Analysis of Privacy and Security Exposure in Mobile Dating Applications

Constantinos Patsakis[1], Athanasios Zigomitros[1] and Agusti Solanas[2]

[1] Department of Informatics, University of Piraeus, Greece
[2] Smart Health Research Group, Dept. Computer Engineering and Mathematics, Universitat Rovira i Virgili, Catalonia. Spain

Abstract. Millions of people around the globe try to find their other half using Information and Communication Technologies. Although this goal could be partially sought in social networks, specialized applications have been developed for this very purpose. Dating applications and more precisely *mobile* dating applications are experiencing a continuous growth in the number of registered users worldwide. Thanks to the GPS and other sensors embedded in off-the-shelves mobile devices, dating mobile apps can provide location aware content, not only about the surroundings, but also about nearby users. Even if these applications have millions of registered users, it can hardly be said that they are using the best standards of security and privacy protection.
In this work we study some of the major dating applications and we report some of the risks to which their users are exposed to. Our findings indicate that a malicious user could easily obtain significant amounts of fine-grained personal information about users

Keywords: User profiling, Location privacy, Online social networks, Security and privacy exposure

1 Introduction

The wide adoption of ICT has drastically modified the way we exchange information and interact with other people. Nowadays, our capacity to communicate with others is no longer bounded to our immediate surroundings. Instead, we can easily interact with people from distant places in almost real time.

Mobile technology has become commonplace and the possibility to easily reach a huge market has fostered the development of numerous and diverse applications. In addition, smartphones and tablets are steadily improving their computing capabilities and come equipped with accelerometers, GPS and a series of other sensors that enable developers to deploy more sophisticated solutions to exploit spatial information. As a result of these new means of interaction, users are more engaged and developers can make them actively participate and provide added-value content and information.

A good example of these communication and behavioural changes can be found in mobile dating applications. The way in which people look for partners

has drastically changed as a result of the use of mobile technology. However, using technology does not come without risks. Due to their very nature, dating applications contain sensitive information. Note that in addition to the sexual orientation and preferences of users, modern dating apps are context-aware and allow the finding of nearby partners. Moreover, user profiles contain other sensitive data such as political views or beliefs.

Certainly, users would like to share some personal information with potential partners as these could create criteria to filter candidates. Notwithstanding, users might not want this shared information to be publicly available. Additionally, the information exchanged between two users might be private and very sensitive. Thus, information leaks could have a huge impact on the reputation of individuals and should be avoided.

It might be thought that these privacy and security issues are well-known and defined. Hence, one would expect that developers put in place the right measures to secure their applications and deter any information leak, especially since these applications are used by millions of people.

In this article we study 18 mobile dating and chatting apps and highlight doubtful software development practices, which we have found to be commonplace. The detected vulnerabilities are, in many cases, quite obvious so are their solutions. The vulnerabilities that we discuss might have a very big impact on the users in a variety of ways and might affect millions. More importantly, we have observed that those vulnerabilities can be exploited very easily and require little, if any, computer skills. Actually, there is no need for reverse engineering of the applications. In our experiments, the most *sophisticated* scenario implies to intercept network packages *e.g.* by using a proxy, while in the simplest scenario, the attacker just needs to eavesdrop exchanged messages.

2 Related work on S&P in Apps

The use of ICT has helped us to communicate and exchange information more easily, specially thanks to the Internet. However, it has opened the door to remote attacks that might affect millions. With the aim to fight against these attacks many organizations strive to raise awareness about secure web development. For instance, OWASP[3]compiles the well-known OWASP Top Ten survey, which highlights the most critical web application security flaws. This survey provides a very good insight on what developers should be aware of when developing applications and it illustrates how attackers would try to penetrate into a web service. Other surveys such as [1] provide similar results.

Mobile applications could be developed as stand-alone services running in a mobile device. However, in most cases, they do not rely on local resources only but use web content and infrastructure. Very frequently, most of the mobile apps content is retrieved, uploaded, and updated through the Internet, and the data gathered by means of the embedded sensors are used to provide value-added services and functionalities.

[3] `https://www.owasp.org`

Currently, the most used operating systems for smart phones are Android and iOS. Android provides a permission-based security model, however, it has been shown that it has vulnerabilities [2, 3]. Beresford et al. [4] created *MockDroid*, a modified version of Android which allows users to intervene and revoke the access rights of applications to particular resources at run-time. Definitely, this approach has a very big impact on the usability of the application because it limits some functions. Notwithstanding, it allows users to define their own privacy policies and reduce their possible information leakage. A similar approach was followed with *TISSA* [5], which allows users to fine-tune their privacy policies and the access level of specific applications at run-time. Enck et al. approached the problem from a different perspective and introduced *TaintDroid* [6]. Instead of constantly intervening, their Android mod tracks down which sensitive data is requested by an app and when it is used, hence, providing real-time analytics. Their analysis can be very useful in the automated characterization of benign, grayware and malware [7]. A more user friendly approach has currently been embedded in *CyanogenMod*[4], one of the most well-known customized distributions of Android, allowing users to block access to apps from specific resources.

In the iOS arena, where publishing an app in the official app store passes some stricter filters, Egele et al. [8] made a comprehensive study over a sample of 1,400 iOS apps (3/5 from the iTunes App Store and 2/5 from the Cydia repository). While most apps were found to be benign, their analysis revealed that more than half of them was leaking the unique ID of the device. To provide users with a clearer overview about how their information flows in their mobile devices, Wetherall et al. [9] created two tools: a browser plugin and a mobile app. These tools alert users of dangerous logins (a.k.a. logins performed without encryption) and inform them about which private information is collected.

Burattin et al. [10] illustrated that while users think that some of their information is hidden, e.g. list of friends, it can be recovered from the OSNs with various methods. Focusing on dating apps, Qin et al. [11] show how an attacker could obtain the real location of users in several well-known dating apps. It was showed that the main reason for this exposure is the poor randomization and obfuscation techniques used.

Recently, the Electronic Frontier Foundation published a review[5] to determine what security is offered by secure messaging products. The review clearly indicates that there are many problems in the architecture of most applications and the user cannot be considered secure as for instance she cannot verify the contact's identity, or the service provider has access to the users' traffic.

3 Experimental Setup

In our experiments we have approached the problem of obtaining information from a non-invasive perspective. Given the legal constrains to apply reverse engineering on an application, and considering the capacities of an average user

[4] `www.cyanogenmod.org`
[5] `https://www.eff.org/secure-messaging-scorecard`

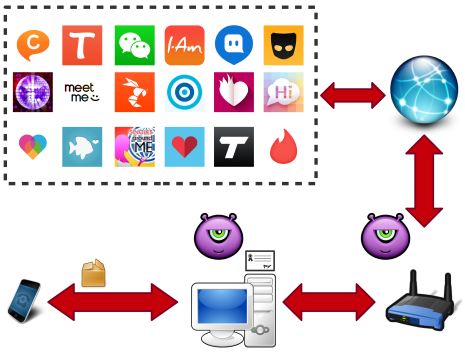


Fig. 1. Experimental setup

or a network administrator, we have installed a proxy that intercepted the messages targeted towards our apps. To ensure that the content of the intercepted packages can be analysed by the proxy, we generated a root certificate for it and installed it in the smartphone. Therefore, even if the packages were encrypted, their content could be read by the proxy. In this sense, the setup is analogous to a *man-in-the-middle attack*, with the only, but important, difference that we are controlling all sides, both the benign and the malicious.

The above setup mimics two real-world attack scenarios: In the first scenario, a network administrator might try to gather as much information as possible about the users in his network. Note that, as it is shown in our findings, there is no need for the network administrator to request his users to install a certificate in order to attack them, as many applications are using unencrypted traffic or leave important information in the header so the administrator only needs to watch regular network traffic. In principle, beyond the malicious network administrator, anyone who can perform traffic sniffing can execute these attacks.

The second scenario involves a malicious user with cyber-stalking intentions. The attacker wants to find probable victims and their whereabouts. To this end, he/she intercepts the packages that are sent and received from the applications and he uses them to extract further fine-grained information. Figure 1 illustrates our setup and where we expect our adversary to be logically located.

To capture the data from the studied apps we have used Fiddler 2 [6] as the debugging proxy and we have installed its certificate in a smartphone. All the experiments were conducted using an iPhone 4 with iOS 7.0.4. However, as it is going to be discussed, the operating system does not have any relevance to the vulnerabilities that we have found. The vulnerabilities that we discuss in this work are mainly due to the exchanged traffic between the apps and web servers that host the services and, in fact, the apps *per se* have not been analysed.

[6] `http://www.telerik.com/fiddler`

4 The findings

We have analysed 18 mobile dating and chatting applications to study whether they send sensitive HTTP traffic, include the current location of users, send the actual distance to other users, use static links, etc. An overview of our findings is depicted in Figure 2 and in what follows we briefly describe the specific details/vulnerabilities of each studied application.

ChatOn: ChatOn uses HTTPS for all its traffic. However, many details could be leaked through the URL of the API. For instance, an eavesdropper can easily find many information about the user's phone, namely, model, operating system version, IMEI, IMSI, telephone number, user ID and app version. The app sends the telephone numbers of all user's contacts to Samsung, and the received packets contain additional information like contacts' birthday. The RESTful API that is used exposes users actions and the profiles that they visit.

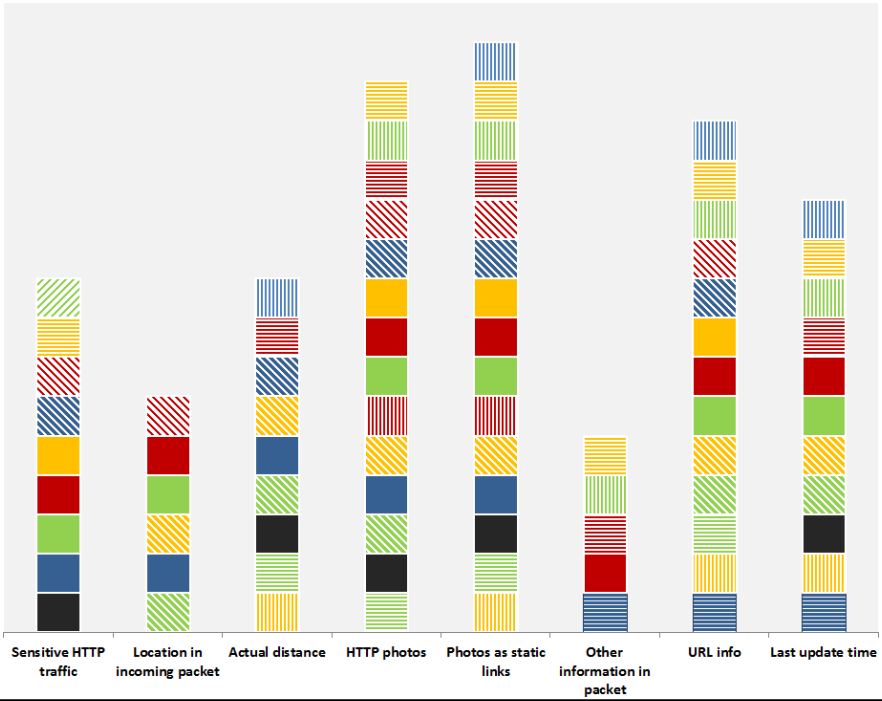
Grindr: Grindr uses HTTPS for most of its communications, but photographs are sent by using static links over HTTP. The API that is called from the mobile app might allow eavesdroppers to extract the actual user location and his/her application ID from the sniffed URL. Additionally, the URL discloses the user's activity and his/her device OS. Moreover, exchanged packets contain the distance only for users that consented and the application might display the relative user distance. However, the messages contain the actual users' location.

Hornet: Hornet encrypts its traffic using HTTPS, but sends the distance with 10m accuracy. Photos are static links sent over HTTP. The API calls allow an adversary to deduce user activity, e.g. chatting, browsing profiles, etc simply by capturing the URLs that users request.

I-Am: Apart from the authentication which is executed over HTTPS, the rest of the traffic of I-Am goes over HTTP, hence, allowing an adversary to have full access to user private data. Images are sent over HTTP and as static links. Regarding spatial data, the actual user location is sent in the URL without any encryption or obfuscation, and the exact distance to other users is sent in the packet along with their birthday.

LOVOO: All traffic of LOVOO is sent over HTTPS, with the exception of photos, which are sent over HTTP. It is interesting to notice that links are dynamic and expire. The actual distance between users is sent with a rounding of 100m, along with their relative (x, y) coordinates. Thus, an adversary could recover the actual locations. Also, the API calls expose in the URLs that are requested the user location, his/her preferences and his/her overall activity.

MeetMe: MeetMe uses mixed HTTP/HTTPS traffic. The user location and his/her preferences are visible in the URL. The actual location of the user is included in the packet if other users are nearby, otherwise their distance is given in Km. Photos are shared over HTTP, and user commands can be seen in the URL.



Application	Version	Installations	Code	Application	Version	Installations	Code
ChatOn	3.0.2	100m-500m		Singles around me	3.3.1	500K-1m	
Grindr	2.0.24	5m-10m		SKOUT	4.4.2	10m-50m	
Hornet	2.0.14	1m-5m		Tagged	7.3.0	10m-50m	
I-Am	3.2	500K-1m		Tango	5.8	100m-500m	
LOVOO	2.5.6	10m-50m		Tinder	4.0.9	10m-50m	
MeetMe	9.2.0	10m-50m		Tingle	1.12	-	
MoMo	5.4	1m-5m		Waplog	3.0.3	5m-10m	
POF	2.40	10m-50m		WeChat	6.0.1	100m-500m	
SayHi	3.6	10m-50m		Zoosk	8.6.21	10m-50m	

Fig. 2. Discovered vulnerabilities per application. Since the App Store is not reporting the downloads, the numbers are from Google Play.

MoMo: While MoMo uses HTTPS to exchange messages with the server, it does not hide users' location. More precisely, the packets that are received from the app contain fine-grained distance information from other users. In addition, URLs contain the visited profiles as well as the current user ID and photos are sent over HTTP by using static links.

Plenty of Fish: It uses HTTP but all messages are encrypted, which most likely means that the app contains a local key to decrypt the contents. On the bad side, photos are sent over HTTP as static links.

SayHi: It uses Facebook for its authentication and then sends everything in clear text. Packets include the fine-grained location of other users and their activity can be seen in the requested URLs. An eavesdropper could also intercept user conversations. Photos are sent over HTTP using static links.

Singles around me: It sends the exact location of other users in the packet. Photos are sent over HTTP with some of them being dynamic links and others being static. However, the received packet contains an additional field: users' emails. In principle, a user needs to ask for the permission of other users to access their email, but this is always sent in the packet. This app significantly exposes users because URLs contain the IDs that a user has been watching. Hence, revealing his/her preferences and activity.

SKOUT: SKOUT uses HTTPS only for its authentication but the rest of the traffic is sent over HTTP. It sends the exact distance to other users in the packets and then obfuscates it in the frontend of the app. The API of SKOUT exposes the user activity because it shows whether the user is typing a message, visiting a profile, etc. Since traffic is sent over HTTP, chat messages are unencrypted.

Tagged: All traffic is sent over HTTP, so all messages can be intercepted and the API exposes user's activity and preferences. Photos are sent over HTTP as static links.

Tango: Tango transmits over HTTP and all messages can be intercepted. The API exposes user's activity as well as his/her phone number and preferences. Photos are sent over HTTP as static links and the messages contain the real location of other users.

Tinder: Tinder uses HTTPS traffic to deter eavesdroppers but the messages contain the Facebook ID of the users. Hence, an adversary can access even more data. Packets do not contain the actual location but the distance to other users, which can be further tuned to recover their actual locations. Photos are sent over HTTP as static links.

Tingle: Tingle does not use location data of other users in the received packets. However, messages contain other important information. Like Singles Around Me, it includes other users' emails and, additionally, it has a device tag indicating, for example, that the user has switched the device. Moreover, Tingle displays the actual location of the user in the URL, allowing an eavesdropper to identify him/her. The URL used by the API contains users' queries, hence, exposing their preferences. Finally, photos are sent over HTTP as static links.

Waplog: Waplog transmits over HTTP. While it does not send the location, it exposes emails of other users. Photos are sent over HTTP as static links. In addition, the API exposes information about the user's device in the URL, the session key and the user's hashed password.

WeChat: WeChat uses HTTP for all its traffic and sends all information in an encrypted file. In our experiments, we didn't notice any handshake between the application and the API to generate a cryptographic key. Therefore, we may safely deduce that the application is installed with a static hard-coded key, same for each user that can be derived by reverse engineering the application. Therefore, one could use the key to fully manipulate the data of all users.

Zoosk: Zoosk uses HTTPS for its traffic. The requested URLs expose the phone model and its OS as well as the user activity. Finally, photos are sent as static links over HTTPS.

5 Discussion

After analysing the above mobile dating and chatting applications we have observed several trends that are next discussed.

Users' locations: Many of the studied apps hand over the locations of other users so as to display them and show whether users are nearby. A typical example is illustrated in Table 1 where "Singles Around Me" sends a JSON file containing the exact GPS location of a user. This approach is susceptible to many attacks. It seems that a better solution would be to use distances instead of real locations. However, as shown by Qin et al. in [11], even distances can be used, via trilateration, to disclose the location of users. A better approach towards protecting users' locations, would be to use *Private Proximity Testing* protocols such as [12]. These protocols disclose a single bit of information, *i.e.* whether two users are nearby or not. By using this approach the desired functionality is provided whilst, at the same time, users' exposure is drastically reduced.

Unencrypted transmission channels: In [13] it is shown that many mobile apps use SSL/TLS code, which is potentially vulnerable to man-in-the-middle attacks. Notwithstanding, these apps were, at least, trying to protect sensitive user data. On the contrary, our analysis shows that major apps transmit private data over

```
{
 "username":"s---------eam",
 "email":"d----------96@yahoo.com",
 "gender":2,
 "interestedIn":1,
 "country":"United Kingdom",
 "region":"London, City of",
 "city":"",
 "gps":[38.------------2,23.8-------------5],
 "age":39,
 "photo":"http://www.singlesaroundme.com/images/cache/1/
    photoface_11-----_--5_--5.jpg",
 "photos":[],
 "birthYear":1974,
 "birthMonth":--,
 "birthDay":-,
 "lastOnline":"2014-10-06 03:28:07 PM",
 "profileAnswers":{"1":"5' 7" - 170cm",
 "3":"prefer not to say",
 "21":"Married","30":"straight","25":"brown","31":"blonde","2
    6":"white","28":"none",
 "29":"Sagittarius","38":["dating","serious relationship","
    friendship"],
 "37":"Greek","36":["English"],"32":"socially / occasionally"
    ,
 "34":"socially/occasionally","35":"quite fit","40":"
    Christian - Other",
 "41":"University graduate","42":"yes living with me","43":"
    yes"},
 "privacySettings":{"gps":0,"profile":0}
}
```

Table 1. Example of a JSON packet from "Singles Around Me".

HTTP instead of HTTPS. It could be said that using HTTPS might imply an overhead in terms of computation and bandwidth. However, in this kind of apps, where very sensitive information is managed, it is difficult to support the use of HTTP.

Multimedia: In general, our study shows that the handling of multimedia content is very poor. In [14], Patsakis et al. highlight the privacy and security risks related to the sharing of multimedia content in Online Social Networks. Note that the case of mobile dating apps could be considered even more sensitive and might imply further dangers. First, it is apparent that using HTTP (instead of HTTPS) allows an attacker to intercept and change the content of the received messages. Second, the use of static links can be considered a very important security vulnerability because an eavesdropper could easily find the images of the profiles that a specific user has visited. These images would allow the adversary to identify the sexual orientation and preferences of users. Depending on the openness of the society, giving away the precise sexual orientation and preferences of a user can lead to social discrimination or even legal actions.

Hidden information and URL parameters: Thanks to our analysis, we have noticed that in several apps, the data packets sent to users contain hidden information about other users. For instance, they contain the email of others, even when they have not consented. Table 1 illustrates this for "Singles Around Me". In the case of Tingle, the analysed data packets contained a device identification field that would allow an attacker to know when users change or switch their devices. Also, we have observed a doubtful development practice, this is, the addition of sensitive parameters in the URLs of the API. These parameters allow an attacker to obtain lots of information. For instance, users' activities (*e.g.* browsing of profiles, chatting, etc), the telephone number or even the location of the user could be monitored simply by eavesdropping the communications. Some examples of the information that can be leaked through the URL are illustrated in Table 2. While in all cases the traffic is encrypted, one can see that e.g. ChatOn broadcasts the IMEI, IMSI and user's phone number in the URL along with some details about the phone, Grindr broadcasts the user's location his ID and some data about the device. Similarly, MoMo leaks which profiles a user is visiting and SKOUT shows what the user is doing, in this case typing.

6 Conclusions

The steady growth in the number of users of dating services might get the attention of cybercriminals, who try to obtain personal information that could be used for user profiling, blackmailing, defamation, and even identity theft.

Web-based dating services are pretty well-established and well-known programming practices are put in place (in most cases). However, we have realised that in dating services provided by mobile platforms the situation turns out to be quite different. Mobile applications have the ability to collect very sensitive information (*e.g.* users current location, sexual orientation and preferences)

Application	URL
ChatOn	https://gld1.samsungchaton.com/prov4?imei=-----\&imsi=-----\&model=iPhone4\&clientversion=3.0.2\&platform=iPhone\%20OS\&osversion=7.0.4
	https://prov4?imei=------\&countrycallingcode=30\&phonenumber=-----\&imsi=----\&model=iPhone4\&clientversion=3.0.2\&platform=iPhone\%20OS\&osversion=7.0.4
Grindr	https://primus.grindr.com/2.0/broadcastMessages?applicationVersion=2.0.24\&hasXtra=0\&lat=53.-----\&lon=6.2----\&platformName=iOS\&platformVersion=7.0.4\&profileId=36850131
MoMo	https://api.immomo.com/api/profile/1121----?fr=98----
SKOUT	http://i22.skout.com/services/ServerService/GCUserTyping

Table 2. User exposure from the URL. For obvious reasons, the sensitive information has been suppressed.

and one might expect that their security measures are even more robust than those implemented on their web-based counterparts. However, our analyses have shown that there are significant security holes that could be easily used even by inexperienced attackers to obtain very sensitive information.

After analysing a significant number of diverse mobile dating applications, we have concluded that most of them are vulnerable to simple sniffing attacks, which could reveal very sensitive personal information such as sexual orientation, preferences, e-mails, degree of interaction between users, etc.

We interpret the results of our study in two main ways: First, we believe that disclosing these vulnerabilities might foster the interest of the society in protecting its privacy, and will raise an alarm for those companies that provide insecure services. Second, we believe that most of the detected vulnerabilities have very simple solutions that do not require much effort. Thus, by disclosing these problems we open the door to a reframing of the programming practices in mobile dating applications.

Acknowledgments

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the *OPERANDO* project (Grant Agreement no. 653704) and is based upon work from COST Action *CRYPTACUS*, supported by COST (European Cooperation in Science and Technology).

Dr. Solanas is partly funded by La Caixa Foundation through project "SIMPATIC" RECERCAIXA'12, by the Government of Catalonia under grant 2014 SGR 537, and by the Spanish Ministry of Economy and Competitiveness under project "Co-Privacy", TIN2011-27076-C03-01.

References

1. Cenzic, Application vulnerability trends report, Tech. rep. (2014).
   URL http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf
2. M. C. Grace, Y. Zhou, Z. Wang, X. Jiang, Systematic detection of capability leaks in stock android smartphones, in: 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012.
3. K. W. Y. Au, Y. F. Zhou, Z. Huang, D. Lie, Pscout: Analyzing the android permission specification, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, ACM, 2012, pp. 217–228.
4. A. R. Beresford, A. Rice, N. Skehin, R. Sohan, Mockdroid: trading privacy for application functionality on smartphones, in: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, ACM, 2011, pp. 49–54.
5. Y. Zhou, X. Zhang, X. Jiang, V. Freeh, Taming information-stealing smartphone applications (on android), in: J. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, Y. Beres (Eds.), Trust and Trustworthy Computing, Vol. 6740 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 93–107.
6. W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth, Taintdroid: an information flow tracking system for real-time privacy monitoring on smartphones, Communications of the ACM 57 (3) (2014) 99–106.
7. W. Enck, D. Octeau, P. McDaniel, S. Chaudhuri, A study of android application security, in: Proceedings of the 20th USENIX Conference on Security, SEC'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 21–21.
   URL http://dl.acm.org/citation.cfm?id=2028067.2028088
8. M. Egele, C. Kruegel, E. Kirda, G. Vigna, Pios: Detecting privacy leaks in ios applications, in: Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011, The Internet Society, 2011.
9. D. Wetherall, D. Choffnes, B. Greenstein, S. Han, P. Hornyack, J. Jung, S. Schechter, X. Wang, Privacy revelations for web and mobile apps (2011) 21–21.
10. A. Burattin, G. Cascavilla, M. Conti, Socialspy: Browsing (supposedly) hidden information in online social networks, CoRR abs/1406.3216.
    URL http://arxiv.org/abs/1406.3216
11. G. Qin, C. Patsakis, M. Bouroche, Playing hide and seek with mobile dating applications, in: N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, T. Sans (Eds.), ICT Systems Security and Privacy Protection, Vol. 428 of IFIP Advances in Information and Communication Technology, Springer Berlin Heidelberg, 2014, pp. 185–196.
12. A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh, Location privacy via private proximity testing, in: Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011, The Internet Society, 2011.
13. S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, M. Smith, Why eve and mallory love android: An analysis of android ssl (in)security, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, ACM, New York, NY, USA, 2012, pp. 50–61.
14. C. Patsakis, A. Zigomitros, A. Papageorgiou, A. Solanas, Privacy and security for multimedia content shared on osns: Issues and countermeasures, The Computer Journaldoi:10.1093/comjnl/bxu066.