Extended Abstract: Are You Sure They Are the Same? Identifying Differences Between iOS and Android Implementations

Daniel Domínguez-Álvarez[†‡], Alessandra Gorla[†], Juan Caballero[†], and Roberto Giacobazzi[‡†]
[†]*IMDEA Software Institute* [‡]*University of Verona*
Madrid, Spain *Verona, Italy*

***Abstract*—Most mobile applications are available for multiple platforms, most often Android and iOS since they jointly cover nearly the entire market. While the functionality of the Android and iOS implementations of an application may be expected to be the same, in reality they may differ significantly due to misalignments during the application development process.**

This extended abstract presents an ongoing project whose goal is to identify the differences, in terms of functionality and security offered to the user, of the Android and iOS implementations of a mobile application. Our current approach focuses on differences in the network traffic. Our preliminary results show that some security functionality may be implemented in only one of the two platforms. In a extreme case, one application encrypts its network traffic in Android, but not in iOS. Other applications only implement TLS pinning on Android and may only check it in some parts of the application.

***Index Terms*—Mobile security, privacy, iOS, Android**

Tipo de contribución: *Investigación en desarrollo*

I. Introduction

Most existing research on mobile security and privacy focuses on the Android platform. One reason for this is that Android has the largest share of the mobile market (over 80% in 2018 [1]). Another reason is that the Android ecosystem is more open than those of other platforms like iOS. With Android, researchers can easily download apps from the official store, and there exist a large number of openly available analysis tools that researchers can easily use out of the box and build upon.

Despite its smaller market share, most publishers offer their apps for the iOS platform as well. Due to the development effort required for each platform, there may be separate development teams for different implementations of the same app, e.g., one team for Android and another for iOS. This may introduce a misalignment in the app development process, leading to different features offered by the same app in two different platforms. This problem can be addressed using app development frameworks that allow mobile app developers to produce releases compatible with multiple platforms including Android and iOS [2]. Using these frameworks would entail that the functionality of the app across different platforms would be aligned. However, because of the many disadvantages that these frameworks have (e.g. poor performance of the produced app and poor look and feel of the app interface), most developers prefer to develop native apps.

In this extended abstract we present our ongoing research, which aims to identify the main security and privacy implications due to having separate teams potentially implementing the same mobile app for different platforms. We focus on the Android and iOS platforms because they jointly cover nearly the entire market.

Our approach is based on differential testing. We first collect the latest versions of an app from the official Google Play and Apple App stores. Then, we compare both implementations in order to identify their security-related differences. Currently, we focus on comparing the network traffic generated by both implementations. By monitoring the network traffic we find applicaitons that implement security functionality only in one platform, but not in the other. In a extreme case, an application only encrypts its network traffic in Android, but not in iOS. Other applications implement TLS pinning only in Android and may only check it in some parts of the application.

II. Dataset and Methodology

We have collected a small dataset of 15 apps from 3 categories that require Internet connectivity: News, Entertainment and Shopping. From each of these categories, we selected the top free apps for the Spanish market in the Apple App store. Then, we searched for those apps in the Google Play store and kept only apps that existed and had high ranking in both stores. We downloaded all apps from both stores on March 28th, 2019. Table I summarizes the 15 apps in our dataset. When needed, we have created accounts to use the apps.

Table I
The list of apps used in our preliminary study.

News	Entertainment	Shopping
El Mundo	Netflix	Amazon
El País	HBO	Wallapop
Reddit	Amazon Prime Video	Milanuncios
Twitter	MiTele	Aliexpress
Activo2	Juasapp	Wish

With this dataset we proceed to analyze the traffic that both implementations generate. For intercepting the traffic, we setup a HTTP and HTTPS proxy that our test devices use to access the Internet. In order for the HTTPS proxy to work, we create a self-signed certificate that we install as a trusted certificate in the devices. The proxy generates HTTPS certificates on the fly using our self-signed certificate as a root CA. With this setup, we can intercept the traffic of any app that trusts the device's keystore. If an app implements any kind of public key or certificate check (i.e., TLS pinning), it would be protected against our traffic analysis. While this limits the applications we can analyze, it also enables us to identify applications that implement TLS pinning in one platform, but not in the other.

We generate traffic by running an app three times in each platform, for a total of 90 runs. In each run, we leave the application idle for the first minute. After the first minute, we interact with the application for up to two additional minutes. During this time we try to trigger network traffic by scrolling through listings, trying to request specific objects from the server and by logging into the app. For consistency, we execute the same actions in different runs of the same app.

III. Preliminary Results

Figure 1 shows the results of our analysis. Each figure plots the number of requests seen per second by the proxy while each application was being monitored. We installed and ran each application individually. To remove the noise caused by the underlying operating system, we filtered the traffic by user-agent and domain name, keeping only requests generated by the app under analysis.

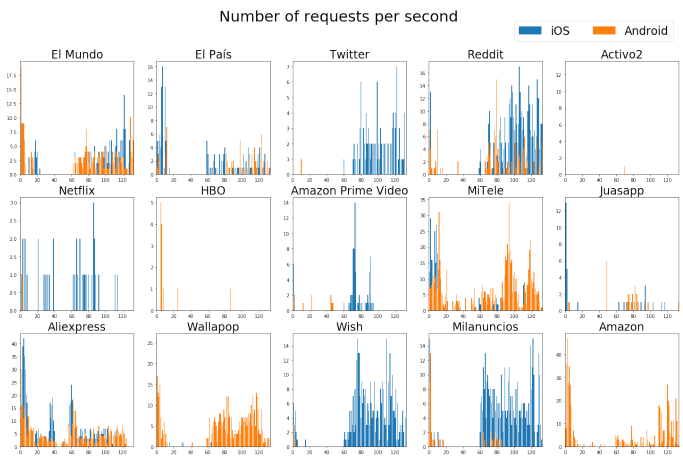

Figure 1. Network traffic generated by each app. On the Y axis we plot the number of requests, and on the X axis time in seconds.

Figure 1 shows that Activo2 and HBO have no traffic in both platforms. The only traffic for HBO is the request of the app to install GooglePlayServices. This indicates that the developers have implemented TLS pinning in both platforms. The checks run as soon as the application starts, preventing the app to work with our proxy. For several applications we observe traffic in one platform, but no traffic in the other one, indicating differences in TLS pinning support across platforms. Some apps lack TLS pinning in iOS, but are protected in Android. Some examples are Twitter, Netflix, Wish and Milanuncios. Other apps like Amazon or Wallapop show the opposite behavior, being vulnerable in Android and protected in iOS.

Some apps showed a very fine grained TLS implementation, probably coupled to a micro-services architecture in the back-end. These apps seem to implement a default TLS policy in non-critical parts of the app, while the more critical parts, such as logins, are secured. The apps that showed this behavior tend to do the same in both platforms. This is to be expected, since developers have put a huge effort in securing each component individually.

An extreme case of missing security functionality in one implementation is the Juasapp app, an application for making prank calls. The app loads information from the server using HTTPS in the Android implementation, but uses (unencrypted) HTTP for the same purpose in the iOS implementation.

IV. Work Plan

We plan to continue our study along several lines. First and foremost we want to have a statistically significant empirical study with several thousands apps. This means automating the execution of each app, which may not be trivial since the same app on two different platforms may differ significantly in the UI and in the functionalities offered. Moreover, existing UI test input generation tools still have significant limitations [3].

We plan to expand our analysis and look for more TLS bugs, similarly to [4], [5], [6]. However we plan to focus on the differences between the iOS and Android platforms.

So far we did not analyze differences in protocols and in the data sent to the server in each platform, but we plan to do it in the near future. There are several challenges that we foresee for this analysis. First of all we would need to properly align the network traffic generated in different executions, and to identify security-relevant data in the network traffic such as usernames, passwords, location coordinates, and phone numbers.

Our longer term plan is to further look into the app implementations with both static and dynamic analyses. This would allow us to have a better understanding of the security checks that are in place and that may be missing in one implemenation. This study is quite challenging, as it requires separate infrastructures for the Android and iOS devices.

References

[1] "Mobile os market share 2018 — statista," https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, (Accessed on 04/03/2019).

[2] I. Dalmasso, S. K. Datta, C. Bonnet, and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools," in *IWCMC 2013: Proc. of the 9th IEEE Intl. Wireless Communications and Mobile Computing Conf.*, 2013.

[3] S. R. Choudhary, A. Gorla, and A. Orso, "Automated test input generation for android: Are we there yet?" in *ASE 2015: Proc. of the 30th Annual Intl. Conf. on Automated Software Engineering*. IEEE Computer Society, 2015, pp. 429–440.

[4] L. Onwuzurike and E. De Cristofaro, "Danger is my middle name: experimenting with ssl vulnerabilities in android apps," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2015, pp. 1–6.

[5] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith, "Why eve and mallory love android: An analysis of android ssl (in) security," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 50–61.

[6] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan, "Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps," in *In Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14*, 2014.