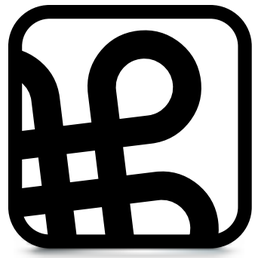


Macoun'11

Sicher nach IOS

Klaus M. Rodewig

Referent

- Senior IT Security Analyst
- CPSSE
- iOS & Mac OS X:
  - Code Audit
  - Reverse Engineering
  - Forensik

Agenda

- Überblick Smartphone-Sicherheit
- iOS-Sicherheit
- Sicherheit im iOS-API
- Erstellen sicherer Apps

Attacker‘s view

- 24x7
- (permanente) Verbindung zum Internet
- dienstliche und private Daten
- Standortbestimmung
- Datenübertragung über Mobilfunk
- zahlreiche Schnittstellen

The vault

Adressbuch

Bilder

Kalender

Keychain

Dateien

Position

The vault

- Adressbuch
- Keychain
- Email
- Exchange
- Safari
- WLAN
- Bilder
- Dateien
- App-Daten
- Zertifikate
- VPN
- App-Dateien
- Kalender
- Position

The vault

- Adressbuch
- Keychain
- Email
- Exchange
- Safari
- WLAN
- Bilder
- Dateien
- App-Daten
- Zertifikate
- VPN
- App-Dateien

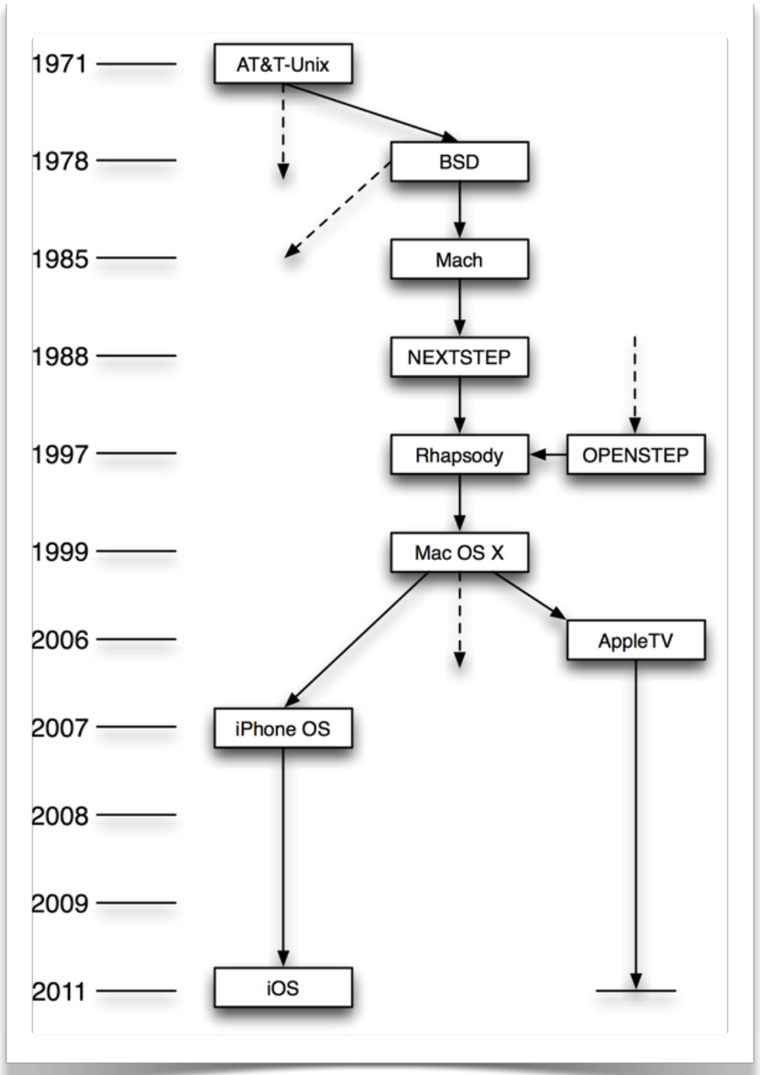
IT-Sicherheit 2011

- Infrastrukturen und Netzwerke sind ausreichend abgesichert
- 80-90% aller Angriffe auf Applikationsebene
  - Microsoft Security Intelligence Report
  - Verizon Databreach Report

Angriffe im Wandel der Zeit

- 1984: Morris-Wurm
- 2006-2008: Debian-SSL-Bug
- 2010: China leitet für 15 Minuten unbemerkt 15% des weltweiten Internet-Verkehrs um
- 2010: Stuxnet
- 2011: Angriff auf Sony (100 Millionen Kundendaten erbeutet)
- 2011: Angriff auf RSA
- 2011: Angriffe auf Rüstungsindustrie mit Kenntnissen aus RSA-Angriff
- 2011: Untergang der SSL-PKI (Comodo, DigiNotar, GlobalSign)

iOS


1971
1978
1985
1988
1997
1999
2006
2007
2008
2009
2011
AT&T-Unix
BSD
Mach
NEXTSTEP
Rhapsody
OPENSTEP
Mac OS X
AppleTV
iPhone OS
iOS

Sicherheit I

- Application Sandboxing
- Code Signing
- Keychain
  - Sqlite-Datenbank
  - Passwörter, Schlüssel, Zertifikate
- Mobile Device Management API

Sicherheit II

- alle Apps laufen unter Benutzerkonto *mobile*
- root-Passwort ist *alpine*
- keine setuid-Dateien
- keine Shell
- Stack und Heap sind NX
- ASLR (>= iOS 4)

Sicherheit III

- Speicher kann nicht RWX sein
- eingeschränkte Hintergrundprozesse
- Hardware-Verschlüsselung
  - physischer Zugriffsschutz

Sicherheit III

-
-
- Hardware
  - physischer Zugriffsschutz

„On iPhone 3G, wiping overwrites the data on the device, which can take approximately one hour for each 8 GB of device capacity. Connect the device to a power supply before wiping.“

iOS-Bugs (Auszug)

- iOS 4.0 (2010): u.a. 19 arbitrary code execution
- iOS 4.1 (2010): u.a. 27 arbitrary code execution
- iOS 4.2 (2010): u.a. 42 arbitrary code execution
- iOS 4.3 (2011): u.a. 53 arbitrary code execution
- alle iOS-Versionen bis 4.3.5: SSL-Bug bei Prüfung des CA-Bits

Jailbreak

- Kompromittierung auf Kernel-Ebene
- Schreibzugriff auf /
- kein Code Signing
- kein Sandboxing
- Firmware & Tools unbekannter Herkunft
- jailbreakme.com

Jailbreak

- Kompromittierung auf Kernel-Ebene
- Schreibzugriff auf /
- kein Code Signing
- kein Sandboxing
- Firmware & Tools unbekannter Herkunft
- jailbreakme.com

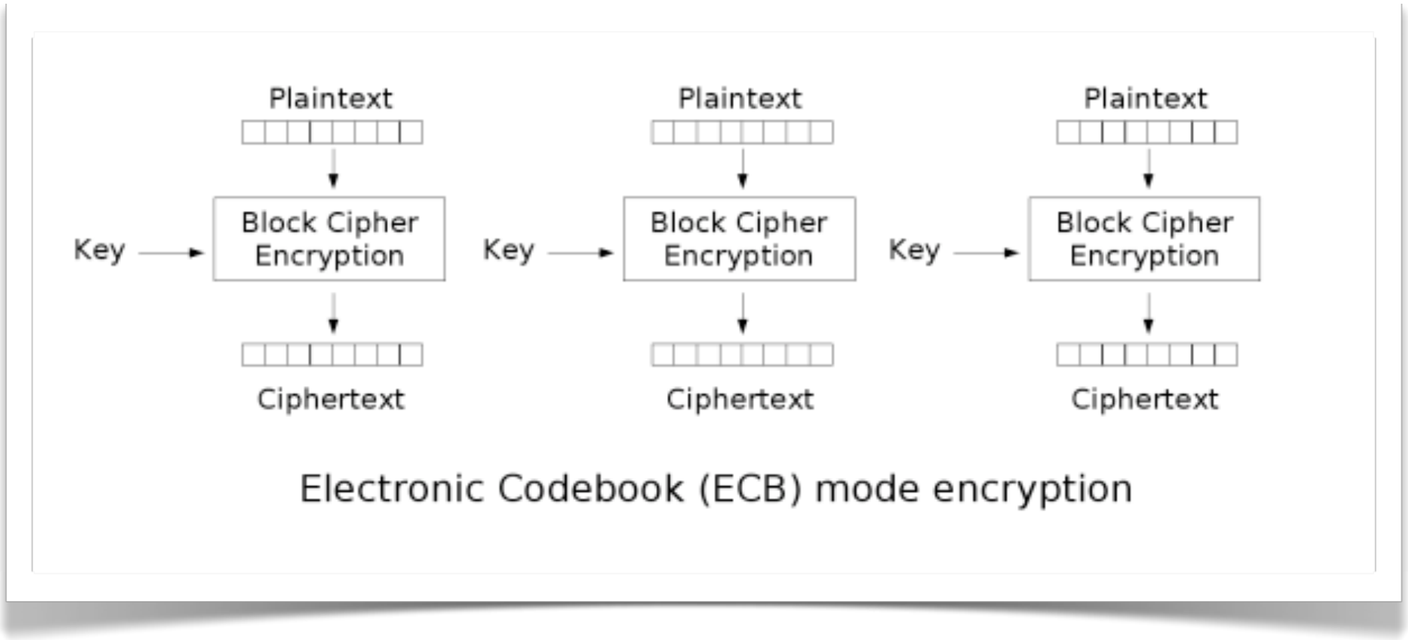
Bootloader-Bug

- bis inkl. iPhone 4 und iPad 1
- Booten einer unsignierten Firmware im DFU-Modus
- Mounten des Dateisystems
- Brute-Force-Angriff gegen Passcode
- kein Schutz durch Verzögerungsmechanismus und forced wipe

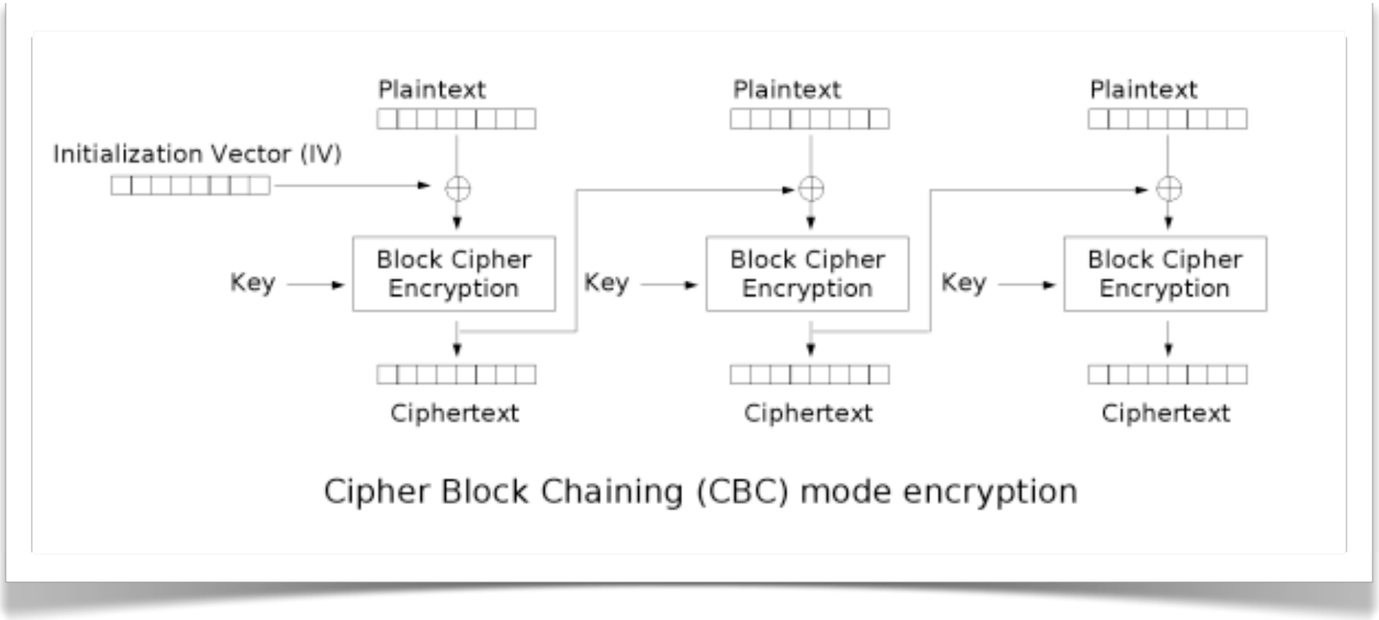
iOS Crypto Library

- Common Crypto Interface CCCryptor(3cc)
- referenziert in Sample Code *CryptoExercise*
- AES, 3DES, RC4, DES, …
- CBC ist default, ECB optional

Electronic Cookbook Mode



Cipher Block Chaining Mode



CommonCryptor.h

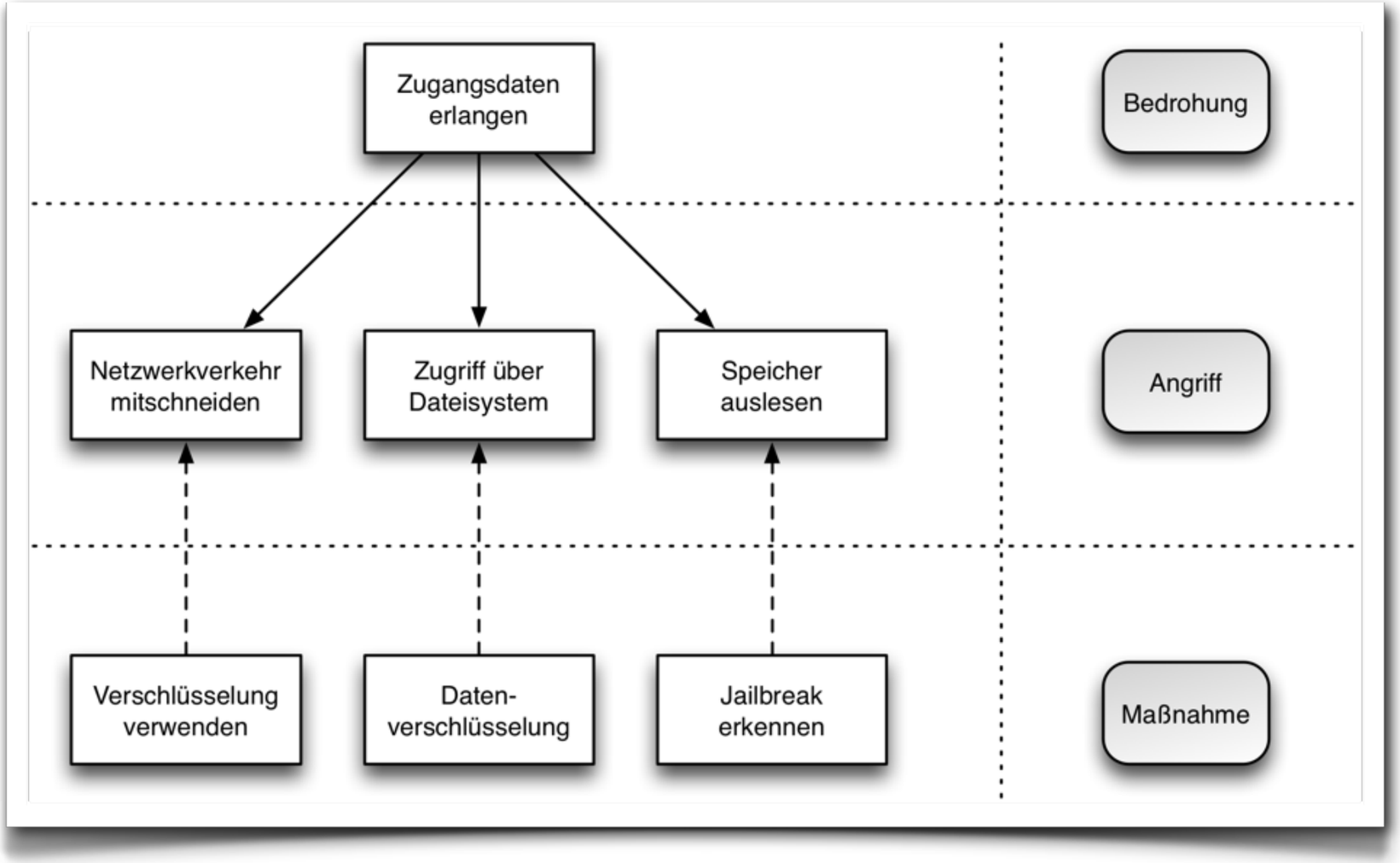
„Another option for block ciphers is Cipher Block Chaining, known as CBC mode. When using CBC mode, an Initialization Vector (IV) is provided along with the key when starting an encrypt or decrypt operation. ***If CBC mode is selected and no IV is provided, an IV of all zeroes will be used.*** *„*

iControl

„Der Entwickler gestand freimütig ein, dass er lediglich Anwendungsprogrammierer, aber kein Sicherheitsexperte sei und iControl nur in seiner Freizeit am Wochenende weiterentwickle. Leider merkt man das auch am Resultat. Wer diesem Programm vertrauliche Daten anvertraut, handelt fast schon fahrlässig.“

Quelle: Heise Verlag

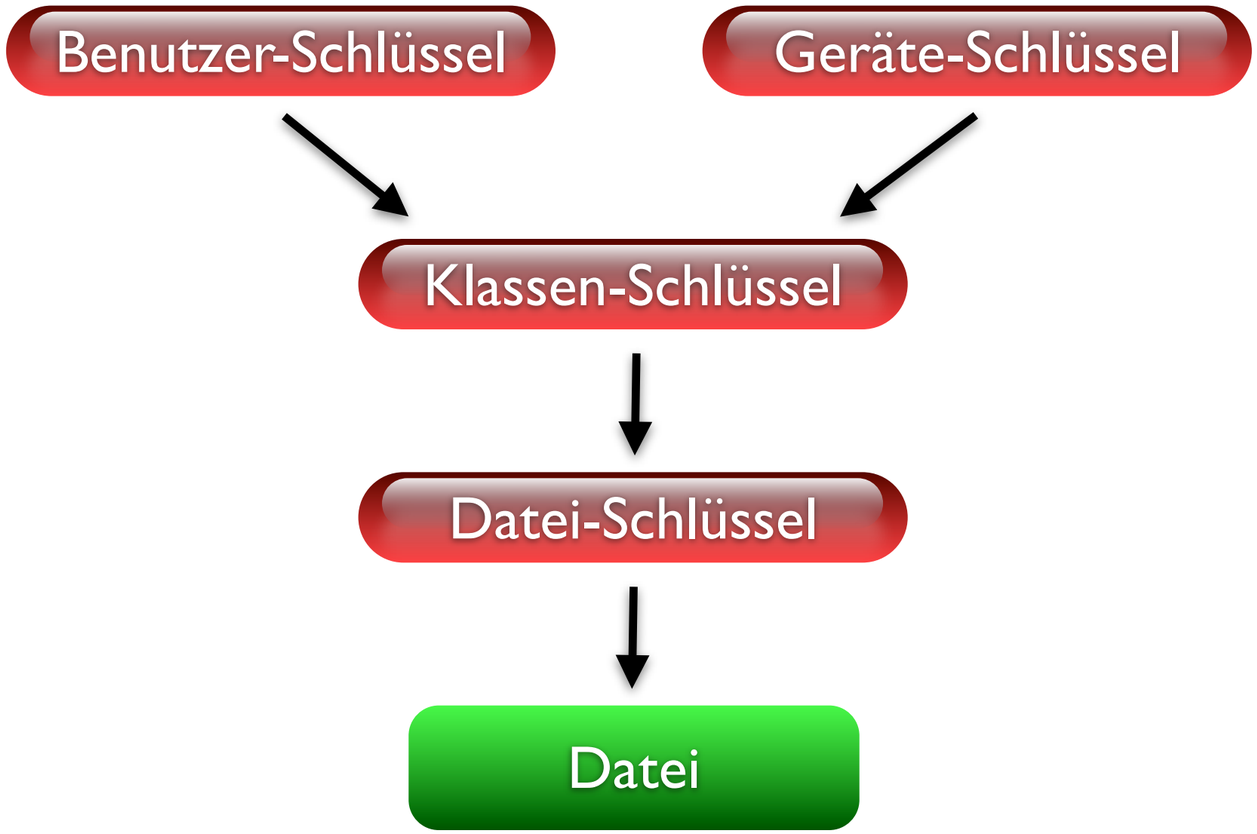
Bedrohung, Angriff, Maßnahme



iOS 4 - Verschlüsselung

- AES 256
- Geräte-Schlüssel
- Benutzer-Schlüssel (Passcode)
- verschlüsseltes Backup

Dateisystem-Verschlüsselung



Filesystem* Protection Classes

Entschlüsselt	Protection class
Immer	**`NSFileProtectionNone`**
Nach erster Eingabe des Passcodes	`NSFileProtectionCompleteUntilFirstUserAuthentication`
Nach Boot oder Eingabe des Passcodes	`NSFileProtectionComplete`
Zugriff im entsperrten Zustand, danach weiter entschlüsselt	`NSFileProtectionCompleteUnlessOpen`

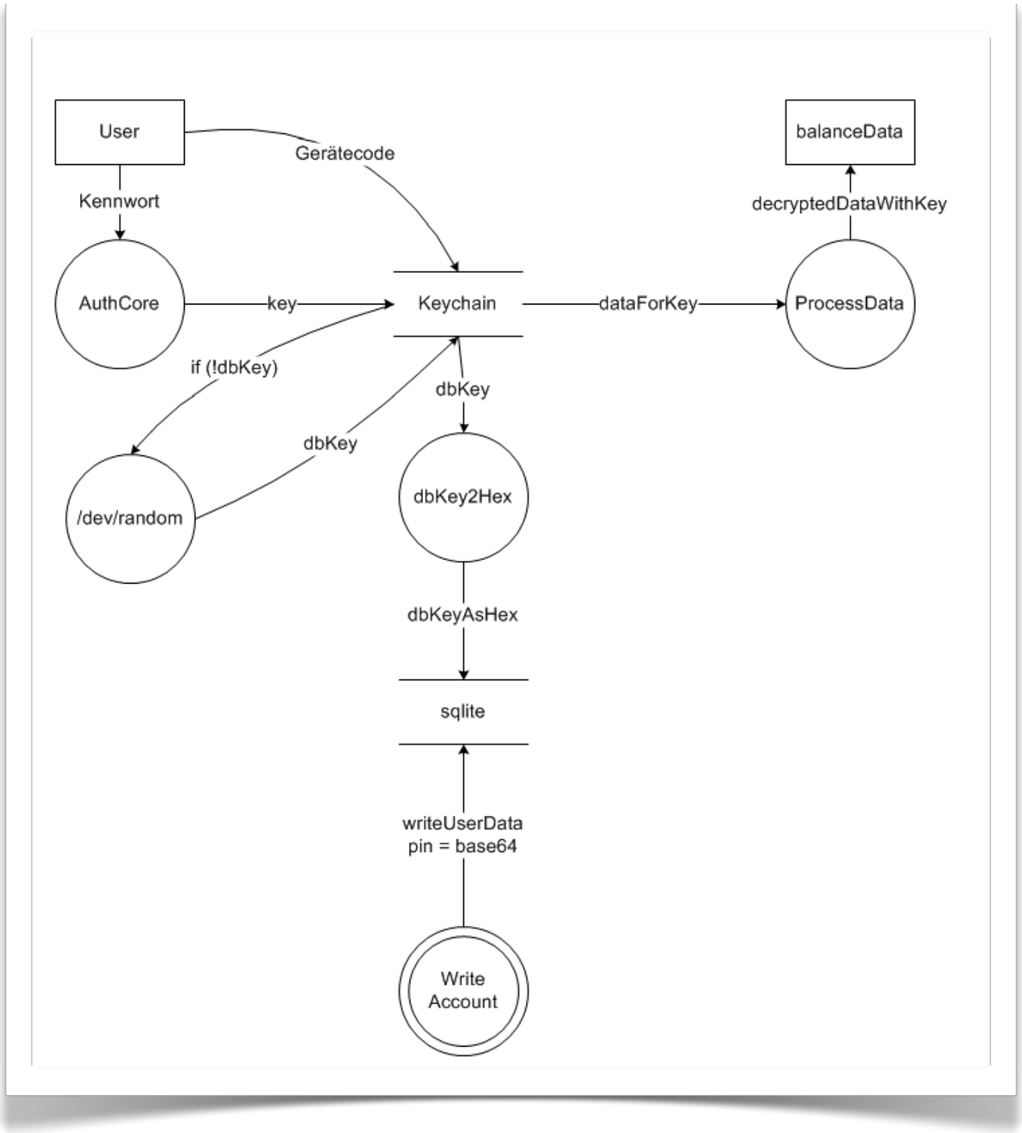
Keychain Protection Classes

Entschlüsselt	Protection class
Immer	`kSecAttrAccessibleAlways`
Immer, nur aktuelles Gerät	`kSecAttrAccessibleAlwaysThisDeviceOnly`
Nach erster Eingabe des Passcodes	`kSecAttrAccessibleAfterFirstUnlock`
Nach erster Eingabe des Passcodes, nur aktuelles Gerät	`kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly`
Nach Eingabe des Passcodes	`kSecAttrAccessibleWhenUnlocked`
Nach Eingabe des Passcodes, nur aktuelles Gerät	`kSecAttrAccessibleWhenUnlockedThisDeviceOnly`

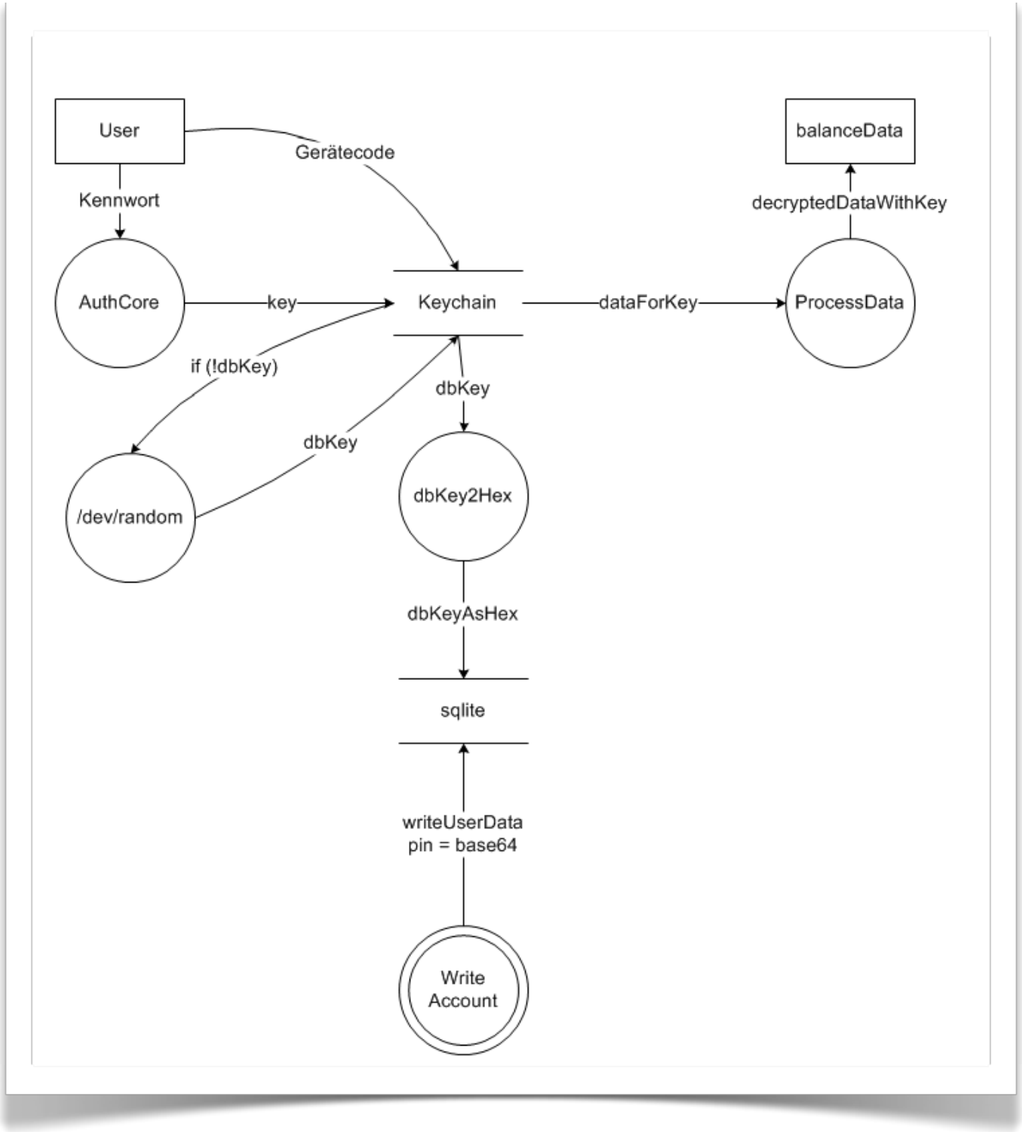
Keychain-Beispiel

Key	Object
kSecClass	`kSecClassGenericPassword`
kSecAttrService	`Cocoanehead Bacefook`
kSecAttrLabel	`1337 Bacefook App`
kSecAttrAccount	`klaus@rodewig.de`
kSecAttrAccessible	**`kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly`**
kSecValueData	`foopassw0rd4711`

Dooooh!


User
Gerätecode
balanceData
Kennwort
decryptedDataWithKey
AuthCore
key
Keychain
dataForKey
ProcessData
if (!dbKey)
dbKey
dbKey
/dev/random
dbKey2Hex
dbKeyAsHex
sqlite
writeUserData
pin = base64
Write
Account

Dooooh!



Key	Object
kSecClass	`kSecClassGenericPassword`
kSecAttrService	`RaffzahnApp`
kSecAttrLabel	`RaffzahnApp`
kSecAttrAccount	**`myS3cretP@ssw0rd`**
kSecAttr-Accessible	`./.`
kSecValueData	`sqlLiteKey`

System-Standards

Item	Protection class
WLAN-Kennwörter	`kSecAttrAccessibleAlways`
Exchange-Zugangsdaten	`kSecAttrAccessibleAlways`
VPN-Zugangsdaten	`kSecAttrAccessibleAlways`
Email-Zugangsdaten	`kSecAttrAccessibleAfterFirstUnlock`
LDAP-, CalDAV-, CardDAV-Zugangsdaten	`kSecAttrAccessibleWhenUnlocked`
iTunes-Backup-Passwort	`kSecAttrAccessibleWhenUnlockedThisDeviceOnly`

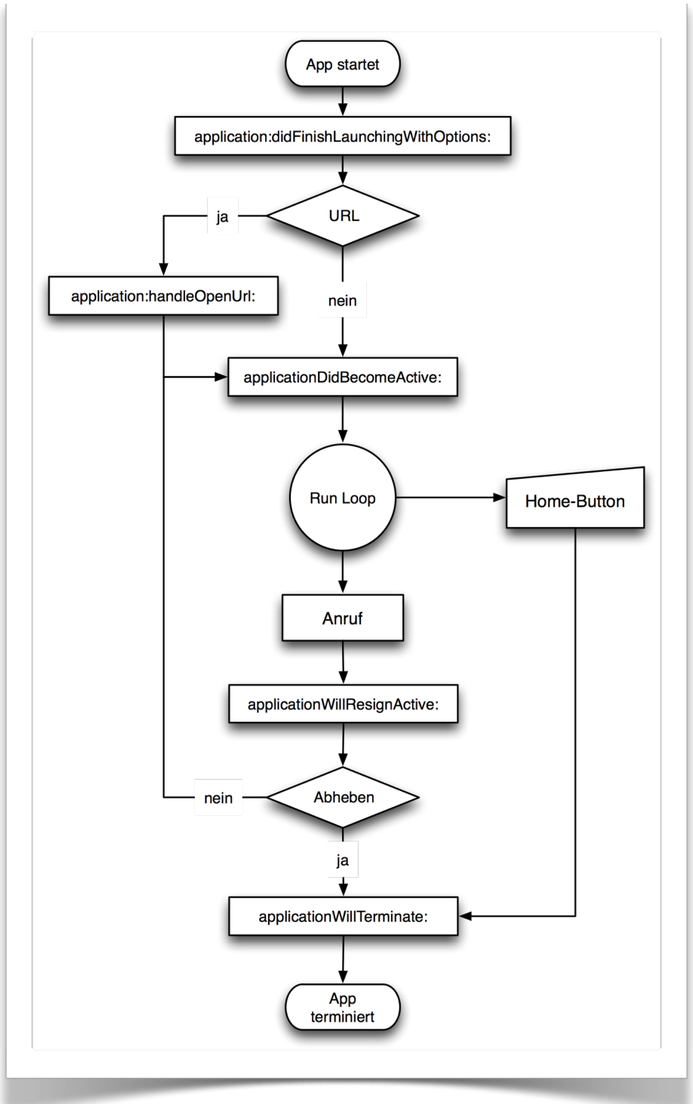
Keychain

```
NSMutableDictionary *query = [NSMutableDictionary dictionary];

[query setObject:(id)kSecClassGenericPassword forKey:(id)kSecClass];
[query setObject:service forKey:(id)kSecAttrService];
[query setObject:label forKey:(id)kSecAttrLabel];
[query setObject:account forKey:(id)kSecAttrAccount];

[query setObject:(id)kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly
forKey:(id)kSecAttrAccessible];

[query setObject:[input dataUsingEncoding:NSUTF8StringEncoding]
forKey:(id)kSecValueData];
```

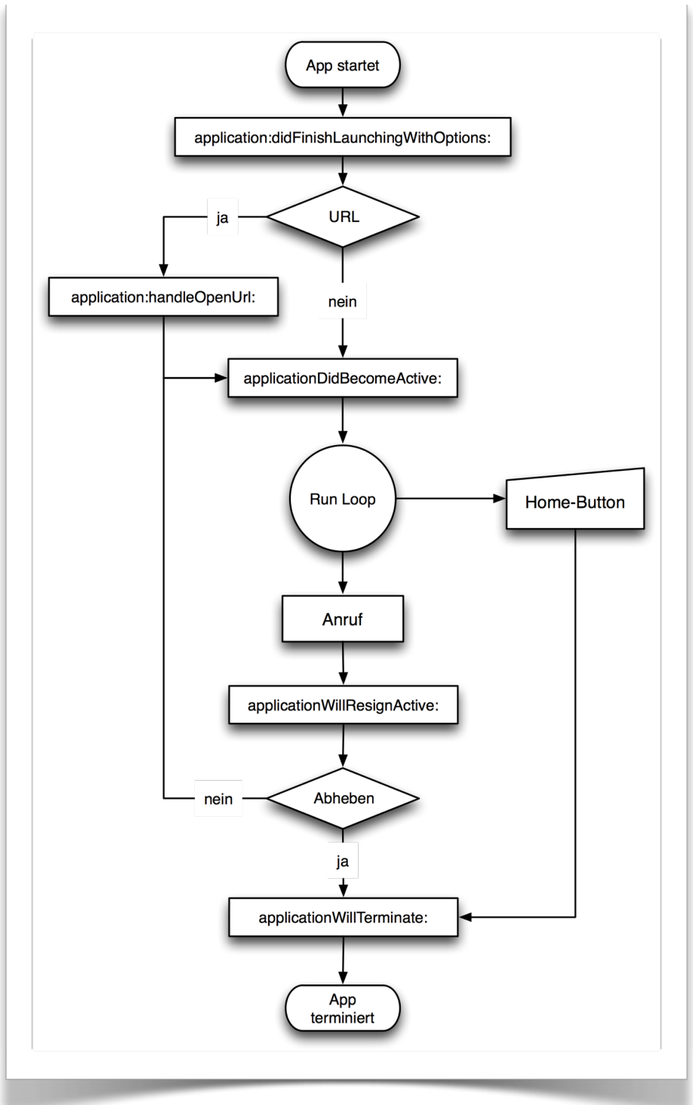
Dateisystem

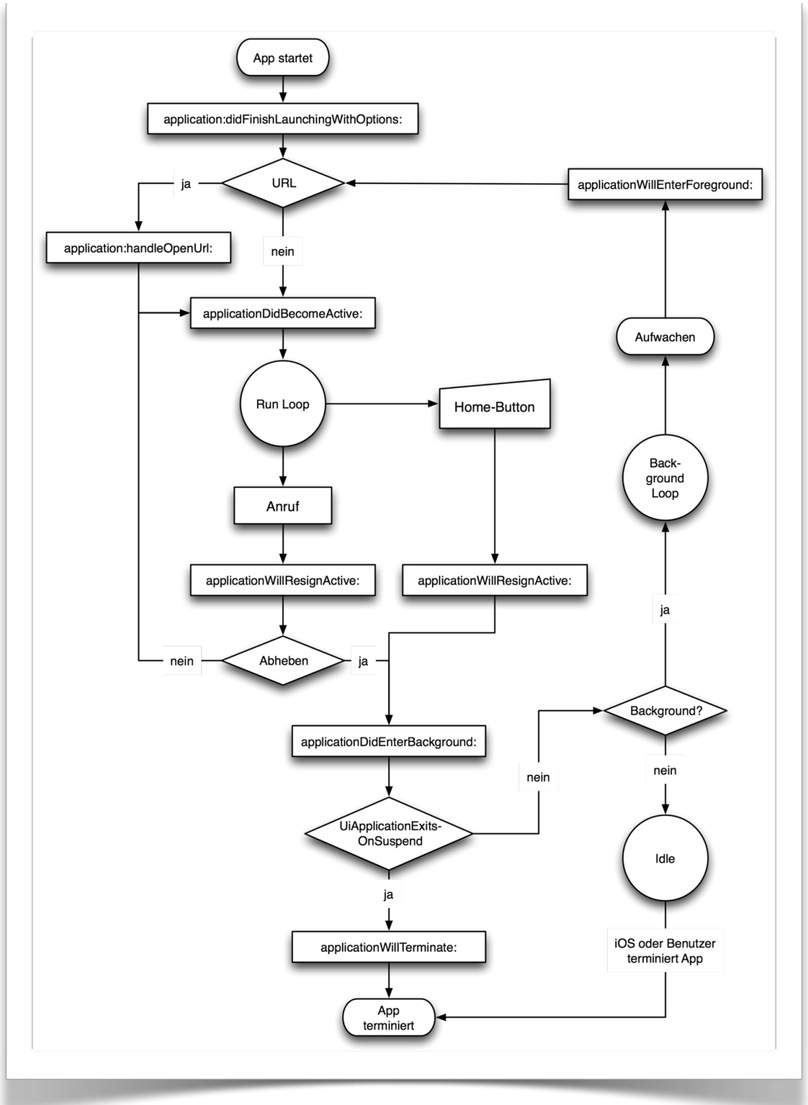
```
NSDictionary *attr = [NSDictionary dictionaryWithObject:

NSFileProtectionCompleteUnlessOpen forKey:NSFileProtectionKey];

if (![[NSFileManager defaultManager] setAttributes:attr
            ofItemAtPath:fileerror:&error]) {
```

Event handling



Event handling





Jailbreak-Erkennung

- / writeable
- /bin/bash
- /usr/sbin/sshd
- /private/var/apt
- …
- unsignierten Code ausführen - !(App Store)

Check /etc/fstab

```
NSString *fstab = [NSString stringWithContentsOfFile:@"/etc/fstab"
encoding:NSISOLatin1StringEncoding error:&error];

NSRange textRange;

NSString *substring = [NSString stringWithString:@" / hfs rw"];
textRange =[[fstab lowercaseString]
rangeOfString:[substring lowercaseString]];

if(textRange.location != NSNotFound)
…
```

Verify previous check

```
 if([localFileManager createDirectoryAtPath:@"/tmp/..."
withIntermediateDirectories:NO attributes:nil error:&error]){
```

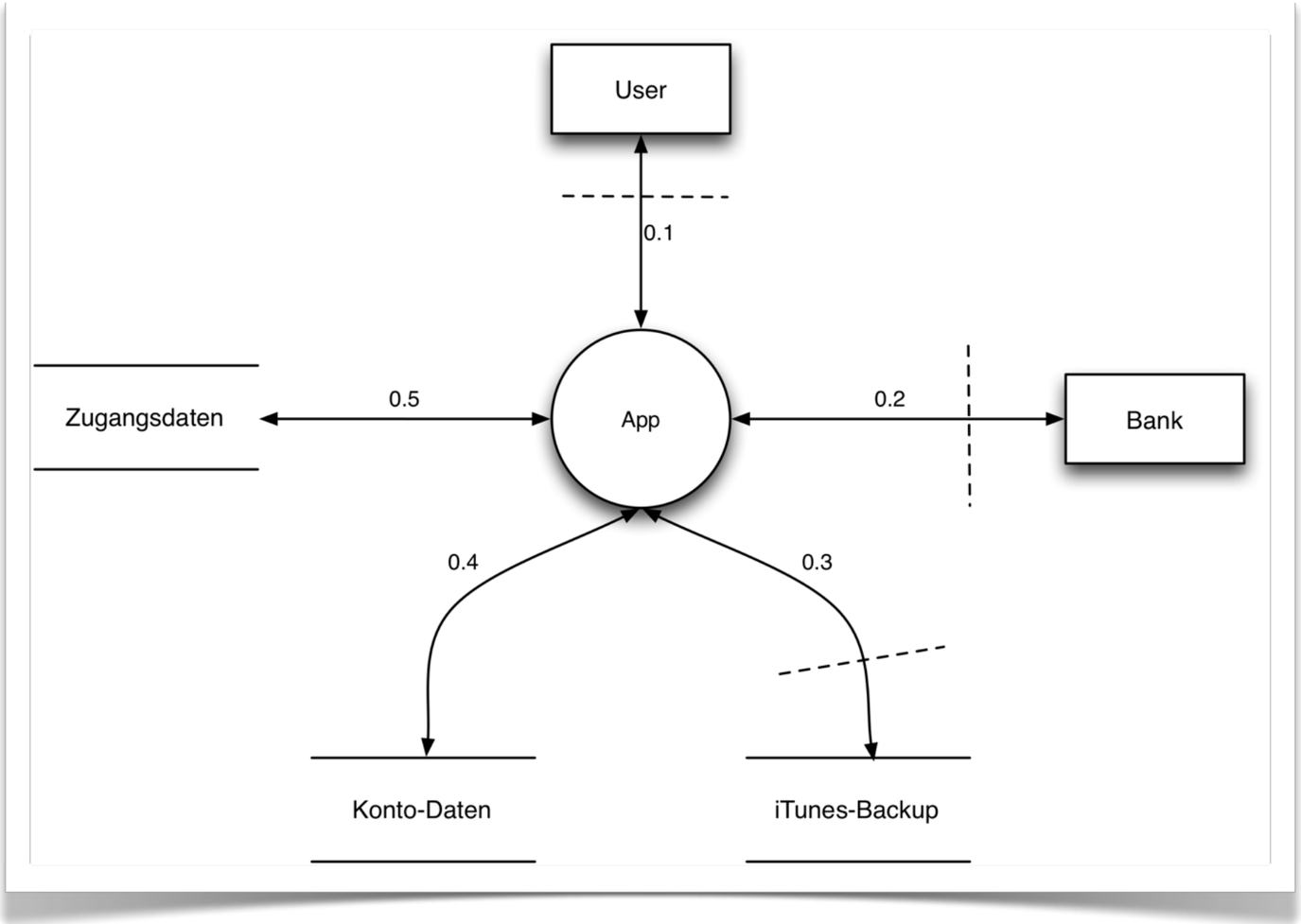
Netzwerk-Sicherheit

- totgeglaubte Angriffsvektoren leben wieder auf
- Verschlüsselung ist nicht teuer
- SSL is not dead - it just smells funny
  - auf iOS-Version prüfen (> 4.3.4)
  - ggf. mit Protectionspaces arbeiten

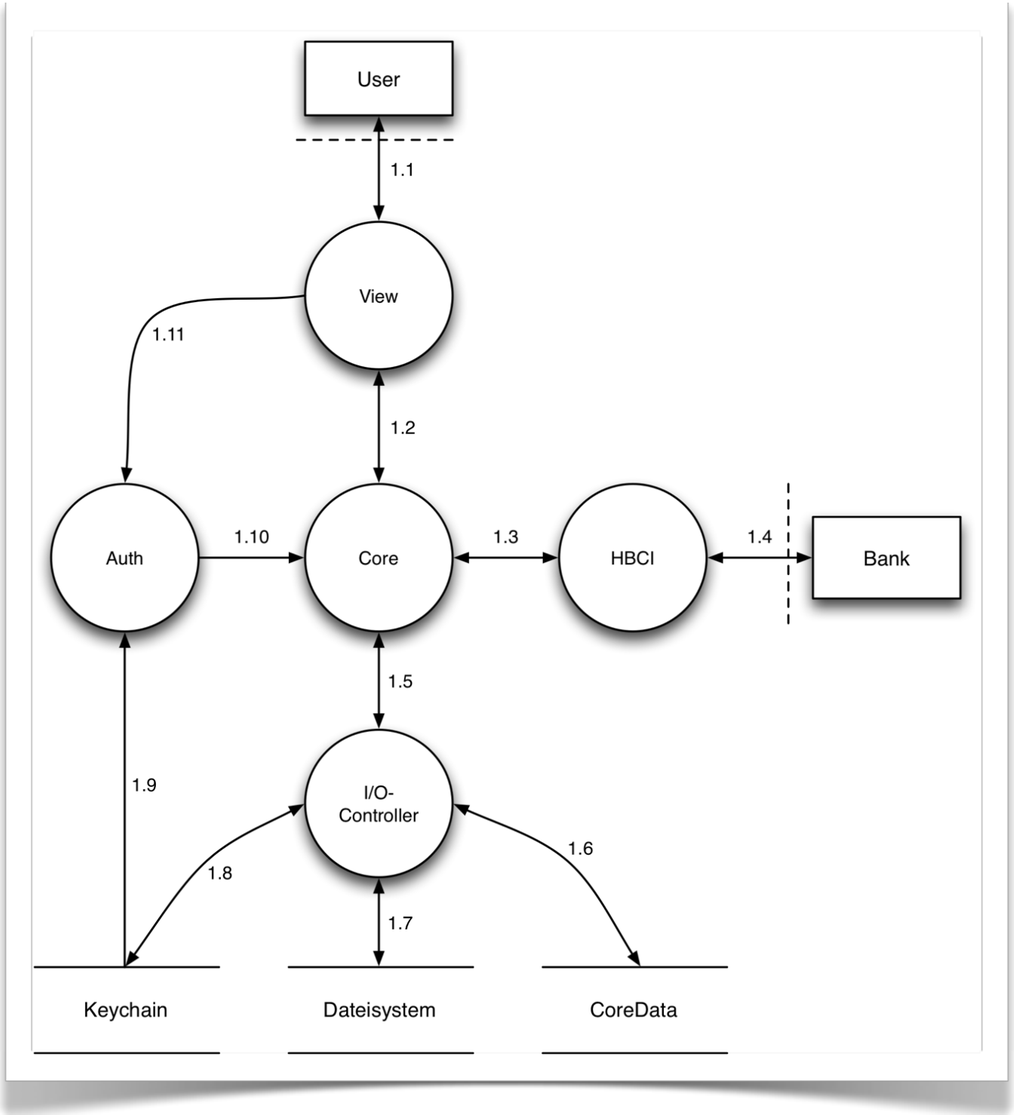
Threat Modeling

- methodischer Ansatz um spezifische Bedrohungen für $TARGET zu identifizieren
- Input:
  - Datenfluss
- Output:
  - Liste spezifischer Bedrohungen
    - priorisierte Liste potentieller Sicherheitslücken

Kontextdiagramm



Ebene 1


User
1.1
View
1.11
1.2
Auth
1.10
Core
1.3
HBCI
1.4
Bank
1.5
1.9
I/O-
Controller
1.8
1.6
1.7
Keychain
Dateisystem
CoreData

STRIDE

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privileges

Spoofing

Bedrohung	Maßnahme
Ein unbefugter Benutzer greift auf die App zu und gibt sich somit als befugter Benutzer aus. Zugriff auch über tethered Jailbreak möglich.	Zugang zur App über eigenen Authentisierungsmechanismus. Datenablage in Keychain über entsprechende Attribute schützen.
Ein Angreifer leitet die Anfragen der App über einen man-in-the-middle-Angriff auf einen eigenen Server um und gelangt auf diese Weise in den Besitz der Zugangsdaten für das Online-Banking.	Prüfen des SSL-Zertifikates der Bank auf Gültigkeit und Authentizität. Aktuelle iOS-Version beim Start der App prüfen und Benutzer informieren, wenn iOS < 4.3.5 installiert ist (Webkit-SSL-Bug).

Tampering

Bedrohung	Maßnahme
Ein Angreifer kann über eingeschleusten Schadcode die Kommunikation zur Bank manipulieren und dadurch Zugriff auf Transaktionen erlangen.	Beim Start der App auf das Vorhandensein eines Jailbreak prüfen und im positiven Fall den Benutzer über die damit verbundenen Gefahren informieren. Weitere Maßnahmen (z.B. Deaktivierung der App) mit Auftraggeber (Bank) abstimmen.
Ein Angreifer kann über manipulierten Netzwerkverkehr oder manipulierte Daten Schwachstellen in C-Funktionen ausnutzen und auf diese Weise schädliche Aktionen im Kontext der App ausführen.	Ausschließliche und korrekte Verwendung sicherer C-Funktionen, bzw. ausschließliche Nutzung der Cocoa-API, um Angriffsklassen gegen C-Schwachstellen zu eliminieren. Ggf. Secure-Coding-Guidelines für C bemühen (z.B. MISRA-C).

Repudiation

Bedrohung	Maßnahme
Ein Angreifer kann Aktionen gegen eine App oder ein aus einer App und einem Server bestehenden Kommunikationssystem durchführen, ohne dass diese Aktionen durch eine Logging-Mechanismus registriert werden.	Schreiben aussagekräftiger Log- Dateien und Speicherung der Daten an sicherer (dritter) Stelle.
Ein Angreifer kann sensible Daten aus Log-Dateien auslesen.	Log-Dateien sollten gegen unbefugten Zugriff geschützt sein und keine sensiblen Daten wie z.B. Passwörter oder Transaktionsinterna (Kontostand, etc.). enthalten.

Information disclosure

Bedrohung	Maßnahme
Unverschlüsselt im Dateisystem abgelegte, sensible Daten gelangen über das iTunes-Backup unverschlüsselt auf die Festplatte des Desktop-Rechners.	Technisch: keine unverschlüsselten Daten ablegen (NSFileManager-Attribute verwenden). Organisatorisch: in den AGB verschlüsseltes Backup fordern.
Ein Angreifer erlangt durch Mitschneiden des Netzwerkverkehrs in den Besitz möglicherweise sensibler Daten (z.B. Anmelde- oder Transaktionsdaten).	Verwendung angemessener Verschlüsselung in Verbindung mit Authentizitäts- und Integritätsprüfung (SSL/TLS).

Denial of service

Bedrohung	Maßnahme
Die Eingabe fehlerhafter Zugangsdaten führt nach dem dritten fehlgeschlagenen Anmeldeversuch zur Sperrung des Kontos.	Nach ersten fehlgeschlagenen Anmeldeversuch weitere Versuche abbrechen und Benutzer informieren.
Ein Angreifer kann über die Importfunktion einer App eine Datei mit manipuliertem Format einschleusen und dadurch den Import- Parser der App lahmlegen.	Robuste Programmierung von Importfiltern und Parsern, um Angriffe über manipulierte Formate zu verhindern.

Evelation of privileges

Bedrohung	Maßnahme
Ein Angreifer mit Zugriff auf das iPhone kann das App-Passwort des Benutzers erraten und erhält auf diese Weise Zugriff auf alle Bank-Daten des Benutzer.	Implementierung einer Passwort-Richtlinie, die die Verwendung ausreichend starker Passwörter erzwingt (siehe dazu auch das Passwort-Tutorial von Heise).
Datenimport von einer nicht geprüften URL erlaubt einem Angreifer das Einschleusen von fehlerhaften Daten oder Schadcode.	Semantische Prüfung importierten Contents, Filterung auf potentiell schädliche Sonderzeichen, Verwenden von SSL, um Kommunikationspartner im Netz eindeutig identifizieren zu können.

SDL-ELEMENTE

- Sicherheit des Quelltextes
- sichere Entwicklungsumgebung
- Erzeugung von Testdaten
- Threat Model
- Umgang mit Tools, Sprachen und Plattformen
- Patchmanagement (auch Fremdcode)

Anforderungsanalyse

- Welche Daten muss ich verarbeiten/speichern?
- Wann müssen welche Daten zugänglich sein?
- Welche Bedrohungen wirken gegen meine App?
- Was muss ich in die Nutzungsbedingungen oder das Lastenheft schreiben?

Konkret

- Verwenden der Keychain (richtig und sicher)
- Verwenden von Protection Classes
  - Keychain, NSFileManager, Core Data, Sqlite, NSData
- Netzwerk-Sicherheit
- Event handling

Misc

- Finger weg von Speicher und Pointern
- Passwort-Policy
- Finger weg von Speicher und Pointern
- Sicherheitsattribute explizit verwenden
- Finger weg von Speicher und Pointern
- nichts (in C) bauen, was es (in Objective-C) bereits gibt
- Finger weg von Speicher und Pointern

Daumenregeln

- Keep it simple, stupid!
- Need to know
- Attack surface reduction
- Datensparsamkeit
- Secure by default
- Defense in depth
- Least privilege

Fragen?

- Antworten hier
- „*Apps entwickeln für iPhone und iPad*“, Galileo Press
- www.cocoaneheads.de