Sikkerhet i mobile applikasjoner
OWASP Top 10 Risks Mobile[P]

OWASP Norway, 24.april 2012

Ståle Pettersen

Om meg

- Ståle Pettersen
- Java-utvikler og sikkerhetsentusiast
- BEKK
- OWASP

twitter.com/kozmic

Mobile egenskaper

- Mobiler blir mistet og stjålet
- Mange applikasjoner kjører samtidig
- Autentisering og autorisasjon
- Bruker ofte mange remote web services
- Tredjeparts tjenester
- Mange mobile platformer, ingen gjør ting likt - økt kompleksitet
- Mye sensitiv informasjon som applikasjoner kan ha tilgang til
    - Kontaktinformasjon
    - Bilder
    - Video
    - SMS
    - GPS-lokasjon
    - Kredittkort (NFC)
    - Tilkoblede enheter (Bluetooth)

Backup-er av telefoner sprer disse dataene

Apple iOS

- Hver applikasjon har sin egen sandbox, men kan kommunisere med hverandre vha schema
- Backuper er ikke kryptert (kan konfigureres av bruker til å være kryptert)
- iOS 5 har kryptert filsystem
- Låsekode enkel å knekke (2-6 forsøk i sekundet ved bruk av API, ingen maks antall forsøk)
- Kun tekst passord er sikkert

Top 10 Mobile Risks, Release Candidate v1.0

BH EU 2012 - Jack Mannino, Zach Lanier og Mike Zusman

1. Insecure Data Storage

2. Weak Server Side Controls

3. Insufficient Transport Layer Protection

4. Client Side Injection

5. Poor Authorization and Authentication

6. Improper Session Handling

7. Security Decisions Via Untrusted Inputs

8. Side Channel Data Leakage

9. Broken Cryptography

10. Sensitive Information Disclosure

Burde være final 22.april, 60 dager etter siste kommentar

1. Insecure data storage

- Sensitive data som ikke er beskyttet (inkludert logger)
- Data lokalt, men også data synkronisert mot cloud-tjenester

- Konsekvenser
  - Konfidensielle data tapt
  - Brukernavn, passord og tokens på avveie
  - Sensitive data på avveie

1. Insecure data storage

- "Husk meg" - Lagring av brukernavn og passord i en fil som alle andre applikasjoner kan lese
  - Android: Ikke bruke MODE_WORLD_READABLE, bruk MODE_PRIVATE
  - iOS: Kan ikke lagre globale filer, men kan eksponeres i backup-filer

1. Insecure data storage

- Anbefales å kun lagre nødvendig data
- Ikke bruk lagringsområder som ikke er beskyttet av sikkerhetsmekanismer (SD-card)
- Bruk sikkerhetsmekaniser innebygd i platform som filkrypterings API-er og keychain
- Pass på at filer ikke kan leses og skrives globalt

2. Weak Server Side Controls

- Backend-tjenester
- Tjenester kan ikke stole på data fra klienten
- Samme regler som for dagens webapplikasjoner

Inngår også i OWASP Top 10, OWASP Cloud Top 10. Autentisering osv er litt anerledes, men det kommer vi tilbake til.

3. Insufficient Transport Layer Protection

- Ingen eller svak kryptering ved transport av sensitive data
- Klient ignorerer sertifikat-feil (utgått, feil domene, ugyldig format), eller fallback er HTTP
- Eksempel på dette er Google Calendar og Facebook som sendte autentisering som HTTP-header over HTTP

3. Insufficient Transport Layer Protection

- Pass på at all sensitive data sendes kryptert
- Dette inkluderer mobilnett, trådløstnett, NFC og bluetooth

4. Client Side Injection

- Applikasjoner embedder nettleser bibliotek (webapp, hybrid og native)
- Klassiske sårbarheter: Javascript-injection, SQL-injection, HTML-injection
- Men kan også gi tilgang til SMS, telefon, betalingsystemer
- Eksempel: Amazon kindle

Amazon Kindle ble jailbreaket med klientside Javascript ved å embedded JS i MP3 ID3 tag. I native apps (Media Player) har en full tilgang JS API som ROOT via debug funksjoner.

4. Client Side Injection

- Ikke stolt på brukerinput - valider og encode
- I hybrid app: Minimer native API som blir eksponert til webapp
- Bruk prepared statements

```
SQLiteStatement stmt =
   db.compileStatement("SELECT * FROM Country WHERE code = ?");
stmt.bindString(1, "US");
stmt.execute();
```

5. Poor Authorization and Authentication

- Ikke stol på verdier som ikke kan endres (UUID, IMSI, IMEI, MAC, ICCID)
- Hardware identifikatorer forandrer seg ikke (finn.no?)
- Eksempel: AT&T - 114 000 eposter eksponert

AT&T lakk 114 000 epost adresser

5. Poor Authorization and Authentication

- Out-of-band (f.ks SMS) gir lite ekstra verdi fordi det er samme enhet
- Ikke bruk enhetsid eller SIM-kort ID alene til autentisering (men kan være bra til dybdesikkerhet)

6. Improper Session Handling

- Brukere forventer at en ikke må logge inn hver gang en starter en applikasjon
- Dette gjør at sesjoner må leve lenger - hvis en ønsker brukervennlighet

- Typisk sesjon implementeres vha
  - HTTP cookies
  - OAuth tokens
  - SSO autentisering (f.eks Facebook connect)
- Hardware identifikater alene er ikke sikkert

6. Improper Session Handling

- Anbefaler å ikke sende brukernavn og passord for hver request (da må dette lagres et sted, som kan lekke informasjon)
- Bruk OAuth 2.0 tokens - Kan fjerne tilgang til enheter
- La brukeren autentisere seg med jevne mellomrom så hvert token ikke har for lang levetid
- Tokens må være kryptografisk sikre (ikke prøv å skriv egen random-generator)

7. Security Decisions Via Untrusted Inputs

- iOS: Misbruk av Schema

```
skype://98888888?call
```

- Android: Misbruk av Intents
- Onde nettsider kan kjøre applikasjoner på telefon uten at brukeren blir spurt
- Kan utnytte sårbarheter i applikasjoner fra nettsteder
  - Hvis skype har XSS i håndtering av skype:// kan en stjele info eller sende SMS

7. Security Decisions Via Untrusted Inputs

- Sjekk hvilken applikasjon som prøver å utføre en handling
- Spør brukeren om dette er ønskelig (slik iOS håndterer tel-schema i Safari)

8. Side Channel Data Leakage

- Sensitive data blir lagret på usikre steder (direkte eller indirekte)
  - Web browser cache inkludert HTML5 storage
  - Logger
  - Midlertidlig kataloger (kan oppstå f.eks når applikasjonen kræsjer)
  - Backup-filer
  - Synkroniseringsprogrammer / Cloud-tjenester
- Forstå også hvordan tredjeparts bibloteker lagrer sensitive data
- Facebook sin mobilapp lagret token i ren tekst (60 dagers gyldighet)

8. Side Channel Data Leakage

- Ikke logg sensitive data som passord
- Sett no-cache headere for embedded nettsider med sensitive data
- Ikke lagre sensitive data i HTML5 storage
- Overvåk applikasjonen og se hvilke filer som skrives (tredjeparts biblioteker dokumenter dette sjeldent, så sjekk selv)

9. Broken Cryptography

- Krypto brukt feil
  - Sterk crypto algorytme, men lagrer nøkkel på disk i plaintekst
  - Egen algorytme og implementasjon som ikke er sikker
- Encoding (f.eks base64) og serialisering er ikke krypto

9. Broken Cryptography

- Å lagre nøkkelen på disk gir liten sikkerhetsverdi
- Ikke finne på egen krypto, du kommer til å gjøre feil
- Utnytt sikkerhetsmekanismene i platformen! (iOS og Android: Keychain)

10. Sensitive Information Disclosure

- Hardkodede verdier i applikasjonen
- Applikasjoner kan reverse engineeres (ikke alltid like lett.. men mulig)
- Private nøkkler, passord, sensitive algorytmer og "hemmelige parametere"

OWASP Top 10 Mobile Risks

- RC 1.0 i dag, men 1.0 final er rett rundt hjørnet
- Trenger flere frivillige, så kan du mobilsikkerhet, eller har lyst til å lære, er det bare å bidra! :)

Case study: Password managers

- ElcomSoft evaluerte 17 "passord managers" (iOS og Blackberry)
- 16 hadde implementert egen sikkerhet
- 7 hadde i praksis ingen kryptering
- 2 av 17 hadde utnyttet sikkerhetsmodellen til telefonen

http://www.elcomsoft.com/WP/BH-EU-2012-WP.pdf

Case study: Password managers

- My Eyes Only™ - Secure Password Manager
  - Bruker iOS sin innebygde keychain til å lagre RSA public og privat keys
  - Dessverre lagres også disse nøklene utenfor keychain rett på disk

Case study: Password managers

- DataVault Password Manager
  - Bruker iOS sin innebygde keychain
  - Den sikrest av password managere som ble evaluert
  - Enkleste å implementere :)

Referanser

- http://www.slideshare.net/JackMannino/owasp-top-10-mobile-risks
  - Jack Mannino
  - Zach Lanier
  - Mike Zusman

- https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Controls
- http://software-security.sans.org/blog/2010/11/08/insecure-handling-url-schemes-apples-ios/
- http://www.grc.com/sn/sn-347.pdf
- Hacking and Securing iOS Applications - O'Reilly Media
- http://nelenkov.blogspot.com/2011/11/using-ics-keychain-api.html