Predicting data leaks in Health apps

Joshua Mwandu University
of Mary Washington
1301 College Ave,
Fredericksburg, VA 22401
jmwandu@indiana.edu

ABSTRACT
Based on data readily available about health apps in the goggle play store combined with network data is it possible to predict if a data leakage will occur before downloading an app. On the market there's an abundance of health apps each of which contain highly sensitive and identifying information making it important that consumers are in a position to make the most informed decision about who to trust their privacy too. During our investigation of this question, we collected information on 250 health apps and trained a machine learning classifier to determine if an app will leak data.

Side note more will be written about the machine learning phase once the data collection phase ends.

Categories and Subject Descriptors
Health Informatics [**Machine learning**]: Mobile App Security

Keywords
privacy, classification

1. INTRODUCTION
Applications on consumer mobile devices are constantly leaking data to third parties without consumers being aware or having many options of recourse besides deleting the troublesome application. In this age of high data transmission many consumers both young and old perceive their mobile devices to be just as private if not more so then their desktop or laptop computer. Even though people are willing accept that many apps on the market use their personal data as currency what is desired is more control over the entities that have access to this information. Depending on the nature of the information being leaked, a third party advertiser can have the ability to track a consumer across different apps and develop a more revealing picture of their digital lifestyle. Raising the question of how much personal information on a mobile phone should be publicly available? Currently, there is a lack in the availability of easy to use tools to determine whether an application will leak data ranging from third party applications to operating system support that makes this task more difficult if a consumer doesn't have specialized knowledge or high computational power. We seek to display in this paper if it's possible to predict if an application is more prone to a data leakage based its characteristics. This work focused on health applications specifically as they contain highly rich and identifying information which needs an extra level of care when deciding who has access to the distribution of this knowledge. Machine learning was applied in the construction of a predictive model when attempting to determine if an application will leak information or not. This model was designed to focus in on whether or not an application has any features readily available on the Google play store which are particularly informative in the determination of information leakages. What we hoped to display using this model was that its possible to discern data leakages in an offline manner that doesn't require heavyweight tools nor need to be run dynamically with the application to convey knowledge about what applications are doing with personal information. Currently many of the options available to consumers can potentially be technically intensive such as rooting the device and tainting all the packets to track and block them. The biggest issue with this though is that many consumers wanting to use health applications might not possess knowledge of how to root their device nor understand the potential security implications that may arise when rooting an application. Other solutions may revolve around decompling the application code or using a third party server to block packets that aren't destined for the applications domain. One key thing to consider with either of these options is that the specialized software used to determine if a leakage is occurring either needs to be run on a machine with greater computational potential then a mobile device or requires use of a third party to determine if information is being leaked which carries it's own set of issues. The impact of this work is to give consumers an additional tool that can be used to help guide decision making for whether an application should be downloaded or not. Also researchers and software developers in the creation of future health applications that handle sensitive data with the goal of giving them a model that can help make there apps more secure.

2. RELATED WORK
2.1 Static Methods and Tainting

The current literature focuses on data leak recognition in order to prevent these occurrences whilst they happen. These approaches range from static methods such as decompiling the code to determine the pathways that result in a data leakage to more dynamic methods such as rooting the device and tainting all the packets the device generates so that they can be tracked and blocked when necessary[6]. The problem with these methods is that they suffer from either a high false positive rate, require specialized knowledge (such as how to root the device if packet tainting is being used) or have a high computational cost (such as when decompiling the code is being used).

2.2 VPN Based Approaches

Another technique is to use a VPN (Virtual private Network) on the application level, such as Privacy Guard or Friend in the Middle, and this works by blocking PII (Personally identifying information) or other forms of data leakages by blocking traffic through at all or if it's not going to it the apps domain. This works on the application level and doesn't require rooting the device or other advanced technical knowledge for use; therefore has as advantage over the other technique from a consumer standpoint. [22]. Recent work by [19] explored using machine learning to determine if PII information is being leaked at the network level. This had the benefits of being deploy able across multiple platforms such as android and IOS with a reasonable level of detection when compared to other methods.

2.3 Where is your information going?

A study by JinYan Zang which tested 110 popular free android and ios apps found that 73% of the Android apps in there sample shared personal information (for example email addresses) whilst 47% of IOS apps shared Geo-coordinates to third parties. [27] Unsurprisingly the two largest third party domains according to their data where Google and apple.

2.4 Introducing new privacy abstractions

Other approaches at dealing with privacy leakages, such as CleanOS, revolved around providing new abstractions for data management by taking sensitive data and turning it into a sensitive data object (SDO) [23]. When this SDO becomes idle a specialized garbage collector takes it out of memory and sends it into the cloud where it can later be requested only by applications that have specialized keys.

2.5 Peoples Opinions on Privacy

The literature also contains a good volume of information written on consumer preferences about how they feel their sensitive data should be collected and the response consumers have to finding out specific uses that applications are doing when distributing data. Jan Boyles collected survey data and found that 54% of her survey participants would not install an app because of how much information it would collect about them [2]. This has an interesting relationship with on of the core theme in here paper which was that smart phone users are more likely the non-users to back up the contents of there phone regardless of their demographic [2].

2.6 This works contribution

The goal we're seeking to accomplish is if it's possible to determine, based on the apps characteristics if it will leak data. If so, then this will help build on other methods already published or still being researched by providing a framework that essentially enables these services to pay more attention to some apps over others before they've gone through any network activity. The other quality that this seeks to provide is to give consumers more control in deciding which apps they'll allow to access their data.

3. METHODS

3.1 Data collection

The data used for training and testing was collected by a physical Android device and several virtual ones this enabled more data to be collected at a faster rate. Network data (containing the destination of the packet and the nature of the data being leaked) was collected by the Friend in the Middle app acting as a VPN (Virtual Private Network) on the application level. Friend in the Middle is an extension of Privacy Guard that has the capabilities to detect whether an app is leaking data as its leaking data. What we did was cycle through a large selection of health apps then collect the network traffic data generated by Friend in the Middle so that it could be used to aid in the training of a classifier. The network traffic was sent to a specialized log file that we created by modifying the logging system in friend in the middle so that it could provide a more detailed breakdown about the data each app was leaking. This log file contained information such as the app name, the nature of the data leakage, the destination IP address, Geo-location data, and the message data that was being leaked.

3.1.1 Collecting Network data

The acquisition process for collecting network data involved running each application and then waiting for it to trigger a notification by Friend in the Middle. For the majority of applications that leaked, these leakages would occur upon the application start up, although there where apps in which it took longer for them to leak. Every application that was tested was actively used for 30 minutes or until it leaked. If the 30 minute period of active usage ended with the application not triggering a notification for a data leakage occurring, then the app would be left to run in the background for an hour. If the app didn't leak after the one hour period of background use ended then we would move on to the next app and periodically select an app at random to be retested. The reason for this procedure is that apps wouldn't leak during the initial usage period or would take over an hour to leak data. After a data collection session the log file generated by Friend in the middle would be extracted from the android device used for testing and a python script would build a list of all the pertinent health apps that leaked data. The script would also inspect that list of apps that didn't leak and ensure that there was no intersection between the set of leaked apps vs the non-leaking apps. We had another script that was deciding which app to do next and kept track of all the remaining apps to be tested ensuring that we weren't retesting an app and we went through all the apps in the data-set.

3.1.2 Collecting Meta Data

In addition we also collected data readily available from the Google play store about various mobile health applications. This information contained details about attributes of apps such as their title, developers, download rate, permissions and other qualities that we used to both label the data and build a portfolio of features to browse through. In order to gather attribute data about each application from the Goggle Play store we used an open source scraper that was written in node.js and python this ensured that information about each app remained consistent and guaranteed expedient data collection. The attribute data collected was the app title, developer, min installation rate, max installation rate, url, score, review number, descriptonalHTTML, latest update, version,size, requiredAndroidVersion, contentRating, genre, genreID, price, developerEmail, developerWebsite and permissions.

3.2 Training

A python script was written that took all the meta-data from the Google play store, in addition to the information conveyed by the network data detailing whether the app in question leaked, and produced a formatted text file on the apps that was used for training. This formatted file contained all of the Google play store meta-data and one Boolean value entitled leaked. Once a large enough data repository was built from the various health apps in circulation, we trained a classifier to make a predication on the leak status of an app. This phase involved not only stringent feature selection inspection but also pitting various algorithms head to head and analysing their performance and computational cost. The classifier chosen was the machine learning algorithm that achieved the greatest degree of accuracy whilst being able to do this utilizing the least resources.

The data repository contained 250 free health applications that where placed into either the testing or training phase of the project. The data was then divided into ten buckets so that a cross validation phase could be used in determining the predictive power of the modal.

4. RESULTS

4.1 Findings

Our investigation into health apps revealed that most health apps don't leak data to third parties based on the features that we were detecting on. Friend in the Middle detects leaks in three categories contacts, location and phone state. From our data-set no apps leaked contact information to third parties but apps were leaking location and phone state information. Under phone state based information the two most leaked pieces of data were IMEI (International Mobile Equipment Identity) and android ID. It makes logical sense that information of this nature would be the most leaked kind of data to third parties because it enables them to build a portfolio of your activities across various applications. This kind of information is revealing in that it enables the third party in question to do a breakdown of your activities by device since IMEI and android ID are unique to every android device.

5. DISCUSSION

5.1 Project Continuation

Moving forward this introductory work could be built upon by including more detection features to Friend in the Middle and expanding the collection of apps beyond health apps. Adding more detection features would allow for a better picture of what kind of information is being leaked. With this a future project could center around building a catalogue of data leaks broken down by app category. This catalogue would convey knowledge describing what kind of sensitive information different classes of apps are more prone to leaking onto third parties. The other direction that would serve this project would be to expand the work beyond health apps and increase the collection of apps in the study. Based on the features that Friend in the Middle was assigning as data leaks it was determined that the majority of health apps don't spew information to third parities. An insightful continuation would be to expand into other types of apps (i.e fiance, gaming, social media, etc) and discover weather the leakage properties observed in health apps hold for these types of apps as well. This would enable the classification model to have more usability for a consumer or researcher consider how an arbitrary app may or may not leak information.

6. REFERENCES

[1] Maged N Kamel Boulos, Ann C Brewer, Chante Karimkhani, David B Buller, and Robert P Dellavalle. Mobile medical and health apps: state of the art, concerns, regulatory control and certification. *Online journal of public health informatics*, 5(3):229, 2014.

[2] Jan Lauren Boyles, Aaron Smith, and Mary Madden. Privacy and Data Management on Mobile Devices concerns about personal information. *Pew Research Center's Internet & American Life Project*, pages 1–19, 2012.

[3] Qi Alfred Chen, Zhiyun Qian, and Z Morley Mao. Peeking into Your App without Actually Seeing It : UI State Inference and Novel Android Attacks. *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, August 2014.*, (August):16, 2014.

[4] Yayun Chen and Hua Zhang. Detecting Sensitive Behavior on Android with Static Taint Analysis Based on Classification. (Icmmcce):3002–3006, 2015.

[5] M. J. Culnan and P. K. Armstrong. Information Privacy Concerns, Procedural Fairness, and Impersonal Trust: An Empirical Investigation. *Organization Science*, 10(1):104–115, 1999.

[6] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. TaintDroid. *Communications of the ACM*, 57(3):99–106, 2014.

[7] William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*, pages 235–245, 2009.

[8] Christian Fritz and Steven Arzt. Highly precise taint analysis for android applications. *Ec Spride, Tu ...*, 2013.

[9] Valerie Gay and Peter Leijdekkers. Bringing health and fitness data together for connected health care: Mobile apps as enablers of interoperability. *Journal of Medical Internet Research*, 17(11), 2015.

[10] Seungyeop Han, Jaeyeon Jung, and David Wetherall. A Study of Third-Party Tracking by Mobile Apps in the Wild. *University of Washington Technical Report*, 2011.

[11] Mobile Health. Mobile Health and Fitness Apps : What Are the Privacy Risks ? 3 . What are the risks to consumers of using mobile health and fitness applications ? pages 1–7, 2015.

[12] Bo Henderson and Prof Raquel Hill. Friend in the Middle. 2016.

[13] Balachander Krishnamurthy and Craig E. Wills. Privacy leakage in mobile online social networks. *Proceedings of the 3rd conference on Online social networks*, pages 4–4, 2010.

[14] Peter Leijdekkers and Valerie Gay. Mobile apps for chronic disease management: Lessons learned from myFitnessCompanion?? *Health and Technology*, 3(2):111–118, 2013.

[15] Elizabeth L Murnane, David Huffaker, and G Kossinets. Mobile Health Apps : Adoption , Adherence , and Abandonment. *Ubicomp/Iswc '15 Adjunct, September 7–11, 2015, Osaka, Japan*, pages 261–264, 2015.

[16] Aarathi Prasad, Jacob Sorber, Timothy Stablein, Denise Anthony, and David Kotz. Understanding Sharing Preferences and Behavior for mHealth Devices. *Workshop on Privacy in the Electronic Society*, pages 117–128, 2012.

[17] Predrag Radivojac and Martha White. B555 : Machine Learning Notes Table of Contents. 2015.

[18] Vuda Sreenivasa Rao and T Murali Krishna. A Design of Mobile Health for Android Applications. (JUNE 2014):20–29, 2015.

[19] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. ReCon: Revealing and Controlling Privacy Leaks in Mobile Network Traffic. pages 1–14, 2015.

[20] a Shabtai, Y Elovici, and L Rokach. *A survey of data leakage detection and prevention solutions*. 2012.

[21] Yuri Shapira, Bracha Shapira, and Asaf Shabtai. Content-based data leakage detection using extended fingerprinting. *arXiv preprint arXiv:1302.2028*, 2013.

[22] Yihang Song. PrivacyGuard : A VPN-Based Approach to Detect Privacy Leakages on Android Devices by. 2015.

[23] Yang Tang, Phillip Ames, Sravan Bhamidipati, Ashish Bijlani, Roxana Geambasu, and Nikhil Sarda. CleanOS: Limiting Mobile Data Exposure with Idle Eviction. *Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation*, pages 77–91, 2012.

[24] Jennifer M Urban, Chris Jay Hoofnagle, and Su Li. Mobile Phones and Privacy. *UC Berkeley Public Law Research Paper No. 2103405*, page 33, 2012.

[25] Dm West. Improving Health Care through Mobile Medical Devices and Sensors. *Brookings Institution Policy Report*, (October):1–13, 2013.

[26] Z Yang, S Zhong, and R N Wright. Privacy-Preserving Classification of Customer Data without Loss of Accuracy. *Proceedings of the 5th SIAM International Conference on Data Mining (SDM)*, pages 92–102, 2005.

[27] Jinyan Zang, Krysta Dummit, James Graves, Paul Lisker, and Latanya Sweeney. Who Knows What About Me? A Survey of Behind the Scenes Personal Data Sharing to Third Parties by Mobile Apps, 2015.