On the Privacy of Real-World Friend-Finder Services

Aristide Fattori†, Alessandro Reina†
Andrea Gerino‡, Sergio Mascetti†‡

† Università degli Studi di Milano
‡ EveryWare Technologies

14th International Conference on Mobile Data Management Industrial Track
Milano, Italy, June 6, 2013
Friend-Finder Services

Friend finders
Popular services that allow their users to discover people that are in the vicinity through their mobile devices

Position Information

- **Explicit position**: give *precise* information on users location
- **Implicit position**: give *only approximate* information
  (e.g., a set of users nearer than a given threshold)

Openness

- **Closed buddies**: users can see information of “friends” only
- **Open buddies**: users can see everybody’s information
Friend-Finder Services

Classification

Position Information

- implicit position
- explicit position

Openness

- closed buddies
- open buddies

PCube
OneDate
Meetic
Badoo
SKOUT

FindMyFriend
Latitude
Friend-Finder Services

Classification

- These services provide a false sense of security
- It is possible to compute users' positions

Position Information

- implicit position
- explicit position

Latitude

open buddies

closed buddies

OneDate
Meetic
Badoo
SKOUT
We analyzed a real-world dating service with more than 150M users (open buddies, implicit position) and found that it is possible for an attacker to infer the position of its users.

**Contributions**

1. Two different attacks to obtain the position of an user
2. Full automation of the two attacks
3. Describe even more threatening attacks enabled by (2)
Scenario Definition

Victim

Attacker

Colluding Buddies
**Attack 1: known distances**

**Description**

- The service returns an upper bound of the distance between the victim and the attacker.
\[ d_a = \bar{d}(v, a) \]
\[ d_1 = \bar{d}(v, c_1) \]
\[ d_2 = \bar{d}(v, c_2) \]
\[ d_a = \bar{d}(v, a) \]
Attack 2: unknown distances

Description

- The service does not return \( \overline{d} \) for every user
- However, the list of nearby people is sorted according to the distance
- Idea: we move a colluding buddy \( c_1 \) away from the attacker until it switches position in the nearby list with the victim
- Then, we know \( d(v, a) < d(c_1, a) \Rightarrow \overline{d}(v, a) = d(c_1, a) \)
- Repeat from 3 different starting position, so that we can triangulate
The attacks we just illustrated can be performed manually by a single attacker, simulating colluding buddies through false location updates.

Developing an automatic client
To make the attack automatic, we must be able to programmatically query the service from different positions.
We install the mobile app in an emulator and use it to communicate with the service server.
Attack Automation

We sniff the network traffic and dump the communication

A. Fattori
On the Privacy of Real-World Friend-Finder Services
The comm is marshalled with a custom protocol
We implement a replay attack in python
Additional attacks

The *getNearby* primitive allows to obtain *precise* distance information because such information are exchanged by the protocol, although not shown in the GUI.

**Automatic attacks enabled by the primitive**

- “*Who is there?*”
- “*Where is Alice?*”
- “*Follow Alice*”
The attacker leverages public information that must be disclosed in open-buddies friend finder services

Mitigation guidelines

- Do not allow un-authenticated queries
- Set a limit on queries-per-user
- Switch to encrypted network protocols
  (Not sufficient per-se, but makes it harder for attackers)
- Identify attack patterns
  (e.g., FTL jumps)
Conclusions

Analyzing real-world friend-finder services

- Analyzed a real-world dating service with > 150M users
- Found two attacks to find the precise position of its users
- Automated the attacks to show their dangerousness
On the Privacy of Real-World Friend-Finder Services

Thank you!
Any questions?

Aristide Fattori
joystick@security.di.unimi.it