Towards Practical Methods to Protect the Privacy of Location Information with Mobile Devices

Christoph Hochreiner, Markus Huber, Georg Merzdovnik and Edgar Weippl
SBA Research
Sommerpalais Harrach, Favoritenstrasse 16, 2. Stock, AT-1040 Vienna, Austria
{chochreiner,mhuber,gmerzdovnik,eweippl}@sba-research.org

ABSTRACT

Smartphones and tablet computers continue to replace traditional mobile phones and are used by over one billion people worldwide. A number of novel security and privacy challenges result from the possibility to extend the functionality of smartphones with third-party applications. These third-party applications require that users provide personal information to third-party applications in exchange for additional features. This paper focuses on one specifically sensitive information requested by third-party applications, namely: location information. We discuss current methods to protect the privacy of location information and evaluate two approaches in depth. First, we introduce an extension to improve the usability of current interception methods on an operating system level. Second, we evaluate the applicability of proxy-level interception on basis of real-world Android applications. Our findings significantly extend the state-of-the-art regarding the protection of location information on mobile devices and further highlight open research challenges.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection;
K.4.1 [**Computers and society**]: Public Policy Issues— Privacy

General Terms

Design, Security, Privacy

Keywords

mobile computing, location obfuscation, privacy protection, Android

1. INTRODUCTION

The growing usage of smart phones and tablets raises novel challenges for user privacy. People rely on third-party applications to extend the functionality of their phones, in exchange for transmitting personal information. Location-based applications are a growing category of mobile applications, which transmit sensitive information to third parties. According to a survey by Kessinger and Gellman [22] almost sixty percent of Americans use location-based applications despite privacy concerns. In this paper we explore strategies to protect the privacy of location information regarding mobile third-party applications. Location aware applications use the geographic location information of the device to provide convenient services, like the search for nearby points of interest (POI). Besides the retrieval of nearby POI, these applications are often used to inform other people about the current location or popular places [25], to record sportive activities [3], to retrieve weather forecasts, or simply to use the application for navigation purposes [21]. Although location aware applications are convenient, they also interfere with the privacy of their users [26, 24]. Security and privacy issues range from unauthorized collection of location information to interpretation of collected data to derive behavioral patterns. These patterns reveal a significant amount of private information about the user, like daily habits, shopping preferences and< working habits [13]. Users are often unaware that their privacy is violated by location aware mobile applications and are thus not concerned about the disclosure of their location information to third parties. In 2005, only about 25 % of users were concerned about the privacy implications which might occur after disclosing their location information [20]. Similar surveys by Kaasinen [21], Barkhuus et al. [6], and Benisch et al. [8] support these figures. Within the last years the sensitivity in disclosing location information however has risen [14]. The main reason for this increased sensitivity is that all major mobile operation systems implemented dedicated privacy settings. Besides unaware users, there are also privacy sensitive users, who want control on their location privacy or at least want to be informed on the location information which is transmitted to external services [8, 11]. The demand of control varies based on the application type respectively on the date and the time of the day. Typical examples for an increased requirement of location privacy are stays in a hospital or nightclub visits, while the location tracking in a foreign city on vacation might create less demand for additional privacy protection methods.

Beresford and Stajano [10] where among the first, who raised the awareness for privacy issues regarding location-based services. Since their publication there have been several researchers who attempted to provide a solution to this problem on an architectural level [29, 7, 18, 5] or by providing usable mitigation strategies, such as obfuscation algorithms [4, 23, 7, 12, 28]. Although there have been numerous projects and publications, no solution exists, which allows the user to use location-aware mobile applications while maintaining the privacy of the user at the same time. The main contributions of this paper are:

- We provide an overview on existing mechanisms to protect the privacy of location information of current mobile operating systems.

- We propose enhancements for state-of-the-art locations obfuscation methods for mobile applications.
- We implement our proposed privacy extensions and evaluate their feasibility based on real-world Android applications.
- We discuss the feasibility of location obfuscation on current mobile platforms and discuss open challenges.

The remainder of this paper is structured as follows. Section 2 provides a brief background on current methods to obfuscate location information on mobile devices. In Section 3 we present the design of our two proposed privacy protection systems and discuss their evaluation in Section 4. We discuss our findings in Section 5 to finally draw conclusion in Section 6.

2. BACKGROUND

With the goal to reduce the negative privacy impacts of location information, researchers as well as mobile operating system developers proposed privacy-preserving frameworks. This section provides an overview on different improvements to enforce location privacy on modern mobile operating systems as well as generic protection strategies.

2.1 OS-specific approaches

In 2013 the two most popular mobile operating systems were iOS and Android, which together accounted for over 90 % of sold mobile devices [15]. Therefore we concentrate on these Platforms and give a short overview on how they handle location information respectively.

2.1.1 iOS

Prior to the sixth major release of iOS, the operating system only allowed the user to enable or disable location services for all applications. In iOS 6 this setting was improved, by allowing the user to enable the location service for each application individually[1]. By default the location services are disabled for every application. Every application, which requests the location information, prompts the user to allow the application to access the location information. The user can then decide to enable the location access for this application or stick with the default setting and deny it. Besides the general settings, there are also user awareness features. Every time an authorized app obtains location information, an icon in the status bar indicates, that the location Application Programming Interface (API) was queried by an application. The user can then open the preferences menu to check which application requested the location information. This feature does not improve the privacy of the user, but it informs the user about issued location requests and therefore improves the awareness about the users location privacy. Due to the restrictive policies by Apple, it is impossible to alter the location information preferences without modifying the stock operating system. Root access on iOS allows the installation of applications, which are not available in the official Apple market. For example, Protect my Privacy is an unofficial application which is designed to protect the user's personal information on IOS devices [1]. To improve the location privacy, this application makes use of the already implemented location privacy settings and adds the functionality to provide a fake location for each application.

[1] `http://support.apple.com/kb/HT5467`

2.1.2 Android

The location privacy settings of current Android releases provide rudimentary options. The privacy settings only allow the user to globally enable or disable the location services but not for each application individually. Besides the global setting, the preferences also provide some functionality to set the location granularity for applications. The user can choose, if he wants to use either GPS - satellites, Wi-Fi and the mobile network location or both approaches to determine the location of the mobile device. Although this feature is intended to save battery power, it can also be used to simulate different levels of location granularity, by choosing one of the different location retrieval approaches. The GPS approach in urban areas and the mobile network location in rural ones reduce the accuracy of the location information and therefore improve the privacy of the user. Besides the minimalistic location privacy settings, the operating system also implements a visual notification upon location requests. Currently the only possibility to adjust the privacy settings is to install external privacy extensions, like PDroid[2], PrivacyGuard [29] or AppFence [19]. Some of these privacy extensions can be obtained from the Google Play store[3], while others require root access to install them. Besides the additional configuration capabilities, some extensions, like MockDroid [9] implement a mocking functionality to fake locations. Henne et al. [17] developed the "Android Location Privacy Framework" (ALPF) to extend the concept of providing only fake location information. Instead of only providing two choices, namely deactivate the location information or provide fake location information, they implemented additional algorithms, which can be used to obfuscate the actual location information. These algorithms provide different levels of location granularity and the user can select different algorithms to adjust the granularity according to his requirements. The ALPF framework is useful to provide people with privacy choices, but for an average user, the configuration possibilities of this application might be confusing. Both MockDroid and ALPF are based on Cyanogenmod[4], which is a developer friendly port of the Android code base. These two applications heavily interfere with the internals of the mobile operating system and it is not trivial to install them on mobile devices.

2.2 Generic approaches

Besides the different privacy extensions, which can be installed on existing mobile operating systems, there are also generic location obfuscation architectures, which we describe in the following.

2.2.1 Proxy

Bellavista et al. [7] propose a proxy system to change the location granularity according to user- as well as service-requirements. The proxy consists of a client proxy which is compulsory and a server proxy which is optional. The client side proxy is deployed on a mobile device and the user configures this proxy by entering a location granularity level, which is acceptable for him. The server side proxy is deployed on the server, which provides the service, and the configuration describes the minimal level of location granularity that is required to provide the service. This architecture provides two different procedures to improve the privacy of the user. If the system only contains a client side proxy, the proxy reduces the location granularity according to the user's configuration and transmits the location information to the server. In the second scenario,

[2] `https://play.google.com/store/apps/details?id=com.privacy.pdroid`
[3] `https://play.google.com/store`
[4] `http://www.cyanogenmod.org`

when the system consists of a client side proxy and a server side proxy, these two proxies open a secured connection and negotiate a location granularity level based on the provided configurations. The goal is to provide as little information as possible, while maintaining the quality of service. This system is a rather theoretical approach and the authors made recommendations on how this system could be embedded into existing mobile operating systems.

2.2.2 Framework

The framework-based approach requires a modification of the underlying mobile operating system and additional changes regarding third-party applications. Such frameworks introduce a more detailed location API, which can be used to negotiate the level of location granularity. The Confab toolkit [18] provides such a comprehensive framework, which can be used by application developers to easily implement privacy-sensitive applications. The framework consists of a data model, which represents different privacy preferences issued by a user, e.g. the location granularity, and of a library which can be embedded into applications. The library provides different methods and policies, which can be used to implement the communication among clients and the server. Each component of the system has to implement the library in order to create a location privacy aware system.

2.2.3 Trusted Third Party (TTP)

The TTP approach adds an additional external component to the already existing system. The existing system consists of applications, which are installed on mobile devices, and service providers, which are deployed on external servers. The TTP component is introduced between the application and the service to obfuscate the location information by means of an implemented location obfuscation algorithm. While the flow of the request message is identical to other architectures, the response message is routed over the TTP component. This routing process is transparent for the server, so that the server does not detect the application of obfuscation algorithms. There are several different proposals, which make use of the TTP concept, such as Caspar [27] or Privacy Grid [5]. Both systems implement the K-anonymity algorithm and implement a similar architecture to preserve the privacy of the user.

Although all of the presented location obfuscation methods can be used to improve the location privacy of users, they also have a number of disadvantages. While the commercial approaches are intuitive and provide a high level of usability, they lack the functionality to assign different levels of location granularity for each application individually. One of the most sophisticated extensions is the selective cloaking extension by Henne [17]. This application allows appropriate configuration possibilities, but there could be some usability improvements. These usability improvements should support the user either by selecting appropriate default algorithms or clustering similar applications to reduce the configuration effort. Besides the usability aspect, this extension is built upon Cyanogenmod and requires a significant amount of effort to be deployed on mobile devices. Generic approaches suggest different promising ideas but often do not apply for state-of-the-art mobile devices. The major drawback of the framework-approach is, that they have to be implemented by every application developer. It can be argued that application developers do not employ these privacy protection measures because a reduced location granularity does not provide any competitive advantage over other applications. The last approach, the TTP, provides a location obfuscation architecture, which can be added on top of the existing application infrastructure. The addition of a TTP does not require any changes to the mobile operating system, besides routing the traffic over the TTP component. Although this approach is promising in terms of implementation for already existing systems, the problem of trust is ultimately shifted from the application server to the TTP component. The major difference between the proxy and the TTP approach is, that the proxy approach is transparent for the applications while in the TTP scenario, the application makes active use of the TTP.

3. DESIGN

This section outlines the design of two different approaches we suggest to tackle the challenge of location privacy on mobile devices.

3.1 Default Situation

All modern mobile operating systems implement a very similar architecture in terms of location information retrieval and communication with external servers. A typical workflow is started by a user who triggers a location -based service within an application and the application presents the result to the user. Hence, the workflow focuses on the retrieval of the location information and the communication between the application and external servers. The user initiates the workflow by requesting a location-based service from an application, which is installed on a mobile device. The mobile device issues a request to retrieve the location information from the location provider (e.g. Wi-Fi, GPS). The location provider gathers the actual location information and returns it to the application. Based on the location information the application then creates a request, which is transmitted to the dedicated server. The server performs different operations and generates a machine-readable result which is sent back to the application (e.g. Weather:Pittsburgh:7:Fahrenheit). At the end of the workflow the application processes the machine-readable result and presents it to the user.

3.2 Proxy-based interception

The proxy based interception approach makes use of an external component to intercept all outgoing traffic from the mobile device. This external component acts as a proxy and is capable of extracting the payload out of normal traffic as well as breaking secured communication to extract the application payload. The payload contains all kind of information, including the location information of the user, which is transmitted between the mobile application and the third-party application server. The proxy filters the incoming payload based on a predefined rule-set and passes all messages, which contain location information, on to an obfuscation service. The obfuscation service modifies the location information and returns the message to the proxy, which transfers the obfuscation location information to the original target server. The application server only receives location information that is conform with the privacy policy of the user.

The most important advantage of this approach is its usability because users can easily deploy this interception mechanism on a mobile device. The user is only required to change the network configuration of the mobile phone and to add the proxy's HTTPS certificate. In addition, this interception approach is operating system agnostic from a technical point of view. Proxy-based interception provides a usable and independent location obfuscation technique but requires preparatory work. Mobile applications do not use a single standard to exchange messages with servers and customized interception templates are thus required.

3.2.1 Architecture

The system design for the proxy-based interception requires two additional subsystems as compared to the default setup. These additional systems are deployed outside of the mobile device on dedicated hardware. In the following we discuss the modifications to the default service request workflow and provide a short description on the additionally required components. The first part of the default workflow (see 3.1) remains identical but the request to the application server changes. Figure 1 outlines the architecture of the proxy-based interception method. When the application tries to send the request directly to the server, the request is intercepted. The interceptor analyses all messages, which originate from the mobile device and filters out messages, which are known to come from location based applications and contain location information. If the interceptor finds such a message, it is passed on to the message utility component, which extracts the location information from the message. Besides the location information, this message utility also extracts the unique ID of the application, which created the request. The unique ID and the location information are returned to the interceptor who passes them on to the obfuscation component. The obfuscation component retrieves the obfuscation configuration from a database, based on the unique ID. It consists of the required algorithm and further obfuscation relevant properties. Based on the configuration, the obfuscation component then selects the designated location obfuscation algorithm and obfuscates the location information. The result of this obfuscation procedure, the obfuscated information, is returned to the interception component. Once more the interceptor makes use of the message utility and replaces the actual location information with the obfuscated one. The modified message now contains the obfuscated location information and is passed on to the dedicated server. As soon as the server receives the message, the result is generated and encoded in a machine-readable structure. This structure is then directly returned to the application. The application interprets the machine-readable results and represents the result to the user in a human readable manner. The workflow above describes a successful message interception, which is only possible if the location based application and the structure of the messages, which contain location information are registered in advance. For all other messages, the interception component performs a wild card search, where the messages are scanned for the keywords *long* and *lat*, which are commonly used to transmit location information respectively the current coordinates. If any message contains these keywords, the interception utility stores the message as a basis for further signature creations. In contrast to the proxy provided in the literature [7], this proxy does not require any interaction with the application developers within their applications on the mobile device as well on their external servers.

3.3 Operating system level interception

An alternative to the proxy based interception approach is the interception of location information on an operating system level and its obfuscation before it even reaches the application. This approach requires the modification of the mobile operating system itself. Therefore, this interception method is only feasibly for open source operating systems such as Android. For closed source systems, e.g. iOS, reverse engineering would be required in order to perform the necessary system modification. The actual modification of the operating system consists in implementing a wrapper for the location API to modify the location information based on specified user policies. The downside of this approach is the setup phase whereas a considerable amount of work is required in order to modify the mobile operating system. Interception on an operating system level is therefore not applicable on a large-scale because it is not easy to set-up by the average user.

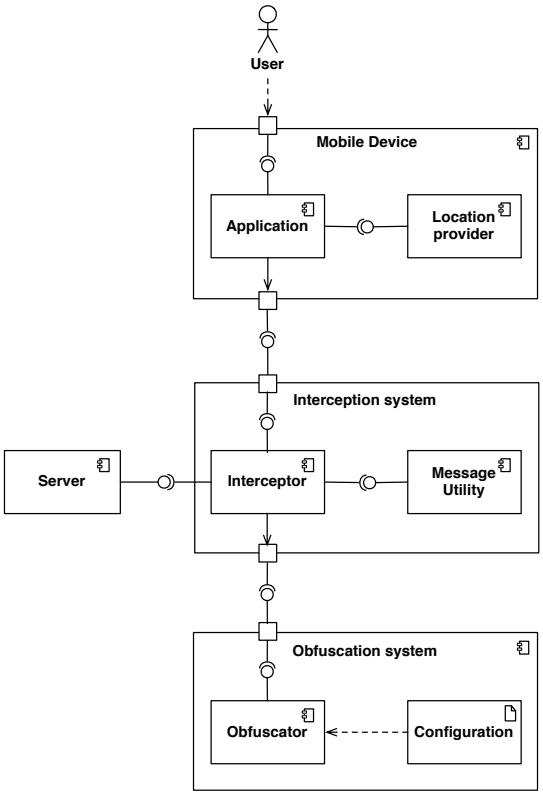

Figure 1: Proxy architecture

3.3.1 Architecture

The system design for the system level interception includes three additional components compared to the initial setup. All of these additional components are deployed on the mobile device and the message flow becomes more complex as compared to the default setup. The trigger for this modified workflow is the same as the one in the initial setup (see 3.1). Figure 2 outlines the architecture of the system level interception method. The user requests a location-based service from an application, which is deployed on the mobile device. In order to provide this service, the application tries to retrieve the location information from the location provider. As a result of the modification of the mobile operating system, an additional component intercepts this location request and stores the unique ID of the requesting application internally. The interceptor then issues a new location information request and retrieves the actual location information from the location provider. The result of this newly issued request and the unique ID of the application are then forwarded to the obfuscation component. The obfuscation component is designed to modify the actual location information according to a default privacy policy or a privacy policy issued by the user. An automatic process carries out the categorization of a newly installed application. Based on the category assignment, the obfuscation component retrieves the obfuscation configuration for the specific application from the internal database. The actual location obfuscation is carried out, by calling the obfuscation method with the retrieved configuration and the actual location information. Based on the configuration and actual location information, the obfuscation method generates a new location information. This newly

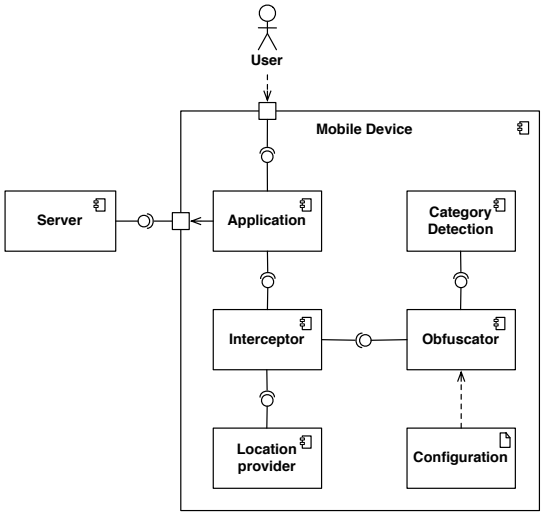

Figure 2: System interception architecture

generated location information is then returned to the interception component, which returns the generated location information to the issuing application. The remaining part of this workflow is identical to the initial setup.

The execution of this workflow is carried out transparently for the application as well as for the user. The user does not notice any differences, between the workflow in the initial situation and the workflow for the system level interception. The requested result is also identical, assumed that an appropriate location obfuscation algorithm was selected.

3.4 Obfuscation of location information

For the actual obfuscation of the location information, there are several obfuscation algorithms available. These algorithms range form very simple spatial approaches [4] over time based obfuscation up to location information based on semantic information about the environment [12, 7].

The privacy improvements provided by a location obfuscation algorithm depends on the use case scenario of the user and the selected obfuscation algorithm. A more detailed description of location obfuscation algorithms is out of scope for this publication, but a good starting point are the classifications provided by Wernke [30] and Andersen [2].

4. EVALUATION

This section describes the prototypical implementation and evaluation of privacy protection methods for location information.

4.1 Proxy based interception

We created a prototypical implementation of the proxy based interception method in Java and in addition relied on mitmproxy[5] to intercept the HTTPS traffic between mobile phones and application servers. To evaluate our approach we deployed the interception proxy on a Raspberry Pi (Model B) and used a second Raspberry Pi to perform obfuscation operations. The sample for our evaluation consisted of the top 100 free applications of the Google Play store and was collected in December 2013.

[5] `http://mitmproxy.org`

4.1.1 Evaluation

In the first phase, all top 100 free applications were installed on a Nexus S Android device. In the course of the installation phase, all applications, which did not request access to the location API, were excluded from further evaluation. The succeeding phase was used to identify whether the applications transmit the location information to an external server and if so, how this location information is encoded. To identify the location information usage of the application, the application was started and events were manually triggered. The proxy recorded all messages, which were sent since the start of the application until the application was terminated. The message utility analyzed the messages, to verify whether they contained the keywords *long* and *lat* or the current coordinates. If these keywords were present, the message and the request URL of the message were stored in the configuration. This basic heuristic offers a suitable approach to identify the majority of all messages, which transport location information. The messages of the remaining applications were either identified by querying the messages with the actual coordinates respectively by manual analysis. In the following we created unique signatures based on the application target hostnames. The stored messages were further used to create location information extraction templates. This process was executed for all applications, which requested access to the location API. In the last phase, the applications were started again and the same events were triggered. During this final phase we verified if the signatures work as indented and if application developers use countermeasures against the modification of messages by a proxy.

4.1.2 Results

Figure 3 outlines the findings of our evaluation. Among the 100 applications, there were 43 applications, which requested access to the location API, but only 31 of these applications actually communicated the location information to an external server. This implies that twelve applications requested access to the location API but did not make use of this permission.

Among the 31 applications, which transmitted the location information to an external server, only half of them actually provided location-based services for the user. The other half only uses the location information for statistical purposes or as a parameter for ad-networks, which was a very common use case within the analyzed applications. The transferred location information could be intercepted and modified for the majority of applications (21). For ten applications the modification of location information was however not possible. Seven out of these ten applications used certificate pinning to prevent the interception of application traffic. Certificate pinning describes a process, where applications additionally check the provided certificate against trusted validation data. The remaining three applications provided photo enhancement features and embedded the location information into transferred pictures within the Exchangeable Image File Format (EXIF). Our prototype focused on the modification of exchanged messages and did not modify embedded location data. Finally, our evaluation shows that popular applications often use location information for advertisement purposes. Location information is thus transferred to third-party advertising companies and is not used for additional application features. Our results also highlight that different applications relied on the same advertising providers. Hence, we could create signatures for these advertising providers and reuse them to modify the location information for several applications. Our findings showed that for further iterations of the prototype it might be useful to compile a list for commonly used advertisement providers, which can be shipped as an initial configuration.

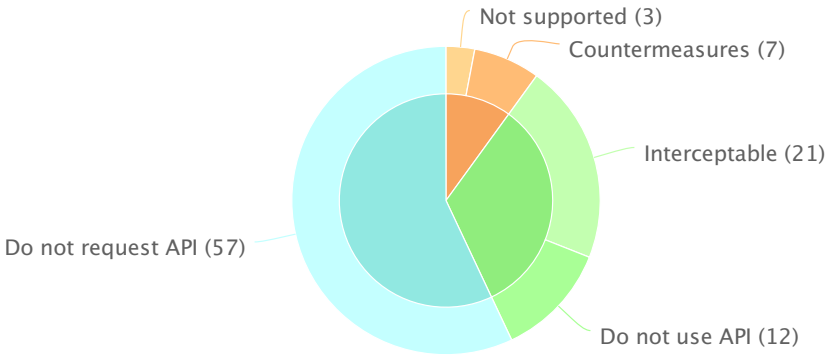

Figure 3: Usage of location information and applicability of proxy interception

The proxy implementation introduces some overhead to the default communication workflow and therefore, the location obfuscation requires additional time. In our test scenario, the additional required time ranged from 0,1 seconds for simple obfuscation algorithms to 0,5 seconds for more complex ones. This additional overhead can be neglected in comparison to the time required to locate the mobile device and therefore it very likely that a user would not even detect the usage of such a privacy improving proxy.

4.2 Operating system level interception

We based our prototype on the Android Location Privacy Framework (ALPF) [16, 17] and used a Nexus S device to evaluate this approach. In a first step we ported the existing source code to the latest stable version of Cyanogenmod (v10.2). In comparison with our proposed design, the original source code also lacked the feature of automatic clustering of Android applications. Hence, we extended the original source code to automatically group applications based on the category assignment of the Google Play Store and assign them to appropriate location obfuscation algorithms. In order to retrieve the category assignment of the Google Play Store we added a web-scraping component.

4.2.1 Evaluation

We could successfully port the existing ALPF project to the latest version of Cyanogenmod (v10.2). Furthermore we verified that the automatic category assignment functioned as intended. Our extended version of ALPF improves the usability of operating system level interception. Weather applications are for example automatically grouped into an obfuscation class, which removes the detailed location information but resolves to the same city. Routing applications are whitelisted in terms of obfuscation and games only obtain fake location information instead of genuine ones. Upon successful publication we plan to submit our improved version to the ALPF open source project. The performance evaluation of the operating system level interceptions yields very similar results as provided by the ALPF framework [17].

5. DISCUSSION

5.1 Mobile Location Obfuscation Techniques

A number of researchers proposed methods to protect the privacy of location information with mobile devices. The framework and TTP approach require developers to incorporate additional functionality in their applications. To the best of our knowledge we are not aware of any mobile applications, which implemented these additional features to protect the location information of their users. Alternative approaches use a different strategy, namely: location information is intercepted and modified before it is received by an application or the third-party application servers. An interception at an early stage, for example between the location information provider and the application requires modifications to the underlying operation system itself. This method has been used successfully in research but due to the required installation effort is not feasible for the average user. Another approach consists of the interception of location information between applications and their third-party servers. In this scenario a proxy is set up, which intercepts the communication of mobile application with servers and modifies the exchanged location information. Compared with system-level interception this approach is easier to set-up by the average user, but requires significant additional implementation effort. In the current scenario, the proxy is deployed outside of the mobile device. This could lead to trust issues. In a further iteration, this proxy could also be deployed on the mobile device of the user and the rules are deployed by means of an subscription service. In this scenario, the user has full control over the proxy component and therefore the potential trust issues with an external proxy component are resolved.

In practice only the proxy approach and the system level interception approach are feasible. All other approaches require the cooperation with application developers, who have no intention to reduce the quality of the available information. The best solution for this interception problem would be that mobile operating system developers either provide an API, which can be used to register obfuscation algorithms or a graphical user interface, where the user can configure already implemented algorithms.

5.2 Comparison of different frameworks

Except from theoretical concepts to protect location information on mobile devices, a number of practical implementations exist. In the following we discuss solutions for the two most prevalent mobile operating systems, namely iOS and Android. Hereby, we compare six different solutions: our two implementations (proxy based and system level interception), Android Location Privacy Framework (ALPF), PDroid, and the default functionalities of iOS 7 and Android 4.3. Table 1 provides an overview of the different frameworks.

	Proxy based interception*	*System level interception**	*ALPF*	*PDroid*	*IOS 7*	*Android 4.3*
Disable location information provider	✓	✓	✓	✓	✓	✓
Disable location information for selected applications	✓	✓	✓	✓	✓	
Spoof location information	✓	✓	✓	✓		
Granular obfuscation	✓	✓	✓			
Applicable for all applications		✓	✓	✓	✓	✓
No system modification	✓				✓	✓
Clustering of applications	✓	✓				
Automatic categorization of applications		✓				

Table 1: Comparison of privacy protection frameworks for location information including our two evaluated approaches ($\star$).

One can observe that the latest version of iOS provides more detailed location privacy settings as compared with Android. On Android it is only possible to enable or disable the location provider for all applications while for iOS it is possible to enable the location provider for each application individually. Besides the individual configuration possibilities, one can also enable or disable the location provider globally for all applications on iOS. This default capability is already provided by both operating systems and no additional system modification is necessary. Besides the two operating systems, the application PDroid was selected, because it is representative for several privacy improvement applications for Android and iOS. PDroid and similar applications provide the additional functionality to spoof location information. ALPF in additions allows to configure the granularity of location obfuscation. Both PDroid and ALPF require a modification of the underlying operating system. Our system level interception prototype extends ALPF to cluster different applications into categories and by implementing an automatic categorization procedure to categorize newly installed applications. We expect that the clustering functionality we introduced improves the overall usability of ALPF significantly. With the clustering capabilities in place, it is straightforward to update location privacy configuration for a large set of applications at once instead of updating them for each application one after another. For example, weather applications are clustered and assigned to a location obfuscation algorithm which provides only the current city but not the exact location. The last framework in this comparison is the proxy based interception approach. The proxy based interception approach provides very similar obfuscation capabilities as the system level interception approach in terms of categorization and configuration. Due to the complex categorization procedure, which requires the definition of new signatures, this prototype does not provide any automatic category assignment procedures. In addition our findings show that the modification of location information is not applicable to all applications. Some applications used certificate pinning which made the interception of communication between the application and its external servers impossible without obtaining root access to the device. The main advantage of our proxy based prototype however is that it can be deployed to all modern mobile operating systems. Overall one can observe that system level interception frameworks currently provide the largest set of capabilities in terms of usable location privacy protection.

6. CONCLUSION

This paper discusses possible solutions to the research challenge of protecting the privacy of location information sent to third parties by mobile applications. We first outlined currently implemented features of mobile operating systems to limit the extend of transmitted location information as well as suggested solutions from state-of-the-art research. In the following we developed an improved software design for two promising location obfuscation methods, namely proxy-based and system-level interception. Additionally we evaluated the feasibility of these two previously mentioned approaches on with an Android based device. Our first prototype extended the functionality of the Android Location Privacy Framework (ALPF) to provide automatic clustering and category assignment of mobile third-party applications. The second prototype implemented a proxy-based method for location obfuscation and we evaluated its feasibility based on popular real-world Android applications. Our findings showed that automated category assignment, based on an existing knowledge base like the Google Play Store of Android applications is feasible and furthermore highlighted possible limitations of proxy-based interception methods. We will release our prototypes under an the Apache License Version 2.0 to make our results available to the scientific community upon publication of our research.

Future research: We plan to extend the functionality of our obfuscation proxy to handle embedded location information such as EXIF information in pictures. Furthermore, we plan to create a comprehensive repository of application signatures for the proxy-based approach in order to detect commonly used third parties, such as advertising providers, in an automated fashion. Such an repository will reduced the configuration effort for the proxy based solution dramatically and makes this privacy preserving approach feasible for practical use.

Acknowledgments

This research was funded by COMET K1, FFG - Austrian Research Promotion Agency. Moreover this work has been carried out within the scope of u'smile, the Josef Ressel Center for User-Friendly Secure Mobile Environments. We gratefully acknowledge funding and support by the Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG and NXP Semiconductors Austria GmbH.

7. REFERENCES

[1] Agarwal, Yuvraj and Hall, Malcolm. ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. *MobiSys*, 2013.

[2] Andersen, Mads Schaarup and Kjærgaard, Mikkel Baun. Towards a New Classification of Location Privacy Methods in Pervasive Computing. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 150–161. Springer, 2012.

[3] Anderson, Ian and Maitland, Julie and Sherwood, Scott and Barkhuus, Louise and Chalmers, Matthew and Hall, Malcolm and Brown, Barry and Muller, Henk. Shakra: tracking and sharing daily activity levels with unaugmented mobile phones. *Mobile Networks and Applications*, 12(2-3):185–199, 2007.

[4] Ardagna, Claudio Agostino and Cremonini, Marco and Damiani, Ernesto and di Vimercati, S De Capitani and Samarati, Pierangela. Location privacy protection through obfuscation-based techniques. In *Data and Applications Security XXI*, pages 47–60. Springer, 2007.

[5] Bamba, Bhuvan and Liu, Ling and Pesti, Peter and Wang, Ting. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th international conference on World Wide Web*, pages 237–246. ACM, 2008.

[6] Barkhuus, Louise and Dey, Anind K. Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns. In *INTERACT*, volume 3, pages 702–712. Citeseer, 2003.

[7] Bellavista, Paolo and Corradi, Antonio and Giannelli, Carlo. Efficiently managing location information with privacy requirements in wi-fi networks: a middleware approach. In *2nd International Symposium on Wireless Communication Systems*, pages 91–95. IEEE, 2005.

[8] Benisch, Michael and Kelley, Patrick Gage and Sadeh, Norman and Cranor, Lorrie Faith. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal and Ubiquitous Computing*, 15(7):679–694, 2011.

[9] Beresford, Alastair R and Rice, Andrew and Skehin, Nicholas and Sohan, Ripduman. MockDroid: trading privacy for application functionality on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. ACM, 2011.

[10] Beresford, Alastair R and Stajano, Frank. Location privacy in pervasive computing. *Pervasive Computing, IEEE*, 2(1), 2003.

[11] Brush, AJ and Krumm, John and Scott, James. Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 95–104. ACM, 2010.

[12] Damiani, Maria L and Bertino, Elisa and Silvestri, Claudio. Protecting location privacy through semantics-aware obfuscation techniques. In *Trust Management II*. Springer, 2008.

[13] Farrahi, Katayoun and Gatica-Perez, Daniel. Daily routine classification from mobile phone data. In *Machine Learning for Multimodal Interaction*, pages 173–184. Springer, 2008.

[14] Fisher, Drew and Dorner, Leah and Wagner, David. Location Privacy: User Behavior in the Field. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*, pages 51–56. ACM, 2012.

[15] Gartner. Worldwide smartphone sales in November 2013. `http://www.gartner.com/newsroom/id/2623415`, 2013. [Online; accessed 30-November-2013].

[16] Henne, Benjamin. Android Location Privacy - A location obfuscation framework for Android. `http://bhenne.github.io/android-location-privacy/`. [Online; accessed 30-November-2013].

[17] Henne, Benjamin and Kater, Christian and Smith, Matthew and Brenner, Michael. Selective cloaking: Need-to-know for location-based apps. In *Eleventh Annual International Conference on Privacy, Security and Trust (PST)*, pages 19–26. IEEE, 2013.

[18] Hong, Jason I and Landay, James A. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189. ACM, 2004.

[19] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652. ACM, 2011.

[20] Junglas, Iris A and Spitzmuller, Christiane. A research model for studying privacy concerns pertaining to location-based services. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 180b–180b. IEEE, 2005.

[21] Kaasinen, Eija. User needs for location-aware mobile services. *Personal and ubiquitous computing*, 7(1):70–79, 2003.

[22] Kessinger, Kristen and Gellman, Marv. Geolocation Use and Concerns Survey. *Tech. rep. ISACA*, 2012.

[23] Krumm, John. Inference attacks on location tracks. In *Pervasive Computing*, pages 127–143. Springer, 2007.

[24] Krumm, John. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[25] Lindqvist, Janne and Cranshaw, Justin and Wiese, Jason and Hong, Jason and Zimmerman, John. I'm the mayor of my house: examining why people use foursquare-a social-driven location sharing application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2409–2418. ACM, 2011.

[26] Minch, Robert P. Privacy issues in location-aware mobile devices. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. IEEE, 2004.

[27] Mokbel, Mohamed F and Chow, Chi-Yin and Aref, Walid G. The new Casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[28] P. Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6):1010–1027, 2001.

[29] Stach, Christoph and Mitschang, Bernhard. Privacy Management for Mobile Platforms–A Review of Concepts and Approaches. In *14th International Conference on Mobile Data Management (MDM)*, volume 1, pages 305–313. IEEE, 2013.

[30] Wernke, Marius and Skvortsov, Pavel and Dürr, Frank and Rothermel, Kurt. A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, pages 1–13, 2012.