There's Wally! Location Tracking in Android without Permissions

Efthimios Alepis and Constantinos Patsakis
Department of Informatics, University of Piraeus, 80, Karaoli & Dimitriou str.,18534, Piraeus, Greece
{talepis, kpatsak}@unipi.gr

Keywords: Privacy, Location, Android, Wi-fi P2P.

Abstract: Context-awareness can be considered as one of the biggest advantage of smart mobile devices as it provides advanced features for developers revolutionizing user interaction and making users more engaged to the applications. Perhaps, the most important factor is location awareness as applications can refine their results according to users' whereabouts. Nonetheless, users' location is a very sensitive attribute as it can disclose a lot of personal information about them. To address such issues, mobile operating systems require users to grant specific permissions to the applications. This work studies a relatively new feature of Android, namely Wi-Fi P2P, illustrating that the location of the user can be easily disclosed, without using location permissions even in the recent version of Android.

1 INTRODUCTION

Mobile devices, especially smartphones, have become an inseparable part of our daily lives. While their processing power cannot be compared with desktop computers, due to their portability and the fact that they have many embedded sensors, they are able to offer advanced user interaction capabilities. In fact, the embedded sensors allow them to be context-aware, adjusting user interface, as well as the presented information accordingly.

To this end, a major advantage of these devices is that they can become aware of the user's location. Certainly, this feature is very important for the user, as it can significantly improve the results of any recommender system, or enable the development of more advanced applications such as mobile social networks. Nonetheless, the location of a user is considered as a very sensitive piece of information as it can be used to deduce a lot of other information, like social connections, political and religious beliefs, medical condition etc. Due to its sensitivity, modern mobile operating systems do not allow applications to monitor users' location without the explicit consent from the user. Therefore, whether an application is running on iOS or Android, the underlying OS will notify the user that this application wants to access the user's location and the user will decide whether this permission will be granted or not. After the recent version of Android, Marshmallow, users of both OSes are able to revoke these permissions whenever they want.

In many cases, applications do not actually need to access the user's location, however, since they operate under the "freemium" model and they want to increase their incomes from ad networks, they result into requesting this permission, among others, from their users. Undoubtedly, requesting from a user the permission to access the GPS creates many issues, as many users are not willing to share so sensitive information and decide not to install such application. Regardless of the privacy issues, using GPS to track users, results in significant battery consumption. To overcome these issues, many applications resolve to other means to locate their users.

While Marshmallow provides many new features in Android, it can be considered a security upgrade, as it contains major changes in the security architecture of the system, with the most outstanding one being the complete redefinition of its permission model. Undoubtedly, a lot of effort has been put in location permissions, making applications that used other means to determine user's whereabouts to request the user's explicit consent. Based on the above, we showcase that despite this effort, applications can still determine users' location efficiently without requiring any such permission. Actually, our proof of concept application does not require any dangerous application from the user, it can even return the results when it is running in the background, so the user is unaware of any of its actions. To achieve this, we exploit the use of WiFi-P2P, a new feature of Android which extends the former WiFi Direct. This exploit, enables us to harvest users' device IDs, such as device names

Alepis, E. and Patsakis, C.
There's Wally! Location Tracking in Android without Permissions.
DOI: 10.5220/0006125502780284
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 278-284
ISBN: 978-989-758-209-7

and corellate them against known spots or with each other. The issue has been reported to Google which acknowledged it and declared to fix in future releases, as it considers it a privacy and not security issue[1].

In what follows, we provide a brief overview of the related work in Section 2. Then, in Section 3 we provide an overview of the new Android permission model. Section 4 details our attack, its effectiveness and its accuracy. Finally, the work concludes with some remarks and possible solutions to the problem.

2 RELATED WORK

Android is by far the most widely used platform in mobile smart devices. In general, Google has a very open platform, compared to its peers, allowing developers to have access to its internals and share their applications easier and cheaper. Moreover, due to its licensing model and wide support for manufacturers, Android is currently used by more than one billion users. All the above, have attracted millions of developers who host their applications on Google Play or other markets. This humongous user base, whose users are most prone to install applications under the "freemium" model, trade functionality with access to their data. Consequently, many companies are striving to monetize the wide base of user's data. To this end, simple applications result into requesting absurd permissions with the sole purpose to harvest user data (SnoopWall, 2014). This constant increase for requests for more and more permissions has made users to relax their standards, ignore notifications and disregard the risks they are exposed to (Felt et al., 2012; Kelley et al., 2012; Balebako et al., 2013). Clearly, of specific interest are the ad libraries which have the actual role of harvesting users' information. Unfortunately, their role in many instances is not benign, as some of them have been proven to use undocumented permissions (Stevens et al., 2012) or probing applications to determine whether they have more privileges and abuse them to derive sensitive user information (Grace et al., 2012; Book et al., 2013) as permissions in Android are granted per application and not per component.

To counter privacy exposure, several solutions have been proposed in the literature (Zhou et al., 2011; Kim et al., 2012; Gibler et al., 2012; Pandita et al., 2013; Enck et al., 2014) where they try to monitor what information is used by applications and whether they try to transmit it. Location is a very sensitive piece of information, highly valued by the users (Danezis et al.,), which can be used by advertisers very easily and improves very much the recommendations in case of targeted advertisements. Therefore, many applications and ad frameworks try to extract it (Fu et al., 2014; Almuhimedi et al., 2015), as there are many means to infer individuals' location, such as Wi-Fi signal (Krumm and Horvitz, 2004; Wind et al., 2016; Sapiezynski et al., 2015; Vanhoef et al., 2016), accelometers (Han et al., 2012), power consumption (Michalevsky et al., 2015) or even ambient sound, light, and color (Azizyan et al., 2009). Several attacks have already been made for location based services, such as (Kune et al., ; Qin et al., 2014; Wernke et al., 2014; Polakis et al., 2015; Farnden et al., ; Shaik, 2016), therefore, several measures have been proposed (Narayanan et al., 2011; Guha et al., 2012; Fawaz and Shin, 2014; Theodorakopoulos et al., 2014; Kotzanikolaou et al., 2016; Patsakis et al., 2015; Polakis et al., 2015). For a more detailed study on the privacy issues in mobile devices, the interested users may refer to (Spensky et al., 2016).

However, to determine one's location, one could also use other means, as many users switch off location services in their devices, not only for privacy, but for battery consumption as well. While Wi-Fi positioning system (WPS) is a very well-known method to achieve location services indoors where GPS signal cannot be reached but the same concept can be used counter-wise. Using the MAC addresses of hotspots, one could determine the position of an individual, knowing of course where these hotspots are located. Therefore, many applications scan to find the available Wi-Fis to collect their MAC addresses and transmit them. Since for some of them the service provider already knows where they are located, it is fairly easy for him to approximate user's location, without requesting access to GPS.

Another twist of the previous method is to collect the MAC addresses of devices that want to connect to a Wi-Fi. Generally, most people leave the Wi-Fi of their devices open, for convenience, regardless of whether they are in proximity to a known network. Since the device does not know whether there is a known network close, it will periodically send a beacon for its preferred Wi-Fis. Up to recently, this beacon contained the original MAC address of the device, enabling an adversary to collect it and correlate it with other captures. Consequently, installing many capture devices around a city, one can easily monitor the location of thousands of people as in the case of London's smart thrash cans[2]. To counter this issue some OSes randomize the MAC addresses when they send these

[1]Bug report 216235 https://code.google.com/p/android/ issues/list

[2]http://www.theverge.com/2013/8/9/4604980/smart-uk-trash-cans-smartphone-speed-proximity-Wi-Fi

beacons, however, more advanced tools like mana[3] or attacks such as the one of Vanhoef et al. (Vanhoef et al., 2016) can bypass this security measure.

Apart from the Wi-Fi beacons, Bluetooth can also be used to track users' location. Similarly to Wi-Fi, one could perform a scan for available Bluetooth devices and use the aforementioned techniques to determine the user's location. More recently, exploiting Bluetooth Low Energy (BLE) wireless transmitters, beacons can be used to track a mobile device's location and trigger notifications once a device is in proximity or leaves a location. The difference in the case of BLE to traditional Bluetooth is that the power consumption is far less, but can transmit shorter messages. Thus, similar to Wi-Fi beacons, Bluetooth ones can be used to identify devices in proximity which is very practical for devices that have low power requirements, such as proximity sensors, heart rate monitors, and fitness devices. Nonetheless, due to the nature of Bluetooth, devices have to be real close to be identified.

3 PERMISSIONS IN MARSHMALLOW

Despite the numerous changes in the latest version of Android, codenamed Marshmallow, this version can be characterised as a major security upgrade. Google abandoned the permission model it had adopted since 2007, in which application permissions where granted by users on application installation. This model inferred a one time offer where the user would either accept to grant the permissions and install the application, or he would not and thus would not be allowed to install the application. Following the example of iOS and custom ROMs, Google decided to completely redesign its permission model, enabling "Runtime Permissions" so that users could install the desired applications and then, during runtime, decide whether access to specific resources would be granted. However, the most important aspect of the new model is that users may revoke permissions to specific applications whenever the want to, enabling them to have fine-grained permission policies.

According to the new model the permissions are categorized according to the risk they expose the user into four categories:

- Normal: These permissions imply the least possible user exposure, so Android automatically grants them upon installation. Not only these permissions are automatically granted, but additionally they cannot be revoked and are not transparent to the user.
- Dangerous: These permissions imply, as the name suggests, serious risk for the user's privacy and security. Therefore, to be granted, Android requests explicit user approval, which can be revoked at any time by the user. Figure 1 illustrates the complete list of dangerous permissions.
- To guarantee that specific applications are granted some permissions, Google has introduced the *signature* permission. This permission grants access to an application only if the requesting application is signed with the same certificate as the application that declared the permission without user notification. This permission facilitates interoperability between applications.
- Finally, to provide advanced privileges for manufacturers, Google introduced the *system* permission which is granted only to applications that reside in the Android system image or signed with the same certificate as the application that declared the permission. Such privileges provide applications with low level control to the device as they are able to reboot, clear the caches or even monitor other processes.

This new model decreases the notifications that a user has to respond to as the amount of interactions involve only the seven categories of dangerous permissions..

4 RETRIEVING USER'S LOCATION

In order to retrieve the user's location we use Wi-Fi P2P[4], the new feature of Android which extends Wi-Fi Direct[5]. Wi-Fi P2P allows two devices to connect directly without the use of another entity achieving greater distances and speeds than Bluetooth. To enable programmatic access to this feature, a developer has only to declare the following permissions `ACCESS_Wi-Fi_STATE` `CHANGE_Wi-Fi_STATE` and `INTERNET` in `AndroidManifest.xml`. Clearly, all of them are considered "normal" permissions from Google, therefore, users not only cannot revoke them, but they cannot even be notified about them. Figures 2a and 2b illustrate the screenshots from our proof of concept application, showing that the afforementioned permissions are not reported to the user. It is important to note that the underlying architecture

[3]https://github.com/sensepost/mana

[4]https://developer.android.com/guide/topics/connectivity/Wi-Fip2p.html

[5]We need to add a link/ref if possible

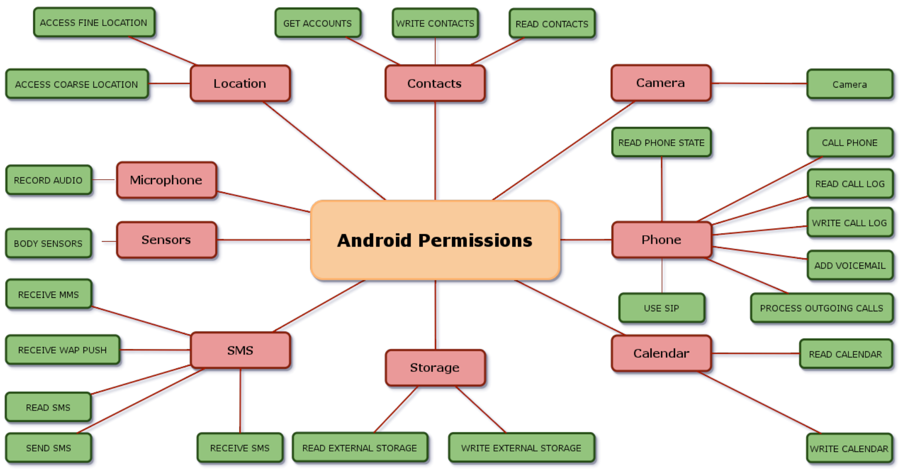


Figure 1: Dangerous permissions Marshmallow.

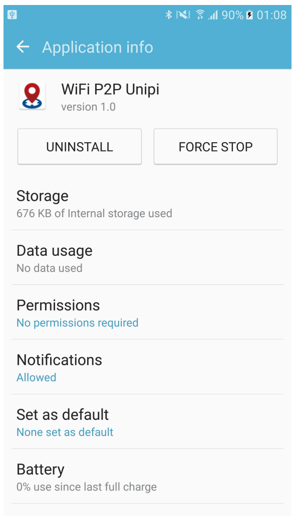
of this specific attack allows a device to simply scan for the available devices without the need to actually connect with them to recover the needed information. Therefore, after a scan, an adversary can easily periodically scan for available devices, see Figure 2c, and harvest their MAC addresses and device names, see Figure 2d, which can uniquely identify them.

Based on the above, tracking users' location without their consent or knowledge is quite straightforward. To showcase the attack scenario, we consider the use case of an ad network, which as already discussed, often tries to collect such information. Therefore, an ad network provides a Wi-Fi P2P library within its package and periodically scans the network for available P2P devices. Moreover, the ad network has installed or cooperated with other entities which provide it with P2P nodes, static or mobile, whose location is always disclosed, and they can be installed in malls, hotels, restaurants, or carried in concerts etc. with a minimal cost. Knowing the location of these end points, the ad network can easily tell where individuals are located, with an accuracy of 100m. Note that at the time of writting, this accuracy cannot be improved by e.g. using the signal strength, as it is hardcoded to be always 60dB. For more fine-grained results, more sophisticated methods could be used such as (Curtis et al., 2014).

Some possible attack scenarios are illustrated in Figure 3. In the firstcase 3a, we have an adversary, whose location is known and can be fixed or mobile. The adversary scans the network using Wi-Fi P2P and finds the MAC addresses of the users nearby, sending them to a remote server. In the second case, we might not have the actual location of any of the users, nonetheless, professional and social connections of users can be inferred using time as an additional parameter, e.g. how long they stay close, what times of day etc. thus resulting in harvesting users' relative location.

It is worth noticing that Google introduced some new rules in the permissions to counter such actions. For instance, in Marshmallow, to perform Wi-Fi or Bluetooth scans, Google mandates developers to request the permission for location, more precisely `ACCESS_COARSE_LOCATION`. Clearly, exploiting the Wi-Fi P2P feature overcomes the restriction of requesting location permission to the user, which cannot be later revoked, but this is not the only case as it is going to be discussed later on. More interestingly, while Marshmallow decided to follow the example of iOS and randomize the MAC addresses during Wi-Fi Scans, this does not happen in the case of Wi-Fi P2P. Actually, the devices disclose most of their actual MAC addresses each time, as it only changes the first two bytes of the address. Clearly, this method does not hide the user's MAC address, in many cases, one can use the bytes from positions 3 to 6 to recover the obfuscated ones, as they characterise a vendor so they are publicly known. Another method is to use as MAC address all bytes apart from the first two, as the chances of collisions are very low, or improve this fingerprint using the device name, something that most users hardly change. Moreover, the devices disclose their name which in many occasions can be considered a unique identifier.

Two permissions are required by a mobile device to perform any Bluetooth communication, such as initiating device discovery, requesting Bluetooth connections, accepting Bluetooth connections, and transferring data, namely `android.permission.BLUETOOTH` and `android.permission.BLUETOOTH_ADMIN` which are both considered normal. However, in order to perform a BLE scan, like in Wi-Fi scan, one needs



Application info

WiFi P2P Unipi
version 1.0

UNINSTALL FORCE STOP

Storage
676 KB of Internal storage used

Data usage
No data used

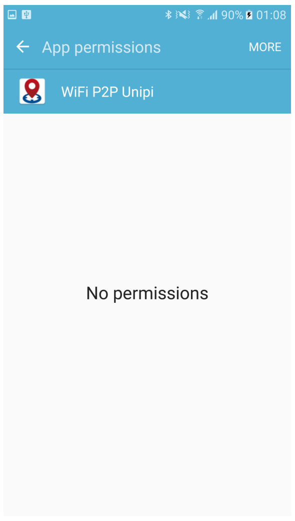
Permissions
No permissions required

Notifications
Allowed

Set as default
None set as default

Battery
0% use since last full charge

(a) Permissions for Wi-Fi P2P proof of concept application.

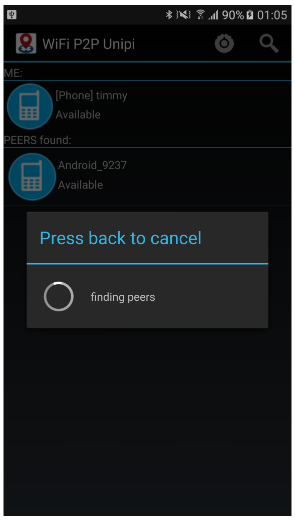


App permissions MORE

WiFi P2P Unipi

No permissions

(b) Permissions for Wi-Fi P2P proof of concept application, detailed.



WiFi P2P Unipi

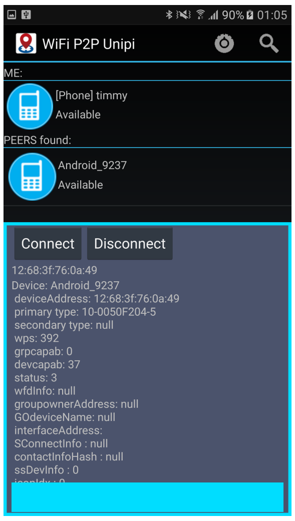
ME:
[Phone] timmy
Available

PEERS found:
Android_9237
Available

Press back to cancel

finding peers

(c) Scan for available Wi-Fi P2P devices.



WiFi P2P Unipi

ME:
[Phone] timmy
Available

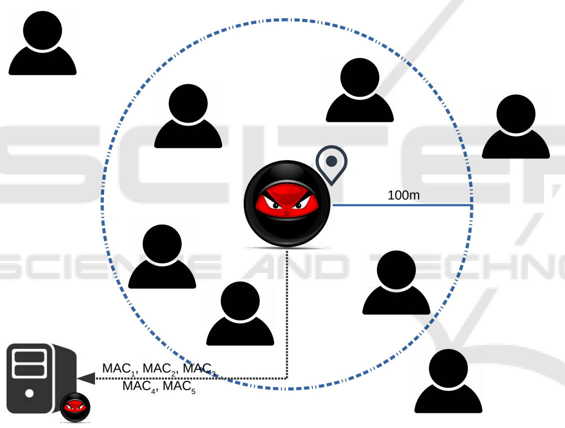
PEERS found:
Android_9237
Available

Connect Disconnect

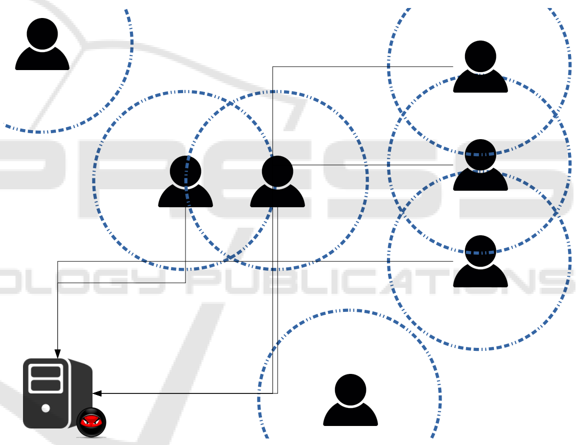
```
12:68:3f:76:0a:49
Device: Android_9237
 deviceAddress: 12:68:3f:76:0a:49
 primary type: 10-0050F204-5
 secondary type: null
 wps: 392
 grpcapab: 0
 devcapab: 37
 status: 3
 wfdInfo: null
 groupownerAddress: null
 GOdeviceName: null
 interfaceAddress:
 SConnectInfo : null
 contactInfoHash : null
 ssDevInfo : 0
```

(d) Details of available Wi-Fi P2P devices.

Figure 2: Wi-Fi P2P proof of concept application.



(a) Attack scenario with an attacker whose location is known.



(b) Attack scenario without knowing locations.

Figure 3: The attack model.

to request the ACCESS_COARSE_LOCATION permission in Marshmallow. However, this restriction has been repeadely reported to be lifted by simply switching from `startScan` method of the current API to the deprecated `startLeScan` method of API 18. Finally, we highlight that the results of the aforementioned scans can be transmitted and stored to a remote server, again without requesting any permissions from the user, since access to the Internet is considered a normal permission.

All the above indicate that the problem of location is very complex as the introduction of new features perplexes the problem since manufacturers cannot always foresee how independent features can be exploited for other purposes. Moreover, the huge fragmentation of Android into many versions which cannot actually be upgraded to newer ones, traps the users and prevents them from properly securing their devices. More importantly, however, the new permission model, does not provide the necessary transparency and control to the users over their applications, allowing developers to exploit it and hide potentially malicious actions in applications which may not require any permissions from the user at all.

It must be highlighted that the security issue we are disclosing does not affect mobile devices that have a “special” setting turned on, therefore it affects potentially every Android device. Once our proof of concept application is executed on a mobile device and its Wi-Fi P2P discovery process is initiated, every available node gets exposed by revealing its identity to other phones running our software. In order to support our claims we have experimented with mobile devices running in Android version 4.4.2, to determine whether older smartphones are susceptible, as well as with newer mobile devices as of version 6.0.1. The experiments also included identification of cross-version devices, meaning that a victim device was running on Android version 4.4.2, while the attacker was running on Android version 6.0.1 and vice versa. According to Google, Android devices that run on Android KitKat (more specifically API level 19) and later represent approximately 73,9% of all Android devices that are active on Google Play Store (December 2016). Apparently, this means that the vast majority of Android devices, more than 87% of the total devices if we add the devices running Android 6, are exposed to this vulnerability. Furthermore, Google, has already acknowledged this issue and is expected to be solved in a future Android release.

5 CONCLUSIONS

This work highlights issues related to the location privacy in modern mobile devices. More specifically, by utilizing Wi-Fi P2P, a novel standard which enables devices to easily connect with each other without requiring any additional wireless access point, we show that we can easily determine passively the location of mobile devices, within a radius of at most 100 meters. It is quite important to notice that the resulting proof of concept mobile application is actually a zero permission application installed in an Android OS device running on the latest version, namely Marshmallow, API level 23.

A possible solution to the aforementioned problem would be to incorporate the same security permission model that is adopted in Android for API level 23 and above concerning traditional Wi-Fi and Bluetooth connections. More precisely, to provide users with greater data protection, starting with Android Marshmallow, the OS removes programmatic access to the device's local hardware identifier for apps using the Wi-Fi and Bluetooth APIs. Correspondingly, `Wi-FiInfo.getMacAddress()` and the `BluetoothAdapter.getAddress()` methods are returning a constant value of "02:00:00 :00:00:00", when called. To access the hardware identifiers of nearby external devices via Bluetooth and Wi-Fi scans, smartphone applications must be granted the `ACCESS_FINE_LOCATION` or `ACCESS_COARSE_LOCATION` permissions. We believe that this approach could be also applied in the case of Wi-Fi P2P to ensure users' privacy. Nonetheless, our work signifies that similar features, Wi-Fi access and Wi-Fi P2P, can have different rules and not generic ones due to the maturity of some solutions and that new features are not properly communicated to all the stakeholders, leaving security and privacy issues which can be easily exploited.

ACKNOWLEDGEMENTS

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the *OPERANDO* project (Grant Agreements no. 653704).

REFERENCES

Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Gluck, J., Cranor, L. F., and Agarwal, Y. (2015). Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 787–796. ACM.

Azizyan, M., Constandache, I., and Roy Choudhury, R. (2009). Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 261–272. ACM.

Balebako, R., Jung, J., Lu, W., Cranor, L. F., and Nguyen, C. (2013). Little brothers watching you: Raising awareness of data leaks on smartphones. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, page 12. ACM.

Book, T., Pridgen, A., and Wallach, D. S. (2013). Longitudinal analysis of android ad library permissions. *arXiv preprint arXiv:1303.0857*.

Curtis, P., Banavar, M. K., Zhang, S., Spanias, A., and Weber, V. (2014). Android acoustic ranging. In Bourbakis, N. G., Tsihrintzis, G. A., and Virvou, M., editors, *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications, Chania, Crete, Greece, July 7-9, 2014*, pages 118–123. IEEE.

Danezis, G., Lewis, S., and Anderson, R. J. How much is location privacy worth? Citeseer.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. (2014). Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5.

Farnden, J., Martini, B., and Choo, K.-K. R. Privacy risks in mobile dating apps. In *Proceedings of 21st Americas Conference on Information Systems (AMCIS 2015)*, volume 13, page 15.

Fawaz, K. and Shin, K. G. (2014). Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250. ACM.

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM.

Fu, H., Yang, Y., Shingte, N., Lindqvist, J., and Gruteser, M. (2014). A field study of run-time location access disclosures on android smartphones. *Proc. USEC*, 14.

Gibler, C., Crussell, J., Erickson, J., and Chen, H. (2012). Androidleaks: automatically detecting potential privacy leaks in android applications on a large scale. In *International Conference on Trust and Trustworthy Computing*, pages 291–307. Springer.

Grace, M. C., Zhou, W., Jiang, X., and Sadeghi, A.-R. (2012). Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112. ACM.

Guha, S., Jain, M., and Padmanabhan, V. N. (2012). Koi: A location-privacy platform for smartphone apps. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 14–14. USENIX Association.

Han, J., Owusu, E., Nguyen, L. T., Perrig, A., and Zhang, J. (2012). Accomplice: Location inference using accelerometers on smartphones. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pages 1–9. IEEE.

Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., and Wetherall, D. (2012). A conundrum of permissions: installing applications on an android smartphone. In *Financial Cryptography and Data Security*, pages 68–79. Springer.

Kim, J., Yoon, Y., Yi, K., Shin, J., and Center, S. (2012). Scandal: Static analyzer for detecting privacy leaks in android applications.

Kotzanikolaou, P., Patsakis, C., Magkos, E., and Korakakis, M. (2016). Lightweight private proximity testing for geospatial social networks. *Computer Communications*, 73:263–270.

Krumm, J. and Horvitz, E. (2004). Locadio: inferring motion and location from wi-fi signal strengths. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 4–13. IEEE.

Kune, D. F., Koelndorfer, J., Hopper, N., and Kim, Y. Location leaks on the gsm air interface.

Michalevsky, Y., Schulman, A., Veerapandian, G. A., Boneh, D., and Nakibly, G. (2015). Powerspy: Location tracking using mobile device power analysis. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 785–800.

Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., and Boneh, D. (2011). Location privacy via private proximity testing. In *NDSS*.

Pandita, R., Xiao, X., Yang, W., Enck, W., and Xie, T. (2013). Whyper: Towards automating risk assessment of mobile applications. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 527–542.

Patsakis, C., Kotzanikolaou, P., and Bouroche, M. (2015). Private proximity testing on steroids: An ntru-based protocol. In *International Workshop on Security and Trust Management*, pages 172–184. Springer.

Polakis, I., Argyros, G., Petsios, T., Sivakorn, S., and Keromytis, A. D. (2015). Where's wally?: Precise user discovery attacks in location proximity services. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 817–828. ACM.

Qin, G., Patsakis, C., and Bouroche, M. (2014). Playing hide and seek with mobile dating applications. In *IFIP International Information Security Conference*, pages 185–196. Springer.

Sapiezynski, P., Stopczynski, A., Gatej, R., and Lehmann, S. (2015). Tracking human mobility using wifi signals. *PloS one*, 10(7):e0130824.

Shaik, A. (2016). Practical attacks against privacy and availability in 4g/lte mobile communication systems.

SnoopWall (2014). Flashlight apps threat assessment report. http://www.snoopwall.com/wp-content/uploads/2015/02/Flashlight-Spyware-Report-2014.pdf.

Spensky, C., Stewart, J., Yerukhimovich, A., Shay, R., Trachtenberg, A., Housley, R., and Cunningham, R. K. (2016). Sok: Privacy on mobile devices–it's complicated. *Proceedings on Privacy Enhancing Technologies*, 2016(3):96–116.

Stevens, R., Gibler, C., Crussell, J., Erickson, J., and Chen, H. (2012). Investigating user privacy in android ad libraries. In *Proceedings of the 2012 Workshop on Mobile Security Technologies (MoST)*.

Theodorakopoulos, G., Shokri, R., Troncoso, C., Hubaux, J.-P., and Le Boudec, J.-Y. (2014). Prolonging the hide-and-seek game: Optimal trajectory privacy for location-based services. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 73–82. ACM.

Vanhoef, M., Matte, C., Cunche, M., Cardoso, L. S., and Piessens, F. (2016). Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424. ACM.

Wernke, M., Skvortsov, P., Dürr, F., and Rothermel, K. (2014). A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, 18(1):163–175.

Wind, D. K., Sapiezynski, P., Furman, M. A., and Lehmann, S. (2016). Inferring stop-locations from wifi. *PloS one*, 11(2):e0149105.

Zhou, Y., Zhang, X., Jiang, X., and Freeh, V. W. (2011). Taming information-stealing smartphone applications (on android). In *International conference on Trust and trustworthy computing*, pages 93–107. Springer.