Marc Blunsdon
Jan Kastbjerg

Sikkerhed

Table of Contents

Intro 2
Fingerskan 2
Kryptering mellem database og mobilappen (RestAPI) 3
Kryptering af facilitetsnøglen og/eller Bluetooth kommunikationen 4
Reference 5

Marc Blunsdon
Jan Kastbjerg

Intro

Dette dokument beskriver systems sikkerheds mangler med en mindre analyse og hvordan de eventuelt kan implementeres.

Fingerskan

Årsagen til at anvende fingerskan var et ekstra sikkerhedslag til login, så selv om andre fik adgang til brugernavn og password, ville de ikke kunne logge ind.

På Figur 1 kan man se de minimumskrav der er for at kunne implementere fingerskan, hvis man anvender de tilgængelige guides [1] og [2], med disse guides er det relativt nemt at implementere fingerskan i ens app.

- **The device is running Android 6.0 or higher**
- **The device features a fingerprint sensor**
- **The user has granted your app permission to access the fingerprint sensor.**
- **The user has registered at least one fingerprint on their device.**

Figur 1 Minimumskrav til fingerskan [1]

Med nok viden på området, burde det være muligt at implementere en fingerskanning, hvor man i brugerens oplysninger på hjemmesiden, også ville kunne gemme et fingerskan, dog er android telefoners fingerskannings setup, ikke designet til det brugsscenarie som ønskes implementeret i vores system.

Måden androids fingerskanning er sat op, er gennem følgende flow:

1. Brugeren godkender at fingerskanning kan bruges.
2. Brugeren opretter et fingerskan.
3. Fingerskannet bliver krypteret.
4. Det krypterede fingerskan bliver gemt i androids keystore, på telefonen.
5. Alle fremtidige fingerskanninger bliver matchet med printet i keystore.

Problemet ved dette, i hensyn til dette projekt, er at der ikke kan manipuleres med keystore igennem en applikation, hvilket videre betyder, at et fingerskan ikke kan blive ”hentet ud” af androids keystore [3], hvilket er et sikkerhedsdesign fra androids udviklere, for at et fingerskan ikke kan blive ”stjålet”.

En idé for at kunne omgå overstående beskrevet problem med keystore, ville det muligvis kunne løses ved at undgå brug af keystore, så der ikke bliver matchet op med et gemt fingerskan på telefonen, dog igen for at dette kan være en variabel idé ville det kræve en dybere viden, om hvordan fingerskannings processen præcis virker.

Det er derfor ikke muligt, inde for projektets tidsforløb, at implementere fingerskan med ønsket funktionalitet for at øge sikkerheden for, at det er den korrekte, bruger der anvender systemet.

Marc Blunsdon
Jan Kastbjerg

Kryptering mellem database og mobilappen (RestAPI)

Systemet bruger en RestAPI på hjemmeside til at kommunikere mellem mobilappen og databasen. RestAPI bruger JSON formation til at strukturere beskeden.

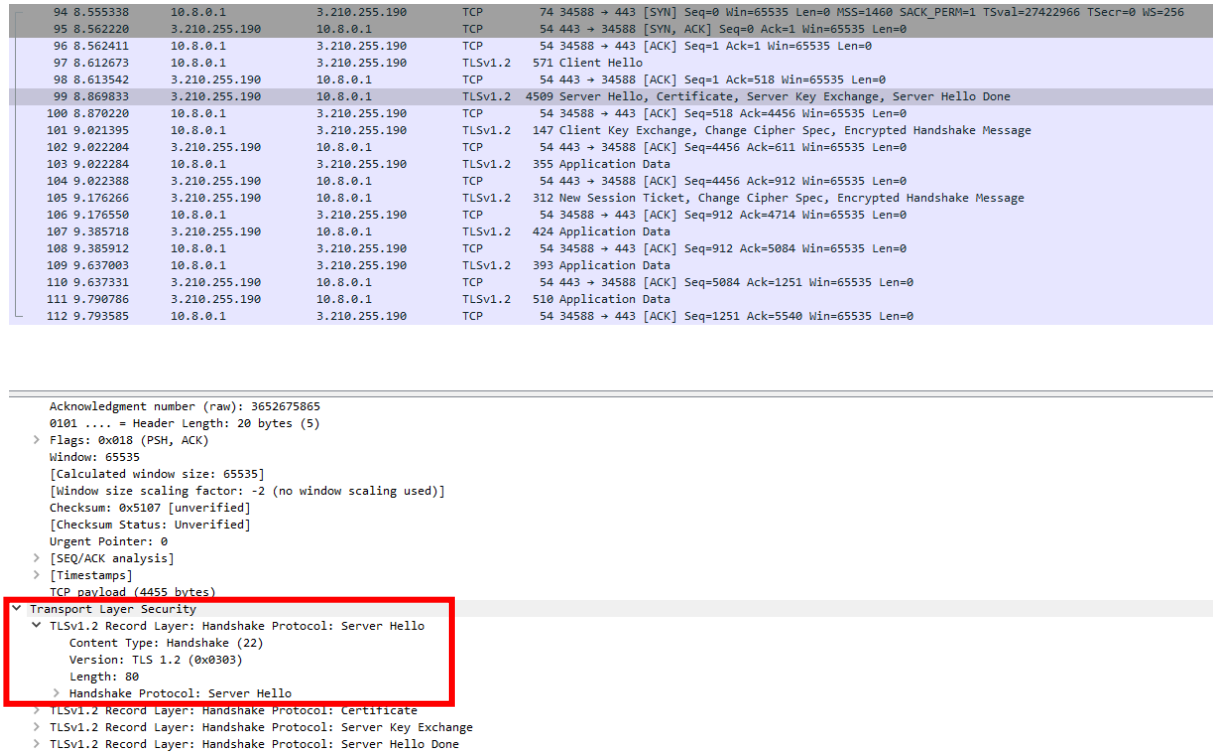
Den nemmeste og hurtigste måde at implementere en kryptering af kommunikationen er ved anvende SSL.

Måden SSL-scenariet vil virke på, er at hjemmeside serveren har en ”public” nøgle, og en ”private” nøgle, hvor appen så skal have en ”kopi” af ”public” nøglen, hvor der så fremover kan blive lavet et handshake mellem app og restAPI’en ved at tjekke public nøglen op mod private nøglen. Når der er lavet et handshake mellem app og restAPI, kan der sendes beskeder med ”Public-key kryptografi”.

En faldgrube ved at bruge SSL på denne måde er at så snart public nøglen er kendt, vil den kunne misbruges af andre apps, til fx at opsnappe og dekode beskeder sendt fra restAPI’en, dog kan dette til dels undgås på 2 måder:

1. have mere end 1 nøgle, hvor der bliver roteret imellem nøglerne.
2. Ved at skifte nøglen ud regelmæssigt.

SSL-metoden er også den metode der bliver brugt i projektet, som kan ses på Figur 2.



Figur 2 wireshark

Dog er det ikke nødvendigt at bruge SSL til at kryptere med, men der kan derimod bruges egne lavet kryptering eller andre anerkendte protokoller/kryptering end lige SSL.

Marc Blunsdon
Jan Kastbjerg

Kryptering af facilitetsnøglen og/eller Bluetooth kommunikationen

For at beskytte mod ”man-in-the-middle-attack”, ville det have været mest optimalt og kryptere bluetooth beskederne sendt mellem telefonen (appen) og embedden (låsen), dog pga. tid er dette ikke blevet implementeret.

For at kunne sikre bluetooth forbindelsen kan der bruges BLE (bluetooth low energy), som hvis sat korrekt op, bruger en kommunikations protokol, som er næsten umulig at hacke.

Normal BLE sikkerheds opsætning går efter følgende parings flow:

1. udveksle enheds informationer
    a. for at sikre enhederne kan snakke med hinanden
2. korttids nøgle generering og udveksling
    a. bruges til udveksling af de fremtidige nøgler
3. langtids nøgle genereringer og udvekslinger
    a. disse nøgler vil blive brugt til al fremtidig kommunikation

Dog hvis man ikke vil parre og lagre enhederne med hinanden, hvilket er måden projekt produktet virker på, kan beskederne krypteres manuelt i stedet, hvor beskederne vil blive krypteret fra appens side, for derefter at blive krypteret på embed siden.

I tilfælde af at der vil bruges manuelle krypteringer, kan der fx bruges sha krypteringer, som er envejs krypteringer, hvilket betyder at beskederne normalt set ikke kan dekrypteres, så nøglen vil blive sha krypteret, sendt til embedden, og embedden vil så tjekke om krypteringen passer overens med den sha krypterede kode på embedden.

Der findes dog også mange andre krypterings muligheder end lige sha og standard BLE sikkerhed, dette er dog 2 måder, som ville være forholdsvis hurtige at sætte op.

Marc Blunsdon
Jan Kastbjerg

Reference

[1]	Implementerings guide af fingerskan https://proandroiddev.com/5-steps-to-implement-biometric-authentication-in-android-dbeb825aeee8 (sidst besøgt 16-11-2021)
[2]	Androids dokumentation af fingerskan https://developer.android.com/training/sign-in/biometric-auth?hl=en (sidst besøgt 16-11-2021)
[3]	Androids egen sikkerhed https://developer.android.com/training/articles/keystore (sidst besøgt 16-11-2021)