Kévin Santos

73422

David Cardoso

79710

Guilherme Quintino

80937

Project Proposal: Secure Child Locator

Group #20 - SIRS – Alameda - IST

Project proposal aims to clarify the problem and the solution to a secure child locator application in terms of the issues regarding security.

Project Proposal: Secure Child Locator

Group #20 - SIRS – Alameda - IST

Introduction

Children are amongst the most vulnerable people on the planet. Also, a great portion of them use smartphones. We constantly worry about their whereabouts, but when you try to call them, they don't pick up, either because they're in class or simply don't notice. What if we could know if something is wrong by having their location on our smartphones?

Security Requirements

Redundancy: App will not be shutdown. Multiple ways of tracking the smartphone.

Authenticity: Multiple level login layers. Local, remote and confirmation on add device.

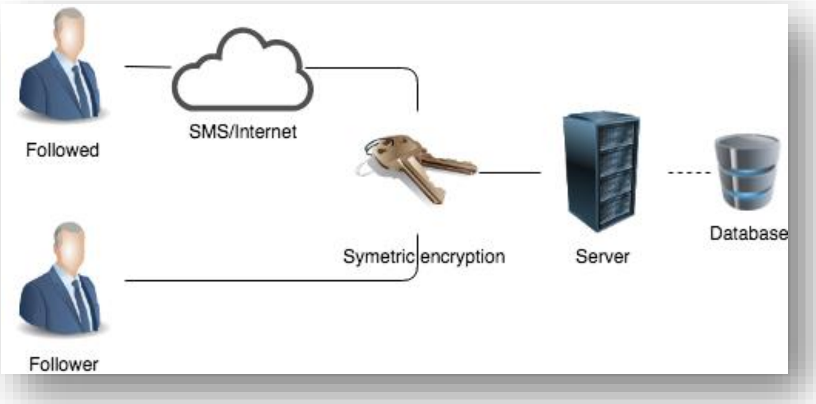
Freshness: The use of nonces on register/add device. Feedback on deprecated data is shown in the app if server/tracking not available.

Integrity: Hashing messages and encrypting them for them to be compared at destination.

Confidentiality: Messages from/to the app are encrypted

Non-Repudiation: Usage of certificates/signatures will ensure that all messages come from the server and the other way around.

Proposed Solution



Assumptions

- Every person owns a non-rooted smartphone.
- Every phone is a smartphone with GPS.
- Either Wi-Fi or 4G or mobile service is available at all times.
- Server is always up.
- Only network threats will be considered.

Features

- The app is concealed to prevent tampering or removal. Only with the correct combination of characters is the app opened.
- More than 3 failed login attempts to access the app result in a e^n minutes timeout (n=number of failed attempts after the 3rd. Resets on success).
- Watchdog to ensure the app never shuts down.

Tool References

- Java:
  - Library Java Secure Random
  - Library Java Security
  - Library Java Secure Channel
  - Library Java MySQL

There will be at least two users and a server in play. The server has a connection to the databases where the login data and the list of users who are tracking other users. Ideally there would be two databases for each data types, but due to simplicity, at the moment only one will be set. The database will be storing passwords (hashed with salt) in order to protect them from unauthorized access.

Users register trough the server, and the server will use the mobile network SMS to validate the registration. Also, SMS will be used to send tracking data in case of no internet connection.

Reauthentication will be necessary every time you update the tracking information (valid for 24h).

The app on the smartphone is protected by a local 4-8 digit pin number (locally hashed). All communication channel is encrypted with a symmetric key.

Certificates will ensure that the user is talking to the actual server and authentication from the user will guarantee the validity of the identity to the server.

The server has two purposes: registration/login and redirection of GPS coordinates.

The app will be simulated through the terminal, one for each user and another for the server. GPS location will be mocked by reading some coordinates of a text file.

Threats

1. Unauthorized requests for another user’s location.
2. The victim's cell phone can be stolen.
3. The attacker gets access to the DB.
4. Since the server is not trustworthy, it cannot have access to the contents of the messages redirected.
5. Eavesdropping.
6. Lack of input validation (SQL injection and buffer overflows).
7. Replay attacks.
8. Man-in-the-middle attack where a malicious user can pretend to be the server.

Basic Solution

For the basic solution we aim to ensure **authentication** through registration (user will be asked to provide phone number, email and the password), login (user will input his/her number and the respective password). [Threat 1]

Also implement the local pin in the app (hashed and stored locally. [Threat 2]

The server will add salt (6 digit) to the plain text password, hash it (SHA-256) and compare with the stored result in the database. [Threat 3]

Encrypted messages symmetric key is the solution to the **confidentiality** problem. [Threat 4]

Encrypted communication channels with symmetric keys. [Threat 5]

Proper input validation will be assured. [Threat 6]

Intermediate Solution

For the intermediate solution we will guarantee **freshness** through nonces (random 6 alphanumeric code) on register and adding the devices. And timestamps on other types of messages. [Threat 7]

Integrity will be solved by adding a hash(SHA-256) of the message to the message before encryption. [Threat 8]

Advanced Solution

Reauthentication on GPS location update requests will be asked (every 24h) so the session has an end. This is controlled through timestamps, on server side. Two factor **authentication** on adding another device will be added in order to reinforce security. [Threat 1 and 2]

Using signed certificates, the user knows he's talking to the real server. [Threat 8]

Work (week)/Person	**Kevin**	**David**	**Guilherme**
30th October	Interface + Pin(hash)	DB	Connections
6th November	Timestamps (last login)	Login/Register (hash+salt)	Secure channels
13th November	Last location (local stored)	Add tracking	Message timestamps/Server redirect
20th November	Nonces (register + add device)	Nonces (register + add device)	Message Hashes
27th November	2FactorAuth/Reauth	2FactorAuth/Reauth	2FactorAuth/Reauth
4th December	Certificates	Certificates	Certificates

Table 1 Basic Solution, Intermediate Solution, Advanced Solution