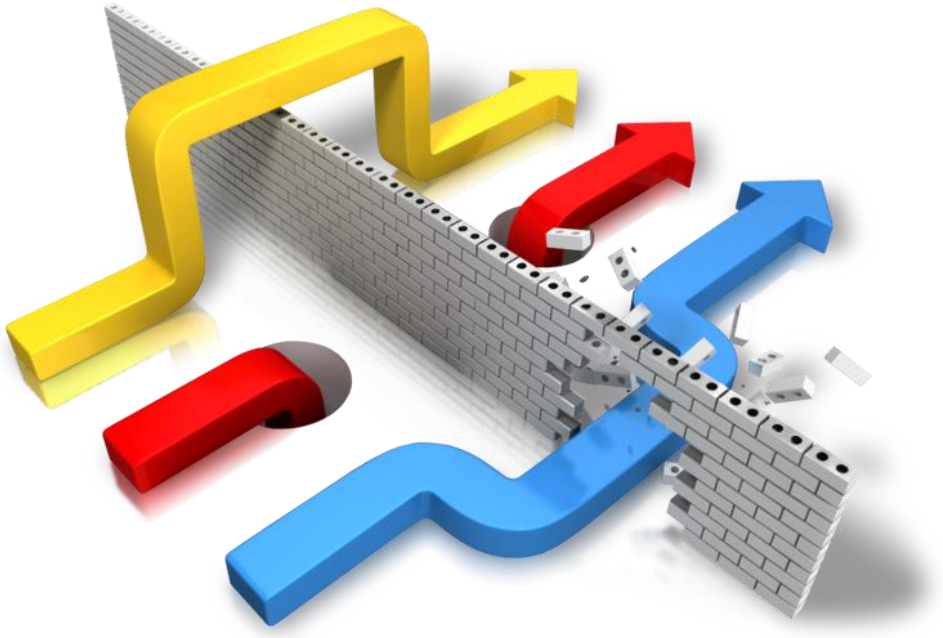
Fraunhofer Institute for Secure Information Technology

Entwicklung sichere Unternehmens-Apps: gut gemeint oder gut gemacht?

Dr. Jens Heider

Head of Department Testlab Mobile Security



Kundenanfragen zu mobiler Sicherheit

- Wie sicher sind iOS / Android im Vergleich zu Blackberry?
- Wie werden Credentials in iOS gespeichert? Ist das sicher?
- Wir möchten MDM Lösung XY verwenden. Wie sicher ist es?
- Ist es möglich iOS sicher im Unternehmensnetzwerk zu integrieren (e.g. VPN)?
- Wie sicher ist App XY? Wie speichert sie Daten?
- Wir entwickeln eine App für iOS / Android. Wie sicher ist unser Konzept / unsere Implementierung?

Platform Security Testing

Network Security Testing

Application Security Testing

Überblick
Vortragsthemen

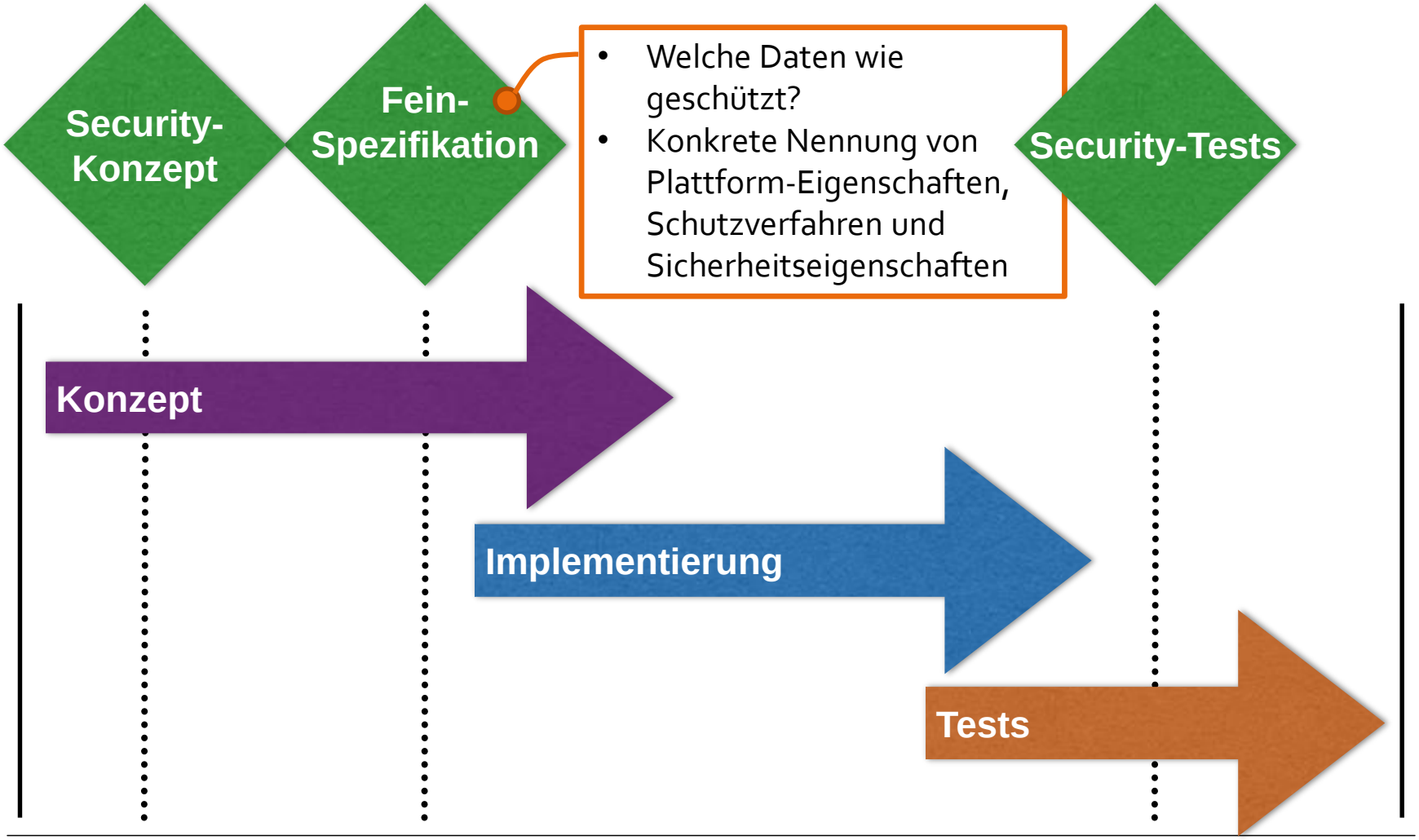
- Unterschied Softwareentwicklung vs. App-Entwicklung?
- Welche Bedrohungen bestehen für Unternehmens-Apps?
- Häufige sicherheitsrelevante Fehler in der App-Entwicklung?
- Beispiele: Wie sichert man Apps richtig?

Entwicklung sicherer Apps – Ein Problem?

- ***Mobile Plattformen bieten konfigurierbare sicherheitsrelevante APIs***
  - Sicherer lokaler Speicher / Cloud Speicher
  - Kommunikationsschutz
  - Kryptografische Funktionen
- ***Aber: fehlende Hilfestellung dies für spezifische Aufgaben sicher zu nutzen***
  - Nutzer Authentifizierung gegenüber Gerät / Server
  - Korrekter Schutz von Daten
  - Richtige Zertifikatsverwaltung
  - Berücksichtigung der Privatsphäre
  - ...

Schwächen beginnen häufig im Entwicklungsprozess

Wann kommt die Sicherheit?



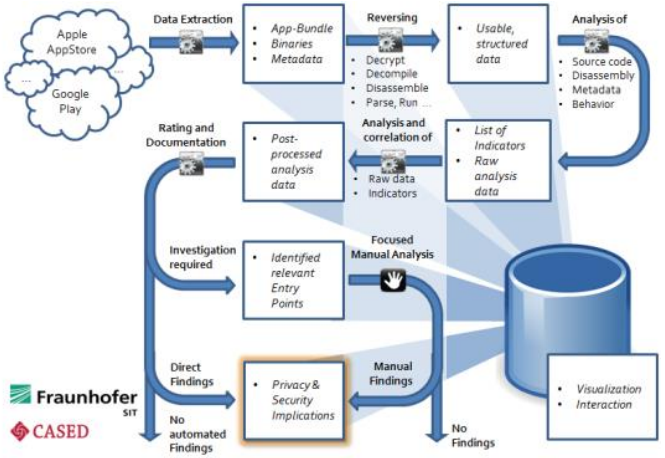
Praxis Tests: App Sicherheit

Investition in Ihre Zukunft

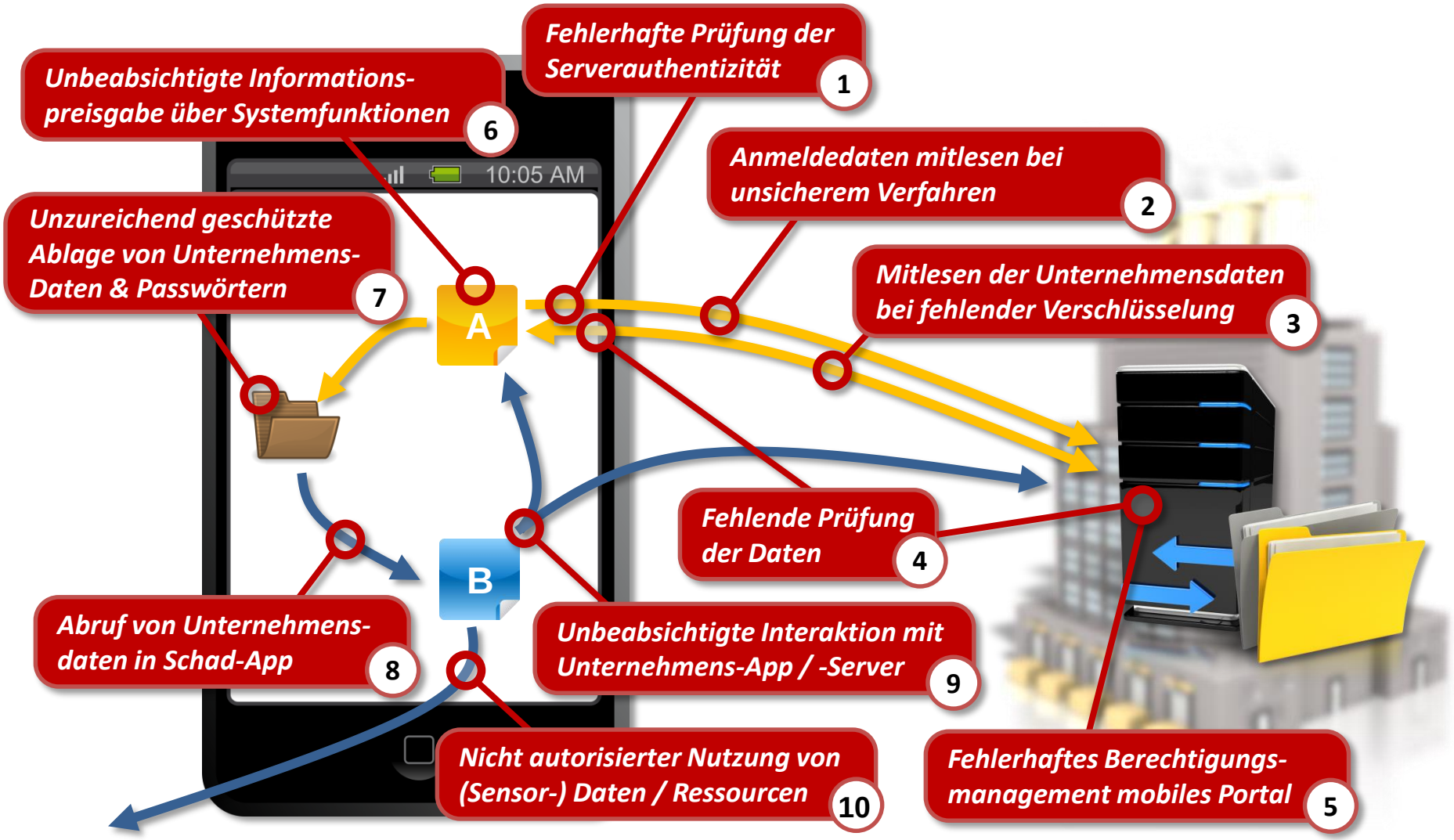
HESSEN

Investitionen für diese Entwicklung wurden von der Europäischen Union aus dem Europäischen Fonds für regionale Entwicklung und vom Land Hessen kofinanziert.

- ***Erfahrung aus zahlreichen manuellen Tests***
  - Sicherheitsqualität kaum verbreitet
  - Schutzpotential wird kaum genutzt
  - Schutzmechanismen teilweise (zu) komplex
- ***Massentests***
  - Framework zur Analyse in Betaphase
  - Testet auf bekannte Schwachstellen und verdächtiges App-Verhalten
  - Basiert auf Wissen aus manuellen Tests und der konzeptionellen Forschung bei CASED



Verwundbarkeiten von Unternehmens-Apps



Beispiel: Unsichere SSL Nutzung

- Gut gemeint:
  - eigene Implementierung „sicherer" oder performanter
  - Sicherheitsimplementierung „mit Suchmaschine"
- Fehler u.a.:
  - Verzicht auf SSL oder Nutzung proprietärer Verschlüsselung
  - Fehlende oder unvollständige Verifikation des Zertifikats
  - Selbst signierte Zertifikate für Backend-Systeme

Real-Android-Beispiel: Smali Repräsentation einer unsicheren Zertifikatsprüfung in Banking App

```
.method public onReceivedSslError
(Landroid/webkit/WebView;Landroid/webkit/SslErrorHandler;Landroid/net/http/SslError;)V
.locals 0
.parameter "iwv"
.parameter "sslh"
.parameter "ssle"
.prologue
.line 118
invoke-virtual {p2}, Landroid/webkit/SslErrorHandler;->proceed()V
.line 119
return-void
.end method
```

Beispiel: Sicherung Datentransport
TLS/SSL gut gemeint

Bitte nicht!

ciphersuite = TLS_DH_anon_WITH_AES_256_CBC_SHA;

Gut genug?
Nur Verschlüsselung! Keine Authentifizierung!
⇨ Man-in-the-Middle Angriffe möglich!

```
NSString * publish_url = @"https://example.com/publish/";
NSMutableURLRequest *request =
            [NSMutableURLRequest requestWithURL:
            [NSURL URLWithString: publish_url]];
[request setHTTPBody:data];
[request setHTTPMethod:@"POST"];
[[NSURLConnection alloc] initWithRequest:request
delegate:self];
```

Beispiel: Sicherung Datentransport
TLS/SSL Standardfall

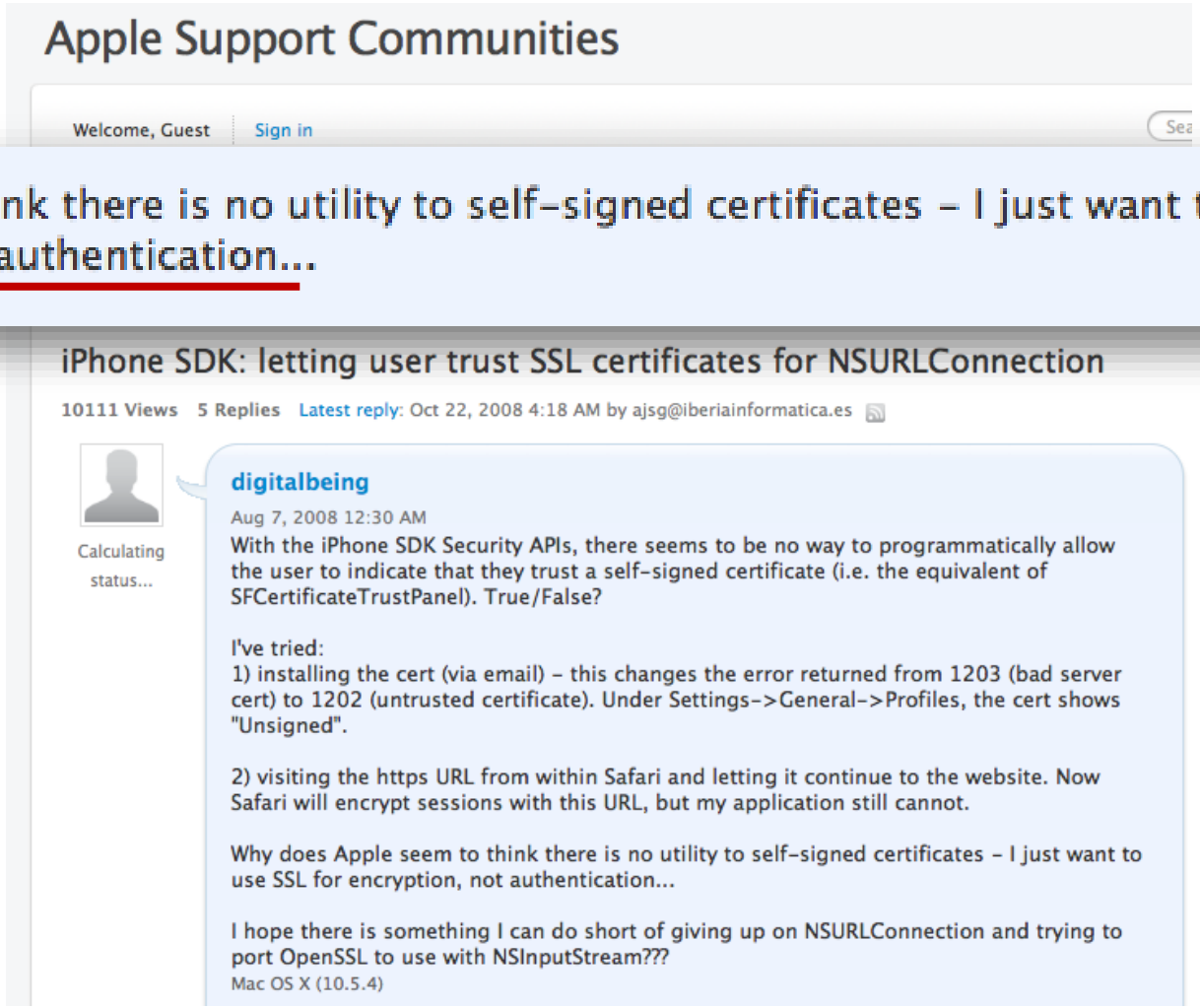
ciphersuite = TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
...
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA

Verschlüsselung und zertifikatsbasierte Authentifizierung
Aber: Funktioniert nicht mit selbst signierten Zertifikaten!

```
NSString * publish_url = @"https://example.com/publish/";
NSMutableURLRequest *request =
        [NSMutableURLRequest requestWithURL:
        [NSURL URLWithString: publish_url]];
[request setHTTPBody:data];
[request setHTTPMethod:@"POST"];
[[NSURLConnection alloc] initWithRequest:request
delegate:self];
```

Beispiel: Sicherung Datentransport

Fehlendes Wissen



Apple Support Communities

Welcome, Guest | Sign in

Why does Apple seem to think there is no utility to self-signed certificates – I just want to use SSL for encryption, not authentication...

iPhone SDK: letting user trust SSL certificates for NSURLConnection

10111 Views 5 Replies Latest reply: Oct 22, 2008 4:18 AM by ajsg@iberiainformatica.es

Calculating status...

digitalbeing

Aug 7, 2008 12:30 AM

With the iPhone SDK Security APIs, there seems to be no way to programmatically allow the user to indicate that they trust a self-signed certificate (i.e. the equivalent of SFCertificateTrustPanel). True/False?

I've tried:
1) installing the cert (via email) – this changes the error returned from 1203 (bad server cert) to 1202 (untrusted certificate). Under Settings->General->Profiles, the cert shows "Unsigned".

2) visiting the https URL from within Safari and letting it continue to the website. Now Safari will encrypt sessions with this URL, but my application still cannot.

Why does Apple seem to think there is no utility to self-signed certificates – I just want to use SSL for encryption, not authentication...

I hope there is something I can do short of giving up on NSURLConnection and trying to port OpenSSL to use with NSInputStream???

Mac OS X (10.5.4)

Beispiel: Sicherung Datentransport
Gefährlicher Rat

RickMaddy

This solved my question Re: iPhone SDK: letting user trust SSL certificates for NSURLConnection

Aug 7, 2008 12:49 AM (in response to digitalbeing)

Level 4 (1,320 points)

I had the same problem. After some research I came across a solution that works but it's not officially supported. In your implementation class using NSURLRequest at the following lines:

```
@implementation NSURLRequest(DataControl
+ (BOOL)allowsAnyHTTPSCertificateForHost:(NSString *)host
{
 return YES; // Or whatever
}
@end
```

This overrides this method. Simply returning YES here is potentially dangerous of course so you may need to return YES in a more controlled fashion. But it should get you further.

15" MacBook Pro Intel Core Duo 2.0GHz, 2GB RAM, Mac OS X (10.5.4), iPod touch 16GB, iPhone 16GB

Like (0)

Keine Prüfung der Gegenstelle!

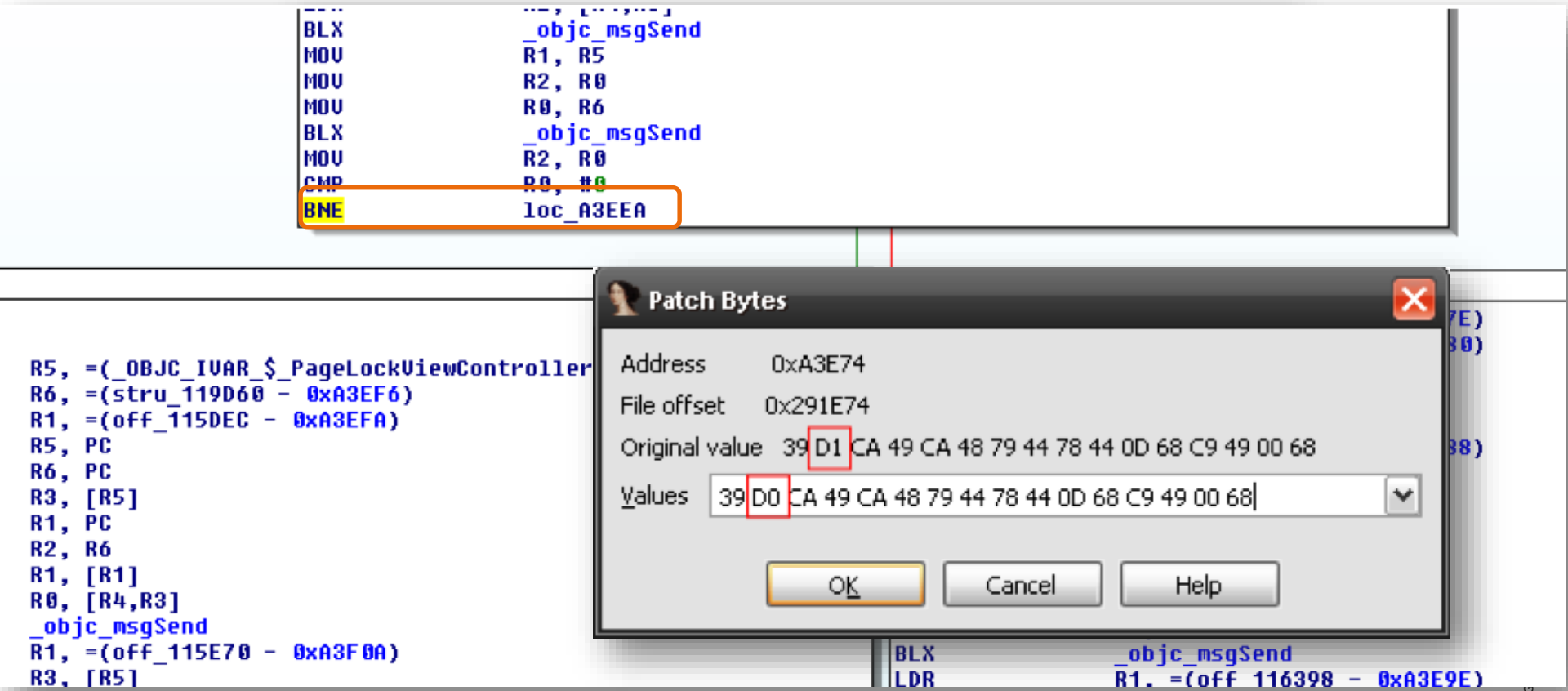
Bitte nicht!

Beispiel: Sicherung Datentransport
MITM via Proxy

- HTTPS Anfragen können von einem Proxy terminiert werden
- Zertifikate verhindern das doch?
  - Selbstsigniertes Zertifikat
  - kann dynamisch erzeugt werden
  - enthält die meisten Felder des korrekten Zertifikats
- Angriff leicht wenn
  - Zertifikate nicht korrekt validiert werden
  - z.B. *[NSURLRequest setAllowsAnyHTTPSCertificate:YES forHost:[url host]]*;

Beispiel: Fehlerhaftes Authentifizierungsschema
Anmeldung an App

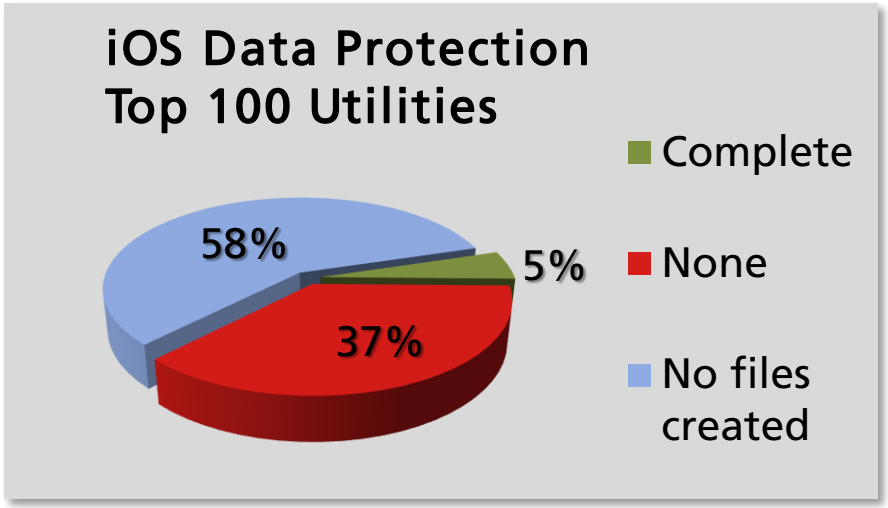
Beispiel: Fehlerhaftes Authentifizierungsschema
Anmeldung an App – De-Assemblierung



Beispiel: Fehlerhaftes Authentifizierungsschema
Was war der Fehler?

- Authentifizierung nicht gebunden an Verschlüsselung
- Zugriff wird gewährt, wenn Passwort korrekt ist
  - *if ([pwdTextField isEqualToString:storedPassword])* ergibt BNE Assembler Operation
  - Ändern von BNE in BEQ invertiert die Funktion zu: *if (!([pwdTextField isEqualToString:storedPassword]))*
- Ändern eines Bits der App von 1 auf 0 ermöglicht Zugriff auf Daten mit beliebigen Passwort (außer dem richtigen)

Beispiel: Fehlender Schutz der gespeicherten Daten



Appicaptor Analyse, Deutscher App Store, 8.5.2013

- iOS Funktionalität um Daten verschlüsselt zu speichern
  - Verwendung des Passcodes als Nutzergeheimnis für Verschlüsselung + Offline Bruteforce Schutz
  - Schutz für verlorene/gestohlene Geräte
  - Kaum genutzt von System Apps
- Untersuchungsergebnisse automatisierter Tests
  - Auch kaum in analysierten 3rd Party Apps genutzt
  - Von 8 Apps mit Office Datei Unterstützung, nur 2 mit Data Protection

Zusammenfassung

- ***Security-Konzept:***
  - so früh wie möglich formulieren
  - Schutzgüter, Schutzziele, Angreifer und Angriffe präzise formulieren
  - auch auszublendende Angriffe formulieren und argumentieren
  - Je nach Schutzbedarf auch auf Vollständigkeit überprüfen lassen

Zusammenfassung (II)

- ***Feinspezifikation:***
  - vor der Implementierung formulieren
  - spezifisch für Zielplattform formulieren
  - Welche Methoden helfen gegen welche Angriffe / Angreifer
  - Wie müssen die Methoden auf der Zielplattform implementiert werden
  - Bei höheren Anforderungen Spezifikation u.a. auf Wahl der geeignetsten Mittel überprüfen lassen

Zusammenfassung (III)

- ***Implementierung:***
  - Sicherheitsrelevante Methoden früh genug implementieren
  - an Spezifikation messen
  - Entwickler-Schulung nicht Google überlassen
  - Externe Prüfung schafft Transparenz

Kontakt

Dr. Jens Heider

Rheinstr. 75
64295 Darmstadt
Germany

E-Mail: jens.heider@sit.fraunhofer.de
Web: http://www.sit.fraunhofer.de
http://testlab.sit.fraunhofer.de