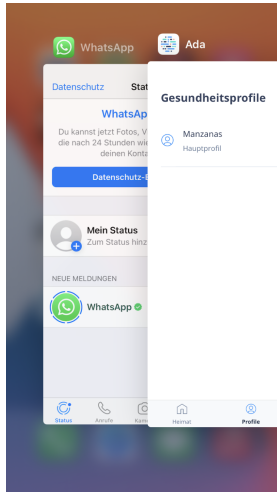


(Vergleiche <https://developer.apple.com/documentation/coredata>)

# Lokaler Datenspeicher: Caches

- On-disk Cache für NSURLRequest  
(<https://developer.apple.com/documentation/foundation/nsurlrequest>):
  - 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



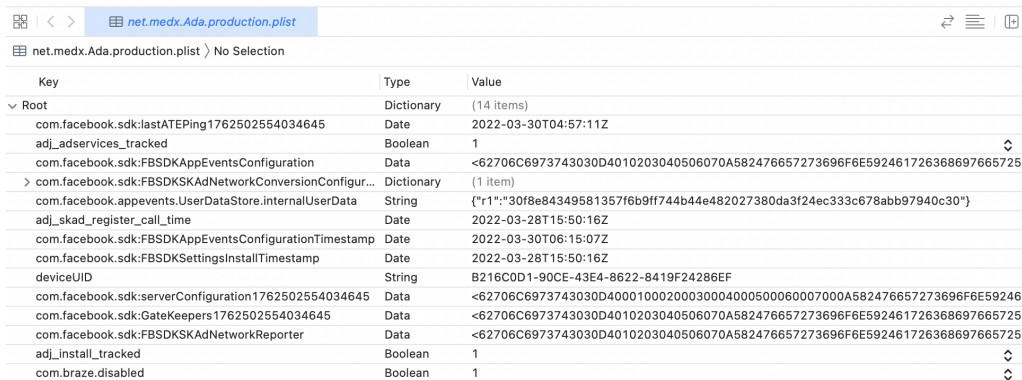
# Lokaler Datenspeicher: URL Cache & Snapshots auslesen

- `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 Anwendungsleistung  
⇒ 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  
⇒ nicht empfehlenswert für sensiblen Daten & Anmeldedaten
- kann für Speichern von Einstellungsdaten verwendet werden

# User Defaults: Beispiel (geöffnet in Xcode)



net.medx.Ada.production.plist

net.medx.Ada.production.plist > No Selection

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	<62706C6973743030D4010203040506070A582476657273696F6E592461726368697665725
> 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	B216COD1-90CE-43E4-8622-8419F24286EF
com.facebook.sdk:serverConfiguration1762502554034645	Data	<62706C6973743030D40001000200030004000500060007000A582476657273696F6E59246
com.facebook.sdk:GateKeepers1762502554034645	Data	<62706C6973743030D4010203040506070A582476657273696F6E592461726368697665725
com.facebook.sdk:FBSDKSKAdNetworkReporter	Data	<62706C6973743030D4010203040506070A582476657273696F6E592461726368697665725
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)
- Netzwerksicherheit (in einer zukünftigen Einheit besprochen)
- ...

# Zusammenfassung

- Apples Software i.d.R. „closed source“
- iOS Updates
- Secure Enclave
- Sicherstellung der Integrität von Software
- iOS Apps

# Abschlussbemerkungen

- iOS hat viele Sicherheitsvorkehrungen, nutzen Sie diese!
- Bleiben Sie kritisch & bedenken Sie, dass Apples Sicherheitsmaßnahmen keinesfalls vor Fehlern gefeit sind. ;)
- Behalten Sie im Hinterkopf, dass Apples Code (größtenteils) nicht offen ist & daher mehr Vertrauen erfordert, da Sie ihn nicht (einfach) überprüfen bzw. selbst kompilieren können.
- Updates installieren ist (i.A.) natürlich (auch bei Apple) eine gute Idee.
- ...und zuletzt: **Don't be evil!** – Apple hat ein Security Bounty Programm (<https://developer.apple.com/security-bounty/>)

# Literaturverzeichnis 1/9

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

# Literaturverzeichnis 2/9

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

# Literaturverzeichnis 3/9

- Apple Dokumentation & Support: <https://support.apple.com/>
- Apple Platform Security (May 2021): [https://manuals.info.apple.com/MANUALS/1000/MA1902/en\\_US/apple-platform-security-guide.pdf](https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf)
- 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](https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=79126)

# Literaturverzeichnis 4/9

- Data Protection:

<https://support.apple.com/de-at/guide/security/secf6276da8a/web>

- Sealed Key Protection (SKP):

<https://support.apple.com/de-at/guide/security/secf5549a4f5/web>

- Keychain:

<https://support.apple.com/de-at/guide/security/secb0694df1a/web>

- Erweiterungen:

<https://support.apple.com/de-de/guide/security/secabd3504cd/web>

- App-Gruppen:

<https://support.apple.com/de-de/guide/security/sec1a976c067/web>

# Literaturverzeichnis 5/9

- Sandboxing:

<https://support.apple.com/de-de/guide/security/sec15bfe098e/web>

- Übersicht:

<https://support.apple.com/de-de/guide/security/secf49cad4db/web>

- Datensicherheit bei Apple-Geräten:

<https://support.apple.com/de-at/guide/security/sece8608431d/1/web/1>

- Datensicherheitsklassen:

<https://support.apple.com/de-at/guide/security/secb010e978a/web>

# Literaturverzeichnis 6/9

- Chell (2015): The Mobile Application Hacker's Handbook
- Apple. Protected Resources. 2021.  
[https://developer.apple.com/documentation/bundleresources/information\\_property\\_list/protected\\_resources](https://developer.apple.com/documentation/bundleresources/information_property_list/protected_resources)
- Apple. Core Data. 2021.  
<https://developer.apple.com/documentation/coredata>
- Apple. UserDefaults. 2021.  
<https://developer.apple.com/documentation/foundation/nsuserdefaults>

# Literaturverzeichnis 7/9

- 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 8/9

- Apple. Item Attribute Keys and Values. 2021.  
[https://developer.apple.com/documentation/security/keychain\\_services/keychain\\_items/item\\_attribute\\_keys\\_and\\_values](https://developer.apple.com/documentation/security/keychain_services/keychain_items/item_attribute_keys_and_values)
- Apple. Keychain data protection. Feb. 18, 2021. <https://support.apple.com/guide/security/keychain-data-protection-secb0694df1a/1/web/1>
- Apple. Keychain Services. 2021. [https://developer.apple.com/documentation/security/keychain\\_services](https://developer.apple.com/documentation/security/keychain_services)
- OWASP. Data Storage on iOS. 2019.  
<https://mobile-security.gitbook.io/mobile-security-testing-guide/ios-testing-guide/0x06d-testing-data-storage>

# Literaturverzeichnis 9/9

- Apple. Bundle IDs. 2021. [https://developer.apple.com/documentation/appstoreconnectapi/bundle\\_ids](https://developer.apple.com/documentation/appstoreconnectapi/bundle_ids)
- Apple. Protecting the User's Privacy. 2021. [https://developer.apple.com/documentation/uikit/protecting\\_the\\_user\\_s\\_privacy](https://developer.apple.com/documentation/uikit/protecting_the_user_s_privacy)

**Vielen Dank!**

`https://establishing-security.at/`