AN INVESTIGATIVE STUDY INTO THE PRIVACY OF IOS APPLICATIONS

MICHAEL SILVERBLATT
MENTOR: MING CHOW
TUFTS UNIVERSITY

1. Abstract

This paper looks to address the many privacy concerns that accompany the vast mobile application usage in todays world. Many users are often not even aware that an app can access their personal information, let alone share it with others and use it for its own benefit. However, despite the increase in scrutiny that apps must go through just to appear in the Apple App Store following the negative publicity surrounding the revelation that a popular app, Path, had been uploading users address books without permission to use for marketing purposes, many popular applications still post information about users to their servers with no immediately apparent purpose. Some of the apps are properly granted permission to use such information; others have found ways around Apples enforcement of its policies. In this paper I discuss the implications of application privacy violation, the current difficulties in determining leakage of private information from iOS devices, and introduce a Python script to aid in the monitoring of applications, called iosscan, which uses mitmproxy to decode SSL traffic and presents a summary to the user. The paper concludes with preliminary findings in traffic from certain popular applications and a note about the importance of remaining vigilant when using the web or any web-connected device.

2. Introduction

Mobile applications are prime targets for malicious applications for many reasons. They are widely used by people with every level of technical sophistication, which leaves for many unsuspecting victims. An exciting new game that may have slipped through the cracks of the testing process could go viral before the developer removes the code snippet that posts GPS locations of users. Mobile computing by nature comes with computational and network limitations, so users have grown accustomed to quirky behavior such as lagging or freezing, which allows time for malicious code to run before a user thinks to stop it.

3. To the Community

Privacy of mobile applications should be of great concern to anyone who uses a cellular device. If all of the information that is logged about you could be gathered in one place, one could likely track your every movement, down to every time you not only check your phone, but switch between applications, as well as every person or thing you ever come into contact with. It can be dangerous for so much information about an individual to be in the open; one is left vulnerable to anything from physical assault to identity theft or e-ransoms.

3.1. **Usage Logs/Analytics Frameworks.**

Mobile analytics platforms undoubtedly provide developers with useful information for the purposes of debugging and improving software, but the extent to which usage is

logged is too extreme to be necessary. Luckily, some of the platforms allow a user to opt-out of the logging frameworks by providing your iPhones Unique Device IDentifier (UDID), but most people would not know where to find that even if they somehow discovered their usage was being logged. In theory, these logs are anonymous and pose no immediate threat, but as I discuss later on, this may not be the case.

3.2. **Marketing Agencies.** Marketing frameworks work similarly to analytics logs, except they attempt to associate personal data to a device to target advertising content rather than debug an application. Apart from the obvious concerns about being publically linked to information gathered from things like web searches, we also run the risk of amassing a largely false body of information. There is often more than one person who regularly uses a device, yet all of their information is likely bundled together and tied to a single identity. Not only could a user not change this information if it is incorrect, they would have no way of knowing it was wrong in the first place. This can be damaging if it were ever to leak to future employers or other persons of interest.

3.3. **NSA/Government.** Protecting ones privacy from the government is of comparable concern to protecting it from corporations and malicious users. As recently leaked confidential documents have revealed, the government, and in particular the NSA, attempt to store and file away any information they can get their hands on. It is far from a bold assumption to think they could potentially access not only your marketing portfolios, but your usage records as well.

4. How We Stop Applications From Accessing Personal Data

4.1. **Privacy Policy.** Apple has a strict policy regarding user information that can be accessed within an application. Unlike Android applications, which request all permissions prior to installation, iOS applications request permissions at runtime. Since a request for permissions can come at any time, this can both be confusing to users and make it more difficult for Apple to deny unauthorized requests. Thus, Apple must rely on a combination of static and dynamic analysis to determine if an application is fit for sale in the App Store, and these analyses are not always accurate.

4.2. **Binary Analysis.** Tools for determining how secure an application is with your private data are far less advanced for mobile platforms than full-fledged desktops. Data can be accessed with a single function call. Developers have discovered clever ways around binary static analysis of their code, allowing them to execute functions that are forbidden. Analyzers like Veracodes AdiOS do a good job of detecting information leakage, but they are not perfect. As shown in a BlackHat presentation by Nicolas Seriot, the following code should not be allowed by an iOS application, but would pass many static analyzers.

```
- (NSString *)stringMinus1:(NSString
    *)s {
        NSMutableString *s2 =
            [NSMutableString string];
        for(int i = 0; i < [s length];
            i++) {
                unichar c = [s
                    characterAtIndex:i];
                [s2 appendFormat:@"%C",
                    c-1];
        }
```

```
        return s2;
}
- (void)viewDidAppear:(BOOL)animated {
 NSString *pathPlus1 =
 @"0wbs0npcjmf0Mjcsbsz0Qsfgfsfodft0dpn
/bqqmf/bddpvoutfuujoht/qmjtu";
 // @"/var/mobile/Library/Preferences/
// com.apple.accountsettings.plist"
 NSString *path = [self
     stringMinus1:pathPlus1];
 NSDictionary *d = [NSDictionary
     dictionaryWithContentsOfFile:path];
 // ...
}
```

4.3. **Dynamic Analysis.** Dynamic analysis presents a whole different set of issues. In certain cases, it is obvious that an HTTP request is sending data it should not have, such as a request to a URL along the lines of *store_all_contacts*. A request from a game to *set_highscore*, on the other hand, would appear to be benign at first glance. However, further inspection may reveal that the payload not only includes the score from the last game played, but a copy of the users entire address book. Other than by looking, it is nearly impossible to detect that such a request would be malicious. To be efficient while dynamically analyzing an application, it is necessary to find a reliable heuristic for which requests are worthwhile to track further.

4.4. **How can we do better?** A paper from the University of California, San Diego titled ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing introduces a novel approach to assessing privacy risks. It details a method of crowdsourcing application data from a large user base, and using said data to recommend privacy settings for an array of different software. Recognizing both that trustworthy and malicious requests can often be indistinguishable without enough extra information, and that access to personal information is not always immediately apparent from source code, Agarwal and Hall overcome these obstacles by attempting to determine which applications should have access to certain information, rather than determining which actually do. Unfortunately, their application, ProtectMyPrivacy, was only made available to users with jailbroken iOS devices, as it monitors system processes other than its own. This limits its use considerably, since the act of jailbreaking a device invites a litany of other security issues, and thus is not the focus of this paper.

5. iosscan

5.1. **A New Way To Scan iOS Traffic.** I looked to create a tool to aid an end user in monitoring the behavior of their iOS devices and the applications that run on them. Building on top of Aldo Cortesis mitmproxy project, using mitmdump in particular, I wrote a script in Python which gives context to the traffic in and out of an iOS device. Mitmdump and mitmproxy are man-in-the-middle proxies for decoding SSL and reading HTTP/S traffic. Mitmdump runs similarly to the more common tcpdump, while mitmproxy runs with a nice interactive interface. Rather than recreate the wheel, I looked to build something that would enhance the use of an existing tool.

5.2. **Usage and Functionality.** Using mitmdump or mitmproxy alone and manually inspecting the output, request by request, is an inefficient and nearly useless practice. My script, iosscan, allows a user to see a live count of personal information being shared and a log of the various apis being accessed from the device, as well as filter traffic by a

specific host api to see who is actually breaking (or not breaking) privacy permissions. The script reads output from mitmdump line by line, summarizing vital parts of each request by searching for various keywords in the URLs and payloads of the requests. It is intended to help a user determine whether or not they should be concerned about the applications they use on a daily basis. I still recommend use of mitmproxy to directly inspect any traffic you are concerned about, as it does a very effective job of detailing requests.

6. Application Data Uncovered with iosscan

6.1. **Facebook.** The Facebook iOS had some of the most obviously suspicious activity throughout my testing. Whenever in the background, the application would sporadically send incredibly detailed device information to the /mobilelogs and similar actions of their api, including carrier information and even battery state. Having noticed that it logged data constantly even when not in use, I immediately uninstalled it.

6.2. **Snapchat.** Analysis of Snapchat revealed multiple concerns, the most outstanding of which being that it posts to a logging framework every time the application is opened or closed. These logs are not anonymized by a device identifier, but actually include the username associated with the device. On top of tracking exactly how and when each and everyone uses the application, many of the features are incredibly unsecured. For example, to prevent users from taking screenshots of the images only intended to last a few seconds, upon viewing a snap, a post request is sent with the number of screenshots as a parameter. Using mitmproxy, it would be a trivial task to intercept this request and modify the parameter to remain 0.

6.3. **foursquare.** The foursquare app was difficult to analyze, since it is a social app based on location and thus much of the personal information it transmits is authorized. However, it does submit a post request upon being opened with a ll parameter which appeared to contain a set of coordinates. Sure enough, the coordinates mapped to my GPS location. It appears the service logs your location whenever you use it, even if you are not searching for nearby friends/attractions or attempting to check-in and post your location.

6.4. **Chase.** Perhaps the most concerning privacy concern comes from Chases mobile banking application. Its difficult to determine whether or not GPS location is being transmitted with a login attempt, but the flaw with their credential management system is painfully obvious. The username/-password combination is sent unencrypted in the HTTP request URL. The developers apparently are confident in the security of SSL and HTTPS, but the ease with which I was able to see my own banking password in plain-text was at the very least disconcerting. An SSL proxy is the only think necessary to sniff Chase passwords, which requires only that a user install a certain SSL certificate (admittedly a difficult thing to do without a user noticing).

7. Conclusion

Through my study of iOS network traffic, I have become increasingly convinced that the aforementioned privacy concerns are far from eradicated, but remain on a downward trajectory. That said, much of the traffic was encrypted on top of the SSL encoding,

so there is a lot going on behind that requires more expertise and privilege to fully inspect. Surprising to me, nearly every application uses some form of tracking or logging framework. It is important that as mobile devices become more and more ubiquitous the public remains knowledgable about the risks that go along with them. I hope that this paper gives a suitable overview of the concerns surrounding iOS application data privacy and the procedures in place to mitigate the harmful consequences of mobile computing.

References

[1] http://www.wired.com/gadgetlab/2012/02/path-social-media-app-uploads-ios-address-books-to-its-servers/

[2] http://mesl.ucsd.edu/yuvraj/docs/Agarwal_MobiSys2013_ProtectMyPrivacy.pdf

[3] http://www.technologyreview.com/news/516416/study-shows-many-iphone-apps-defy-apples-privacy-advice/

[4] http://www.blackhat.com/presentations/bh-dc-10/Seriot_Nicolas/BlackHat-DC-2010-Seriot-iPhone-Privacy-slides.pdf

[5] http://seclab.cs.ucsb.edu/media/uploads/papers/egelendss11.pdf

[6] http://phd.gccis.rit.edu/weile/docs/le_mobs13-2.pdf

[7] http://media.blackhat.com/bh-eu-11/Nitesh_Dhanjani/BlackHat_EU_2011_Dhanjani_Attacks_Against_Apples_iOS-WP.pdf