Windows 7 Information Leaking

<Students' names redacted>

Introduction

The inspiration for this project came from a news article that mentioned how some users were getting notices from their service providers that they were approaching the limit on their data usage for the month(1). What was special about this was that the users didn't feel like they did much data transfer on their phone at all and so they brought the issue up with Microsoft, who did not immediately know what the issue was. The data leak was eventually traced to Yahoo. This was discovered by a coder going by the name of "Rafael Rivera", who was a victim of the data leak.(1) He decided to take a look at all the data going to and from his phone, and took apart the IMAP packages belonging to these messages. He found that Yahoo was sending a lot more data than it needed, up to 25 times more. This was due to a synchronization problem between the Yahoo mail client and the Windows 7 Phone mail client, were it would constantly search for and download new updates and e-mails. A small number of users were affected when this communication between the two clients went on a lot more frequently than it needed to. A temporary fix suggested was change how frequently the Yahoo!Mail e-mail client downloaded new content and e-mail. Since this was a problem with Yahoo there were questions why this problem didn't exist on iPhones. The answer was that Windows phones had a higher threshold for the amount of messages they could accept per minute, and this problem only happened at those higher levels.

Data Access

Securing private data on the smart phone is partly dependent on how applications can access this data. The following section will describe how data is accessed and used on various smart phone platforms. These platforms include the Android, iPhone, and Windows 7.

The Android platform bases its application security on a permission based system. By default, applications have no permission to perform operations that can adversely impact the operating system, user , or other applications. At the time the application is installed the user is prompted with a list of permissions that this application requests. At this point the user can continue installation or stop it. The main permissions that an application needs for extract personal data on an Android phone include: Network Communication and Your personal information permissions. With these two permissions an application would be able to access your personal data and send it to a remote server.

Accessing personal data on the iPhone is slightly different and a bit more relaxed. iPhone applications ask for the user to allow it to use GPS data, via pop prompt during run-time, but it does not restrict the application from accessing contact data and various other types of data. The responsibility is mostly on the user to only download applications they trust.

Accessing user data on the Windows 7 platform was found to be the most restrictive. This restrictiveness is very convenient in terms of security, but with this added security, developer power is reduced. On the Windows 7 platform an application is provided with a black boxed function that they are allowed to use to pull user data. When data is being accessed via this function, it prompts the user to select what information to give the application. The application never has any direct contact with the data and must go through the user to retrieve any information on the phone. This provides very good protection of user data, but also restricts the power of applications.

Regardless of how the information is accessed, the main problem with securing private data is HOW it is used. Most applications use private data to enhance the user experience in a good way. Without this ability, many applications would lose much of its value to the user. Even the most restrictive protocol for allowing apps to access private data, cannot stop the abuse of the data that is released. The user does not know if the data is being sold to advertisers, used for malicious purposes, or stored on an unsecure database. In the end, the user will have to trust that this data is not being abused. The current solutions to this problem mainly rely on the policing of applications by the marketplace they are hosted on. Android and iPhone marketplaces have both had occurrences where they have removed applications they have found to be abusing private data. All three marketplaces require developers to sign up in order to upload applications to the marketplace. This system helps hold people accountable for what their applications do. A third party solution that is available on the Android platform is an application that is called TaintDroid.(2) This application was created by Intel Labs, Penn State, and Duke University and tries to determine whether and application abuses private data by monitoring HOW the data is being used. Using this application, researchers found that out of 30 popular Android applications that were studied, half of them were found to be selling user location data to advertisers.

Just how prevalent is data abuse? The numbers vary depending on the source. According to the

App Genome Project, started by mobile security firm Lookout, a significant amount of applications can access location and user data. The numbers are as follows: 29% of applications on Android and 33% on iPhone can access user location, 14% on iPhone and 8% on Android can access user contact data, and 47% of free Android apps have third party code. Third party code is defined as mobile ads and analytic tracking. In general, most applications do not abuse data, but the potential for abuse is very prevalent. The best way to secure and make sure your private data is not being abused is dealt with on a user-by-user basis. There are ways you can encrypt all of your data using third party applications like LookOut,(3)(4) but this severely limits the power of many applications. LookOut also provides malware/spyware detection and is available for all major mobile phone platforms. Overall, users need to watch what applications they download. Also, depending on how secure they need their phone to be, they can explore third party solutions like LookOut and TaintDroid.

The Windows Phone 7 has a very sophisticated set of security features. The first that I will talk about is what they call Isolated Storage. Isolated storage is a security feature that has been in place since Silverlight version 2. Isolated storage is a location given to each application to store its settings, files, or any other data. This is necessary because Microsoft does not allow developers access to any other memory locations on the phone. By only giving each application access to its own local storage, the operating file system is protected from manipulation by any entity besides the operating system itself. Since all I/O operations are contained in the operating system layer these too are protected. Along with any major security systems comes functionality issues and this is no different. Sometimes applications may need to talk to other or pass information back and forth. With isolated storage this is not easily achievable because as stated before, programs only have access to their own storage spaces. For applications to communicate with each other both applications must upload current versions of their storage spaces to the internet for retrieval by the other one. The shared data cannot be local to either application. Local copies can then be downloaded to each application's isolated storage space where it can then be used by the application. There is no imposed limit on the size usable by isolated storage so developers must be careful with how they handle this space.

Since applications can only access their own isolated storage, there has to be a workaround in any case where an application wants to access certain phone and operating system features not available in the application itself. It would be pretty redundant for developers to have to implement complimented features such as phone, email, etc. that are already a part of the phone. This is achieved through two API collections called launchers and choosers. Launchers and choosers are called upon using the various tasks that perform all different kinds of operations. There are launchers that allow the application to perform the following actions: Launch the email application, launch the marketplace, launch the marketplace detail task, launch the marketplace review task, launch the marketplace search task, launch the media player, make a phone call, launch the search task, launch the SMS text message task, and launch the web browser. Choosers are nearly identical to launchers. They only differ in that choosers return values back to the calling application when they finish while launchers do not. There are choosers that allow the application to perform the following actions: return an email address, return a phone number, launch the camera application, and return a picture. Developers can access launchers and choosers by Microsoft.Phone.Tasks in their code.

Even with all the other security measures in place there is one final security measure in place. Every application that is submitted into the marketplace must pass an inspection and adhere to a series of requirements given by Microsoft. The general procedure for getting an application live in the marketplace starts with obtaining a developers license. Then the app must be submitted to Microsoft for inspection. If it passes then it will go onto the marketplace but if not it is sent back for fixing and resubmission. Once submitted it is tested for reliability, effect on phone's performance, and permission checks before changing settings. It is also tested for viruses and malware and all exceptions must be handled. It also must not pass any personal information or data to any website, person, or application without first asking permission. On the technical side the program must start up and become responsive within 20 seconds and can never use more than 90 MB of data at any time. If all of these requirements are met the program will be made available for widespread usage.

Implementation

In creating our app one of the most important things was how we were going to get our information out of the phone after it was acquired. This was because it would be hard to show the information after it was saved on the phone; and even if this was eventually achieved it would be of little purpose if there was no way to get it out. The reasoning behind this was that a hacker would not want to physically go and get a hold of someone's phone to get the information out.

First we attempted to have the phone covertly e-mail us the data. The first version of this was just saving the data to the e-mail body and then

sending it to a pre-determined e–mail address. The problem with this was that applications on the phone have no direct access to the communication services, e-mail included. What you can do is decide the content, subject line, and "To" address of the e-mail. The way the phone sends e-mail is for the application to launch an "e-mail service" which runs on top of the application. Our application had to access to this service, and the user would then access their e-mail account and send an e-mail from there. To overcome this we tried to change the font of the message content to an invisible font. However while it was possible to have custom fonts on the Windows 7 Phone, this did not transfer to the e-mail launcher.

After this we decided to move in a new direction. Taking some time to research on the internet we decided that we would save all of the data into the isolated storage of the application. Once this was decided our efforts moved into retrieving and accessing this information. There are various methods online of varying difficulty and coverage. When we say coverage we mean the phone models. One of the programs which we used lets you access all of the phone's memory on the phone itself was not supported on our phone model. Next we found a program that made use of WCF communications to communicate between the phone and the PC. We also encountered problems with this as we could not set up communications between our PC and the phone. Eventually we were pointed in the direction of Windows Phone 7 Isolated Storage Explorer. This was a program that you installed on your desktop and could then access the phone through the usb cable. After some tweaking we eventually got it to work and this is what we went with. The program lets you access the isolate storage for the phone and you can then download individual files to your PC.

The application itself had a simple layout with just a few buttons. The user is asked to type in data to be saved to isolated storage and this data is then shown when they click the second button. When the user selects to save the data a second process is started in the background. This process is a Geo Location watcher that is used to access GPS data. Once the data is retrieved it is then saved to the isolated storage in a new line. When the user asks for the data back only the first line is shown. We later also added he functionality to also retrieved the device ID. The third button on the screen launches an emailchooser provided by Microsoft, which lets the user pick an e-mail that they have previously saved to their phone. After they choose an e-mail address we then save this data to isolated storage as well. After all this was done we use the Windows Phone 7 Isolated Storage Explorer to download the file to the desktop. Then we opened this file to retrieve all of the personal information we have saved.

When it came time to actually show the application we had various issues with our program hanging. After a lot of work we discovered it had to do with not getting a GPS signal inside of buildings. We were eventually able to make it work inside of the Caldwell 112 lab (not sure why it would work in here). We showed this to our grader Adam Champion and this was the end of our project.

REFERENCES

1. Yahoo! behind mysterious Windows Phone 7 data leak - http://www.geek.com/articles/mobile/yahoo-behind-mysterious-windows-phone-7-data-leak-2011021/
   - accessed on 02/22/2011

2. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones - http://appanalysis.org/tdroid10.pdf
   - accessed on 03/10/2011

3. LookOut Mobile Security - https://www.mylookout.com/
   - accessed on 03/10/2011

4. App Genome Project - http://blog.mylookout.com/2010/07/introducing-the-app-genome-project/
   - accessed on 03/10/2011