10.53477/2668-2028-21-35

INVESTIGATING PRIVACY LEAKS IN IOS APPLICATIONS

Adriana-Meda UDROIU, PhD
National Institute for Research and Development in Informatics
meda.udroiu@rotld.ro

Abstract: *The concern for protecting personal data has been raised many years ago and has been kept on sight ever since. People have to trust their devices and the applications they use so smartphone company survive on the market. Apple treats the security side very seriously and tries to reduce any malicious behaviour as early as possible. This is the main reason they introduced the vetting process to filter out the applications that seem suspicious and only publish legit applications on the App Store. Although the theory looks just fine, there have been multiple applications removed from the App Store because of malicious behaviours that couldn't be found during the vetting process. In this paper we focus on finding personal data leaks through already published applications in the App Store.*

Keywords*: data protection; privacy; Apple iOS.*

Introduction

One of the most important parts of security is related to data privacy. Data privacy, also called information privacy, is the aspect of information technology (IT) that deals with the ability an organization or individual has to determine what data in a computer system can be shared with third parties[1]. Most of the countries of the world have a strict regulation regarding data privacy about how, when and why information should be shared between parties. For example, in the United States, the Children's Online Privacy Protection Act (COPPA)[2] gives parents control over what information websites or mobile applications can collect from their children. Similar to this, is the General Data Protection Regulation (GDPR)[3] meant to harmonize data privacy laws across Europe.

Any application installed on an iOS device communicates with external servers and exchanges information with them for different reasons. The user is always asked if he allows the application to share certain private information with the server (e.g. for statistical purposes), but only for that information.

Private Data Classification

Before finding a right approach to follow our case, we had to describe what information about someone is considered sensible. We define what data is private according to multiple data privacy regulations from around the world. Outlining which data is considered private and which is not helps us filter the data being sent from the mobile phone to external entities.

Part of our study, we chose 10 worldwide regulations from 4 different continents and created a list of identifiers that represent private information. These can be split into multiple type depending on their nature. For example any online identifier such as location information, a device serial number, email, IP address, biometric or username are marked as personal information by multiple regulations such as GDPR[4], COPPA[5], DPA[6], RDPL[7], LGPD[8], PDPA[9]

[1] https://searchcio.techtarget.com/definition/data-privacy-information-privacy
[2] https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-onlineprivacy-protection-rule
[3] https://gdpr-info.eu
[4] https://gdpr-info.eu
[5]https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-onlineprivacy-protection-rule
[6] http://www.legislation.gov.uk/ukpga/2018/12/contents/enacted
[7] https://www.buzescu.com/romanian-data-protection-laws/
[8] https://www.cookiebot.com/en/lgpd/
[9] https://www.dataprotectionreport.com/2020/02/thailand-personal-data-protection-law/

or LRK[10]. Other pointed out identifiers are ones name and surname, date and place of birth, physical address, gender or eye color. These are generally used to legally identify a person and are considered personal information by regulations like GDPR and COPPA. Last but not least, health and culture factors, beliefs, education or possessions are also mentioned by some regulations in the personal identification information section.

Lumen[11], the Android app that analyzes mobile traffic and helps identify privacy leakes inflicted by Android apps and the organizations collecting this information[11] defines a list of Personal Identification Information they had in mind when designing the application. Part of their list are the email, phone number, geolocation, router MAC address and the router SSID.

Table 1. Most Common PII According to Regulations&Lumen

PII	Regulations
biometric data	GDPR, DPA, RDPL, LGPD, PDPA, LRK
health information	GDPR, DPA, RDPL, LGPD, PDPA
beliefs (of any type)	DPA, RDPL, LGPD, PDPA
any online identifier	GDPR, COPPA, RFLP
location	GDPR, COPPA, Lumen
device identification numbers	COPPA, Lumen
email	COPPA, Lumen

Breaking Regulations

Online data manipulation is very often encountered. Personal information can be easily collected and used afterwards for multiple purposes such as contributing to all kinds of statistics or creating custom recommendation lists[12].

This behaviour is limited by all the data privacy regulations around the world. According to most of them, it is forbidden to record, store or use pieces of private information without ones consent.

Although regulations are rather strict, some actions are not prohibited. For example, if one grants location access to an application, this has the right to send the location back to the server, use it to optimize features for the user based on its location, but it would not be OK if the location is sent to other external entities without specifically having the users consent.

The model of application

Mobile phone applications installed on Apple devices are usually third party applications that come from external developers. Because of this, all applications have to pass the Apple vetting process before being publicly released. We do not know much about what the vetting process does in order to check whether the application meets their standards or not, so we do not know how data privacy is tackled by Apple.

We want to come up with a solution that can guarantee that an application installed on an iOS device does not leak any personal information. We want to provide an answer to the research question *How strong is the Apple iOS security model in preventing privacy leaks by potentially malicious 3rd party apps?* By creating an automatic dynamic analyzer that will be

[10] http://www.ilo.org/dyn/natlex/natlex4.detail?plang=enpisn=96710pcountry=KAZpcount=7https://haystack. mobi

[11] I. Reyes, P. Wiesekera, A. Razaghpanah, J. Reardon, N. Vallina-Rodriguez, and S. Egelman, *Won't somebody think of the children? examining coppa compliance at scale*, 2018; I. Reyes, P. Wiesekera, A. Razaghpanah, J. Reardon, N. Vallina-Rodriguez, S. Egelman, and C. Kreibich, *Is our children's apps learning?" automatically detecting coppa violations*, 2017.

[12] https://www.wired.com/story/wired-guide-personal-data-collection/:text=It%20matters%20not%20just% 20what,or

able to detect privacy data leaks made by any iOS application and provide an answer to the question above. It is important to know if the Apple vetting process is as strong as it promises to be so that any security issue found can be solved as soon as possible.

Workflow

We propose a simple workflow described in Figure 1.

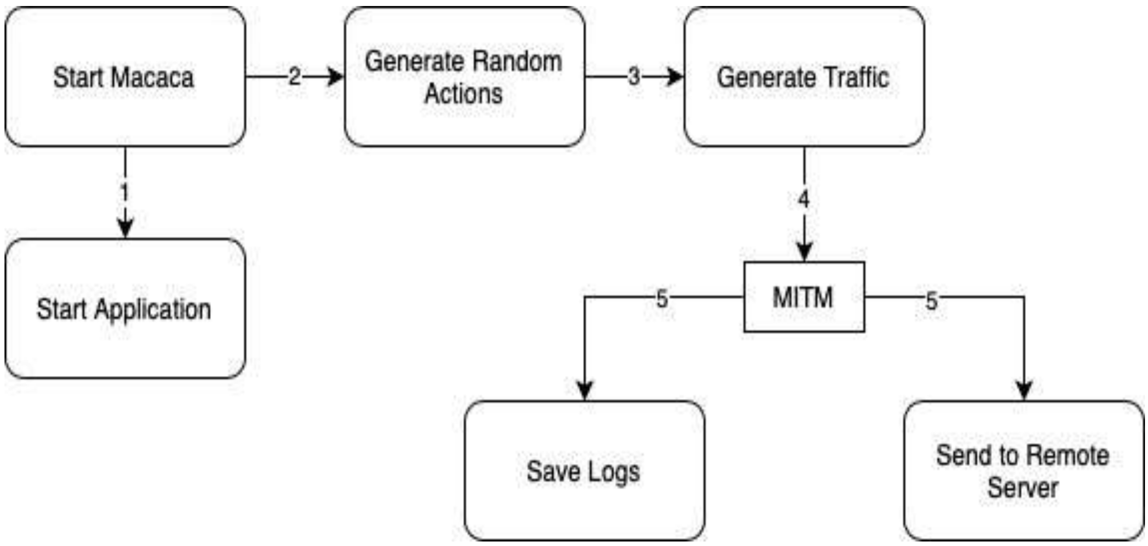

Figure 1. Basic Workflow

There are 2 main components part of the whole story that cannot be absent: a jailbroken iOS mobile device connected to the Internet and an application installed on it that might require Internet access. In most cases, the client application usually communicates to its server via the Internet. This traffic going back and forth is what interests us most. If an application leaks data, this means the data is sent to an external entity. We discuss in Section 3 more about who that external entity can be and why this is rather important.

After starting the application, we need to generate traffic as if a regular user would. We also have to have another component that can intercept and log the outgoing traffic. All the previous components described up to now work along.

Another separate component is responsible to analyze the traffic that we gather. As an output from this component we get a set of applications that did exfiltrate some information to external servers. Because each application might be allowed to do different actions (e.g. one might be able to access the users real time location, but another is not allowed to do that), we define these in a separate zone. By combining the separate zone with the information gather after analyzing the traffic we will be able to say if an application leaks private data it shouldn't or not.

Design

The workflow described in section 2 implies the fact that we gather a set of logs which have to be analyzed. We consider that defining a formal model of our problem eases out this process.

In order to define this model, we need to describe the 3 main components we have: user data, the application, the remote server. We split data into 2 separate categories: the user data that is defined within the system and the data that gets exfiltrated. Even though the last mentioned is included in the first one, we treat them as separate entities in our definition.

Before defining the premises, we define some the basic functionalities in table 2.

Table 2. Basic functions for the formal definition

Name	**Parameters**	**Functionality**	
A	string: x	returns true if	**x** is an application
D	string: x	returns true if	**x** is an personal data
R	string: x	returns true if	**x** is a remote server
sensitive	D: x	returns true if	**x** is sensitive data
owns	A: x, R: y	returns true if	**y** is a remote server associated with **x**
has access	A: x, D: y	returns true if application **x** has access to data **y**	
sent	D: x, R: y	returns true if data **x** was sent to remote **y**	
connects	D: x, R: y	returns true if application **x** connects to remote server **y**	

According to worldwide regulations the data defined within a system can be sensitive or not and we define a premise on that as below.

$$\exists y D(x) \land sensitive(y)$$

We describe a similar premise regarding remote servers. A remote sever can be authorized or not. A server is not authorized if is not owned by the developer of the application communicating with it.

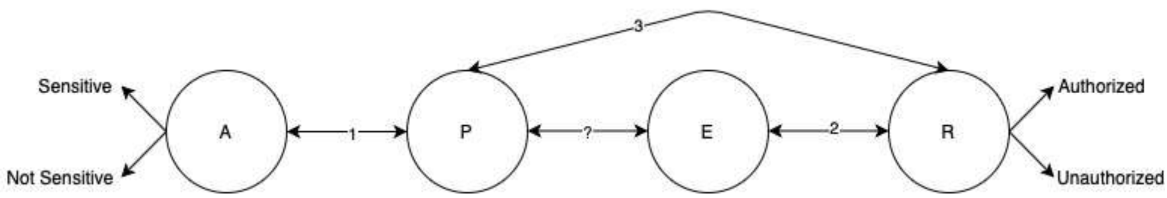

Figure 2. Information Mapping

$$\exists x A(x) \land \exists y R(y) \land owns(x,y) \Rightarrow authorized(y)$$

In order to define a formal model, we first have to define what types of supporting evidence we have. These are actually mappings of information between 2 of the 4 components described in Figure 2.

The first mapping refers to the link between an application and the data it has access to. We define a set of supporting evidence as a mapping between an application and a piece of information it has access to. For example, if there is an application x and a piece of data y we can say that application x has or doesn't have access to data y, we define the following premise:

$$\exists x A(x) \land \exists y D(y) \land has\ access(x,y)$$

The second mapping refers to the relation between the exfiltrated data and the remote server. We define data as exfiltrated if we the application sends it to any type of server.

$$\exists y D(Y) \land \forall z R(z) \land sent(y,z) \Rightarrow exfiltrated(y)$$

The last mapping points out the connection between the application and the remote server. We define the following premise for this connection.

$$\exists x A(x) \land \exists y R(y) \land connects(x,y)$$

Based on the premises defined above we define the following inferences.

$$\begin{aligned}&\exists x A(x) \land \exists y D(y) \land \exists z R(z) \land \\ &sensitive(y) \land has\_access(x,y) \land exfiltrated(y) \land \neg authorized(z) \\ &\Rightarrow \neg valid()\end{aligned}$$

$$\begin{aligned}&\exists x A(x) \land \forall y D(y) \land \forall z R(z) \land \\ &sensitive(y) \land \neg has\_access(x,y) \land exfiltrated(y) \\ &\Rightarrow \neg valid()\end{aligned}$$

If at least one of the inferences states true then we have a mathematical proof that there is at least one application installed on the device that leaks private data.

Implementation

To follow up the proposed design, we need a method gather logs and other information to properly define the premises. Besides this, we need a way to generate traffic in the first place. We describe both parts below.

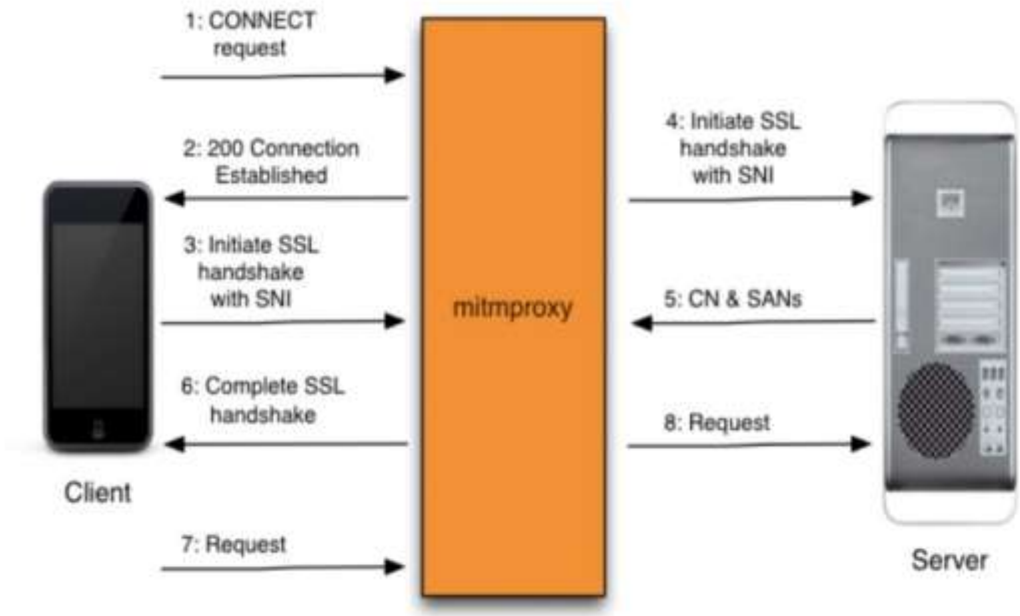

Figure 3. How mitmproxy works

Creating the Premises

In order to define the premises linked to the first mapping in Figure 2 we have to find out what data can an application access. This step is custom for each application used. An iOS user has the opportunity to grant or deny access to certain data and this is separately defined for each application installed on the device.

When gathering data about the connection between the exfiltrated data and the remote server, we use mitmproxy[1]. Mitmproxy is an HTTPS that can be used to intercept, inspect, modify and replay web traffic such as HTTP/1, HTTP/2, WebSockets, or any other SSL/TLS-protected protocols. Mitmproxy has multiple modes of operation, but at a basic level it just acts as an intermediate between the device and the Internet that is able to intercept and read requests coming from both sides as shown in figure 3.

Unfortunately, at this point we are unable to tell which application generated a given request. Without knowing the application, we cannot satisfy the model described before. In order to solve this problem, we describe the third mapping between the application and the remote server. The request captured by mitmproxy comes with a timestamp which we keep in mind. The lsof tool[13] can provide information regarding the active connections a mobile device has. The advantage here is that connections are mapped to a specific process which mean we know the application that has that active connection. Lsof can also capture the exact moment when the connection is established or closed. In this case, if we have a request made at a certain point in time that has a certain destination, we can map it to one one of the (established, stopped) intervals associated with the same end point. We parse the lsof output to retrieve the application name and the open connections it has. We also track only connection through sockets at the moment. We ignore systems daemons.

[13] https://linux.die.net/man/8/lsof

```
$ lsof -n -i -p ^1 -c ^wifid -c ^identitys -c ^lockdownd -c ^mDNSRespo -c ^sshd | tail -n +2
       | tr - " " | tr \> " " | tr -s " " | cut -d" " -f1,10 | cut -d":" -f1
(..)
TikTok 2.16.187.65
TikTok 2.16.187.65
(..)
```

Automating Traffic Generation

The Android Monkey is a program that runs on your emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. You can use the Monkey to stress-test applications that you are developing, in a random yet repeatable manner.[14]

Similar to the Android version, Costin Carabas, a PhD student from University Politehnica of Bucharest, found and setup a similar application, Macaca, that is doing the same job but for iOS devices.[15] We use this to generate all our user events instead of actually manually interacting with the device since our final goal is to have all these steps automated.

Preliminary Results and Current Status

We created an online profile as one would normally do. We added as much data to that account as possible and we linked it to as many applications as possible. We consider that it is more likely to exfiltrate any type of data when the user is logged in with an account. A person linked to an account is much more valuable than a person using an application occasionally.

Using mitmproxy[1] we captured a starting set of logs from which we extracted some information regarding the remote hostname and IP, the HTTP method of the request, the response code and length, but as mentioned in previous sections, these requests cannot guarantee the application that generated the request. Initial information we collected are listed in table 3.

Table 3. Preliminary Results

Application	**Host**	**IP address**	**Method**
TikTok	graph.facebook.com	31.13.84.8	GET
TikTok	dm16.musical.ly	92.123.103.9	GET
TikTok	play.googleapis.com	216.58.207.74	POST
TikTok	api2-16-h2-useast2a.musical.ly	92.123.102.200	POST
Find the Differences	wd.adcolony.com	34.224.46.26	POST
Find the Differences	firebaselogging-pa.googleapis.com	216.58.206.10	POST
Find the Differences	rv-gateway.supersonicads.com	54.230.159.20	GET
Find the Differences	api.gameanalytics.com	3.218.26.150	POST

To bring up to a full number, we started gathering information using lsof. We currently setup a script that runs every 2 seconds on our device and prints out a mapping between the applications running at that point and their active connections. We use launchd[16] to setup this script to run continuously and most important to keep it alive. We use the plist in the listing below.

[14] https://developer.android.com/studio/test/monkey
[15] https://macacajs.github.io/guide/cross-platform
[16] https://www.launchd.info

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
    PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>com.run.lsof</string>
        <key>ProgramArguments</key>
        <array>
            <string>/bin/bash</string>
            <string>/var/root/run\_lsof.sh</string>
        </array>
        <key>StandardOutPath</key>
        <string>/var/root/file_out.log</string>
        <key>StandardErrPath</key>
        <string>/var/root/file_err.log</string>
        <key>RunAtLoad</key>
        <true/>
        <key>KeepAlive</key>
        <true/>
        <key>StartInterval</key>
        <integer>2</integer>
        <key>KeepAlive</key>
        <true/>
</dict>
</plist>
```

Conclusion

At this moment we have already started working on the implementation of our model as mentioned in the previous section. We started working on gathering data regarding the communication between an application and a remote server, but we need to add more to it.

One thing would be to find an appropriate database or storage model to save all the connections an application makes. Afterwards we will define the best and most efficient way to store this much data. Besides storage, we need to find a fast and formal querying language in which we can define all of the premises we find throughout our research.

BIBLIOGRAPHY

1. *Mitmproxy*
2. REYES I., WIESEKERA P., RAZAGHPANAH A., REARDON J., VALLINA-RODRIGUEZ N., and EGELMAN S., *Won't somebody think of the children? examining coppa compliance at scale*, 2018.
3. REYES I., WIESEKERA P., RAZAGHPANAH A., REARDON J., VALLINA-RODRIGUEZ N., EGELMAN S., and KREIBICH C., *Is our children's apps learning?" automatically detecting coppa violations*, 2017.