Analysis of Android In-App Advertisement Kits

Karine de Pontevès & Axelle Apvrille
Fortinet, FortiGuard Labs EMEA
120, rue Albert Caquot
06410 Biot Sophia Antipolis, France
{kdeponteves,aapvrille}@fortinet.com

Abstract

Android captured 70% of smartphone shipments in the December quarter of 2012. With this explosion, Android has become the world's biggest magnet for smartphone applications, and mobile malware.

Individuals and organizations who develop legitimate applications, benefit financially either by selling them, or by embedding advertisement kits. Building free, ad-supported apps helps developers side-step the hassle of the Google Checkout flow, hence becoming the most popular form of monetization.

In this paper, we focus on the security risks and inefficiencies posed by ad-kits. And more particularly those embedded into malware. To this end, we study the Android platform, 120,000 malware samples and closely reverse-engineer 60 different ad-kits.

Our results show how most ad-kits collect a lot more private information than expected, and the disturbing extensiveness of details on it. We reveal the numerous fields collected, how they retrieve user account emails and even perform silent updates without notifying users. We discover recent ad-kits still continue to send geographical location in clear text. Finally, we demonstrate how users can be tracked by an ad provider across applications, and by a network sniffer across ad providers.

1 Introduction

Smartphones have exploded in popularity in recent years. The last quarter of 2012 has witnessed Android taking over 70% [1] of smartphone shipments, with some 1.5 million Android activations each day [2]. With the explosion of smartphones, application markets have also boomed. Google claims over 850,000 third-party programs for its platform [3], and although these are meant to be legitimate apps, Android's popularity has also attracted greedy malware authors.

Application (app) developers, whether individuals, organizations or malware authors, can either directly sell their apps on the official Google Play Store, or publish free apps and monetize by embedding advertisement libraries (ad-kits). Publishing free ad-supported apps is the dominant monetization form, saving developers from going through the hassle of

the Google Checkout flow, and asserting the mobile advertising industry's place in this market.

Success for the ad industry is linked to the success of user profiling: the more accurate the user profile, the more profitable the advertisement. A successful ad campaign requires access to personal information.

The online advertising model is comprised of 4 actors, see Figure 1:

- *Advertisers*, who pay *ad-providers* to disseminate their advertisements
- *Ad-providers*, who in turn, compensate *publishers* (the developers) to display their ads.
- *Publishers*, who display ads within their apps.
- *Users*, who are the targeted recipients of the advertisements.

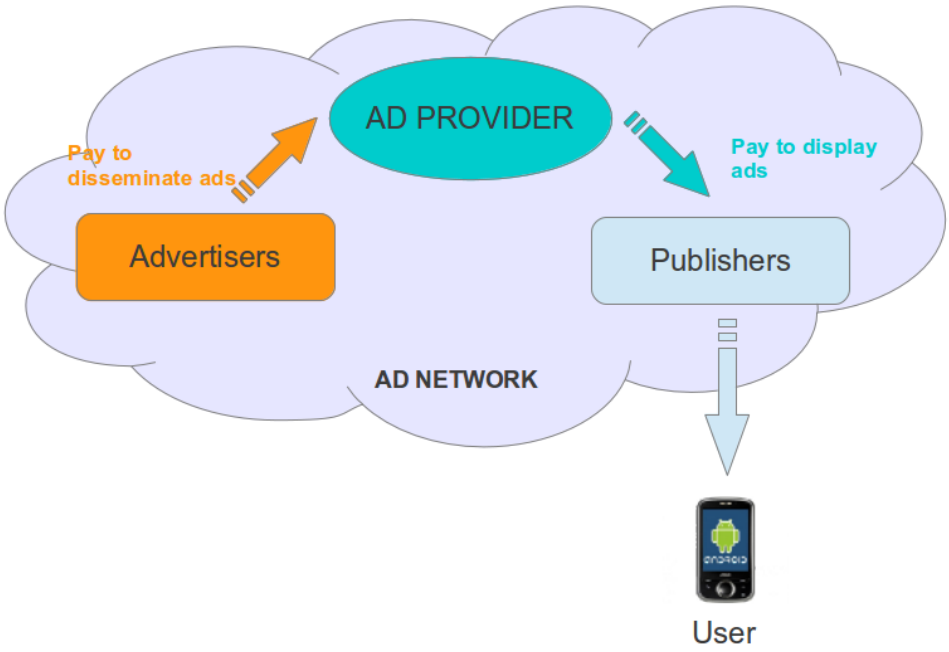

Figure 1: Online advertising model

This translates into ad-kits communicating with the ad-network (see Figure 2). The libraries send *ad-requests* to *ad-servers* owned by the *ad-providers*. User and device information gathered by the ad-kit will be sent in the ad-request. The ad-servers respond with the more targeting ad, this is called *ad-serving*. The libraries then display the ad and handle user interaction with it. Although Android is designed so that each application runs as a separate Linux user, third-party libraries run in the same process space as the app they are embedded in, thus being granted the same permissions as the host app [4] [5]. Hence, it is how and which information is collected and sent by ad-kits that raises concern about user privacy.

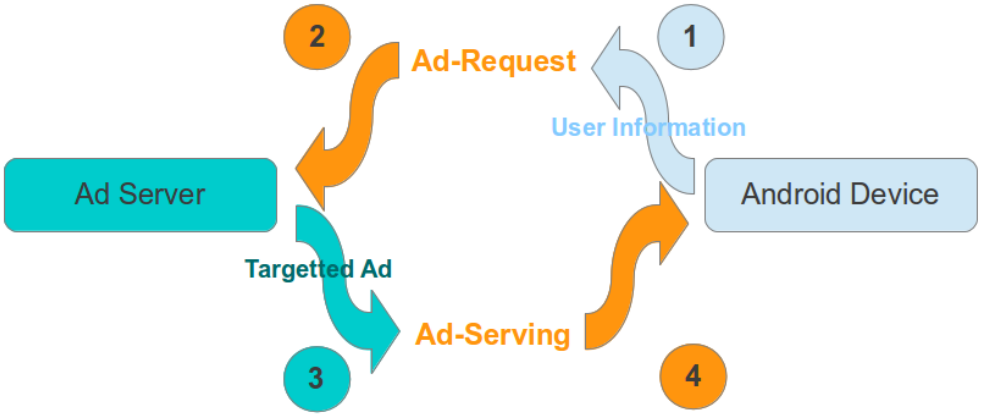

Figure 2: Ad-serving process

In this paper, we examine the behaviour of ad-kits. We've collected 120,000 Android malware samples for our study. From this dataset, we extracted statistics showing ad-kit usage and popularity. We designed analysis tools to inspect the code within these samples and the embedded ad-kits, and identified potential risks. We then specifically reverse-engineered closely over 60 different ad-kits and assessed the extent to which they collect and use private information. Finally we developed 2 applications to expose how they track users across multiple applications, and expose the vulnerabilities allowing a Man-In-The-Middle attack.

Research Goals: Although a number of studies have documented that Android libraries are able to use application permissions to gain access to private user information, these studies have focused on permissions and static analysis. This paper seeks to extend that work by reverse-engineering the code inside ad-kits, and analysing both static in-code data, and dynamic data exchanges between ad-providers and their ad-kits.

2 State of the Art

[4] studied potential privacy and security risks posed by embedded in-app advertisement libraries on the Android platform. Libraries are statically examined and study is focused on permissions and their abuse. Their results indicate the relationship between embedded libraries and host applications is one main reason behind the exposed risks, and show the need for better regulating the way ad libraries are integrated in Android applications.

[6] conducted research on the evolution of uses of permissions in ad-kits. They observed an increase in the number of permissions ad-kits were using, with the exception of localization permissions (FINE_LOCATION and COARSE_LOCATION) which were being less used than before.

[5] investigated potential privacy vulnerabilities in advertisement kits. They found that several ad-kits checked for and used undocumented permissions. They also regularly sent private data in ad requests such as age, gender or keywords.

[7] analysed the mobile advertisement ecosystem by making use of a unique anonymized dataset corresponding to one day of traffic for a major

European mobile carrier with more than 3 million subscribers. Their analysis demonstrates the importance of energy and network overhead of mobile ad traffic.

3 Methodology

In order to conduct our analysis, we collected 120,000 Android malware samples from our malware collection. The analysis of large quantities of malware is difficult, mainly because of the time it takes. For instance, simply counting the popularity of a given ad-kit (number of times the ad-kit is used for the malware database) becomes an issue if scripts are not properly designed. In that case, decompiling each sample and searching for ad-kit usage is not an option: too long.

The 1st method that was tried was to compute 1 hash per ad library. The idea was to quickly identify ad libraries from our sample collection. This method proved to have some major setbacks. Because each library has multiple versions as it evolves and changes with time, and because many developers use tools like ProGuard to shrink and obfuscate code, one ad-kit could produce many different hashes. Rather, we went for designing scripts that automatically inspected code within the Dalvik executable itself, thus saving time from decompilation and rendering more accurate results on ad-kit usage.

From our malware collection, we selected samples embedding at least 1 ad-kit. This 1st selection showed more than a 3rd of our total samples collection actually implement 1 or more ad-kits.

We were then able to run our Droidlysis tool, specifically designed to analyse security risks and breaches inside mobile applications, on the 40,000 samples containing ad-kits. Running Droidlysis helped us identify some 60 different ad libraries from the disassembled code.

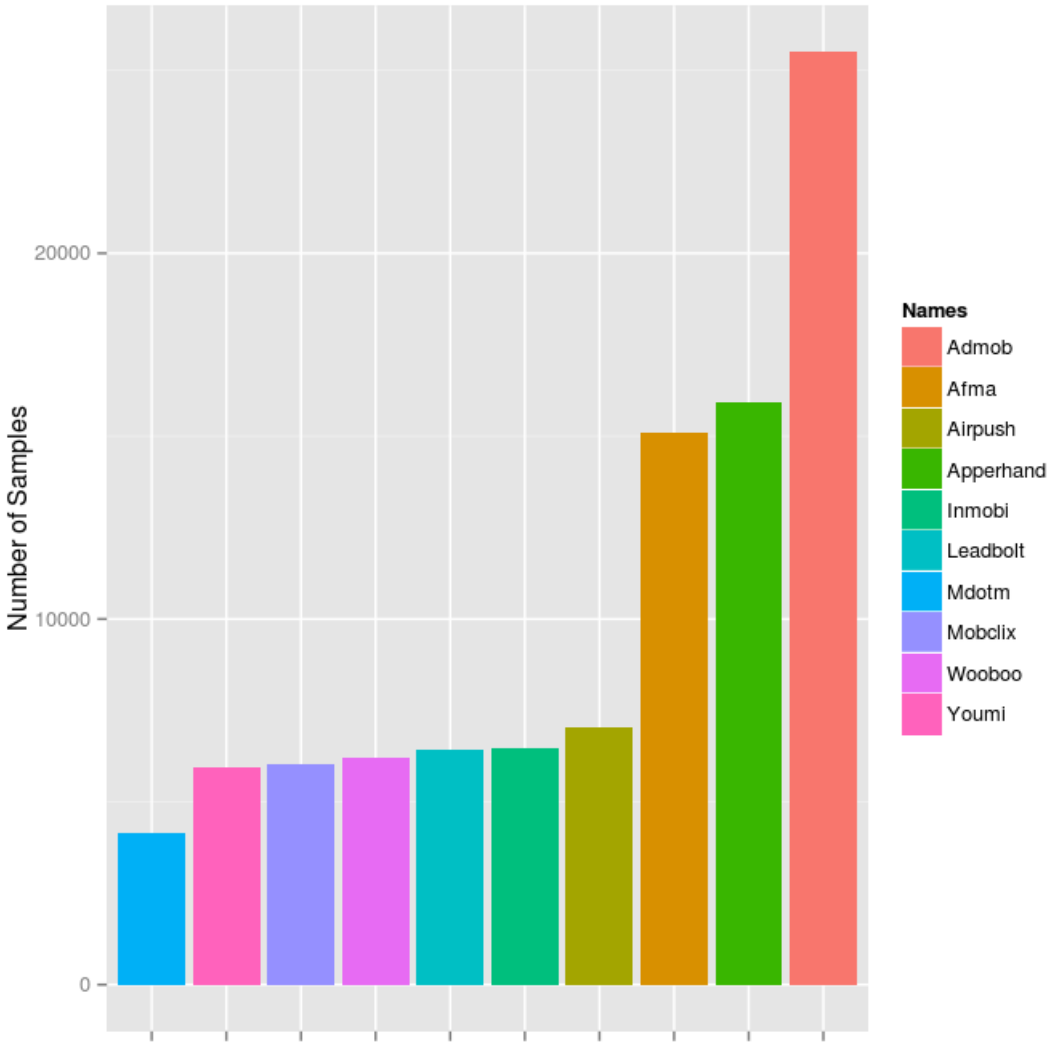

Figure 3: Number of malware samples containing each of the 10 most popular ad-kits

4 Reverse-engineering of Ad-kits

Because [4] and [6] have already extensively covered the risks related to permissions abuse, we've only re-examined the 10 most popular ad-kits in order to assess any changes/improvements on that matter. The rest of our study, which goes beyond required permissions, covers 60 different ad-kits.

We carefully examined each ad-kit's official SDK Integration Guide, comparing guidelines to what is actually done in their code. Our results show there haven't been any improvements regardless of the number of articles and papers on the subject.

We re-use a similar notation to [5] for Table 1: ad-kits may require

Adkit	INTERNET	CHECK ROOTED	ACCESS_NETWORK_STATE	READ_PHONE_STATE	ACCESS_COARSE_LOCATION	ACCESS_FINE_LOCATION	CAMERA	CALL_PHONE	WRITE_EXTERNAL_STORAGE	READ_CALENDAR	WRITE_CALENDAR	READ_CONTACTS	WRITE_CONTACTS	SEND_SMS	RECEIVE_BOOT_COMPLETE	GET_ACCOUNTS	ACCESS_WIFI_STATE	INSTALL_SHORTCUT	READ_HISTORY_BOOKMARKS	WRITE_HISTORY_BOOKMARKS	VIBRATE	BATTERY_STATS	WAKE_LOCK
Admob	R		R														R	R	O	O			
Apperhand	R			R																			
Adsense	R																						
Airpush 5.0	R		R	R	O	O									R	O	O	R	P	P	O		
InMobi 3.6.2	R		O		O	O		O									O							
LeadBolt	R		R	R											R		R	R					R
Wooboo	R		R	R	R				R			R		R			R						
MobClix 4.1.0	R	P	O	R	P	P	P			P	P	P	P			P					P	P	
Youmi	R		R	R	O				R								R	O					
Mdotm	R		R	O					O								O						

Table 1: Revised table of permissions as of June 2013

(R) permissions, recommend optional (O) permissions, or at worst, probe (P) for undocumented permissions.

Without any surprise, ad-kits like to collect and use lots of information from our devices. Perhaps, the surprise is rather to which extent.

Appendix A lists data we have seen to be used in the 60 ad-kits we reverse engineered closely. There are more than 50 fields, the most sensitive ones being: cell ID, GPS coordinates, IMSI, income, marital status, politics, religion, indicator whether device is rooted or not, sexual orientation. Top queried fields are listed at Figure 4.

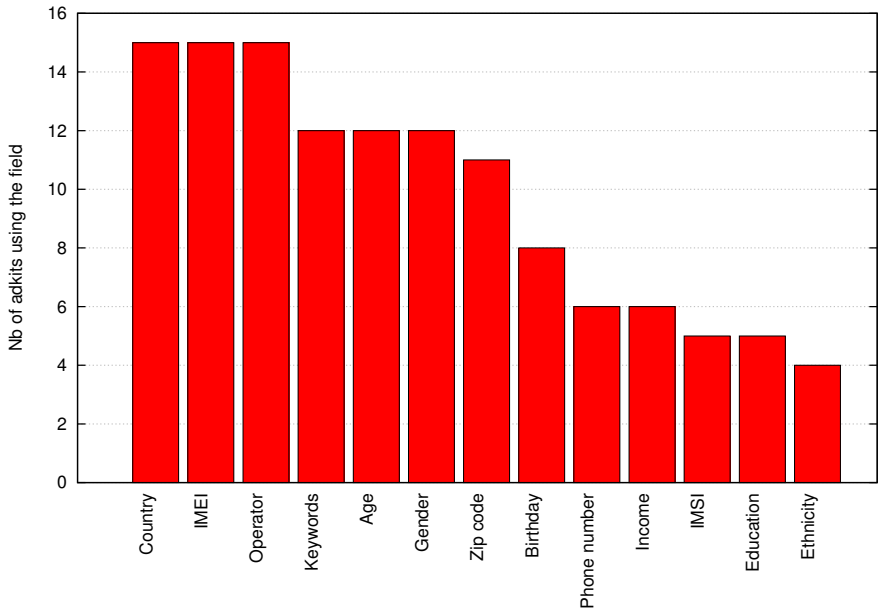

Figure 4: Top queried fields in 60 different versions of ad-kits

The sensitivity of those fields are quite personal. They may also vary from one country to another. For example, in Lebanon, religion is part of one's identity like a first or last name, and may thus be considered as non sensitive. But in other countries, certain specific religions are considered suspicious.

The wide range of fields potentially collected is not our sole concern:

Ad-kit	Field	Entries
InMobi Android SDK 3.0.1	Education	`highschool, somecollege, incollege, bachelorsdegree, mastersdegree, doctoraldegree, other`
Millennial Media 3.6.3	Sexual orientation	`straight, gay,` **bisexual**, **notsure** (see code at Figure 6)
	Marital status	`single, married, divorced,` **swinger**, `relationship, engaged`
Mobclix 3.4	Gender	**both**, `female, male, unknown` (see code at Figure 5)
Mobclix 4.0.1	Religion	`buddhism, christianity, hinduism, islam, judaism, other, unaffiliated, unknown`
	Marital status	`married, single available,` **single unavailable**, `unknown`
Quattro Wireless SDK 2.1	Age	`12-17,18-24,25-34,35-49,50-54,`$\geq$`55`
	Income	`<35000, 35000-49999, 50000-74499, 75000-99999,100000-149999,`$\geq$`150000`
Transpera Mobile Video Ad	Ethnicity	`asian, white, black, hispanic, american indian, alaska native, pacific islander, other`

Table 2: Striking level of details for field entries collected by Ad-kits

the level of details of those fields is also an issue. The most striking details we saw are listed at Table 2.

```
public static final String Gender = "dg";
public static final int GenderBoth = 3;
public static final int GenderFemale = 2;
public static final int GenderMale = 1;
public static final int GenderUnknown = 0;
```

Figure 5: Mobclix 3.4 Demographics gender field

```
if ((this.orientation == "straight") ||
    (this.orientation == "gay") ||
    (this.orientation == "bisexual") ||
    (this.orientation == "notsure")) {
    ...
```

Figure 6: Millennial Media 3.6.3 sexual orientation field

Also, despite the fact [6] notes a decrease in use of GPS permissions (`ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION`), we observe geographic coordinates are still very much used by ad-kits (approximately

50%), and more worrying, that at least 8 ad-kits send those coordinates in cleartext:

- AdGroup SDK 1.4.3
- AdYip 1.0
- LeadBolt 1.3
- Mobclix 2.0.0
- MobFox SDK 1.2
- MoPub 1.6.0 and 4.0
- Smaato SDK 2.0.1
- Wooboo SDK 1.1

For example, this is a sample URL sent to Mobclix ad servers:

```
http://ads.mobclix.com?p=android
  &i=APPLICATION_ID
  &u=DEVICE_ID
  &m=MOBCLIX VERSION
  &ct=CONNECTION TYPE
  &ll=LATITUDE,LONGITUDE
  &l=LOCALE
  &dt=MODEL
  &sv=ANDROID VERSION
```

Note the presence of latitude and longitude in clear text.

In all ad-kit versions we reverse engineered, only a single one, Ximad v2.2, posts the coordinates to a HTTPS page. Google Ads 4.3.1 takes care to encrypt the coordinates using AES-CBC (a robust algorithm) but unfortunately uses a hard-coded key, easy to find after decompilation of any package... (see Figure 7)

```
v1[1] = on.valueOf(((long) (p9.getLatitude() * 10000000.0)), v4);
v1[2] = on.valueOf(((long) (p9.getLongitude() * 10000000.0)), v4);
v1[3] = on.valueOf(((long) (p9.getAccuracy() * 1000.0)), 1000.0);
com.google.ads.util.AdUtil.b(String.format("..."))
..
```

In com.google.ads.util.AdUtil.b:

```
v0 = javax.crypto.Cipher.getInstance("AES/CBC/PKCS5Padding");
v3 = new byte[16];
v3 = {10, 55, 144, 209, 250, 7, ... };
v0.init(1, new javax.crypto.spec.SecretKeySpec(v3, "AES"));
v1 = v0.getIV();
v0 = v0.doFinal(p6.getBytes());
```

Figure 7: Encrypting GPS coordinates with an easy to find hard-coded key is not a safe way to protect location... (excerpt from Google Ads 4.3.1)

Note that tracking is not limited to GPS coordinates. Ad-kits also make extensive use of IMEIs, UUIDs, MAC addresses or IP addresses (see Figure 8), which raise concerns for anonymity. Using a hashed or encrypted identifier is hardly better. It conceals the real values - which is

interesting in particular for the IMEI as that identifier is used by operators to declare phone theft - but does not improve anonymity (end-user can still be tracked). From a development perspective [8], UUIDs are the best - but yet not perfect - choice, and for anonymity too, if UUIDs are not shared among various applications.

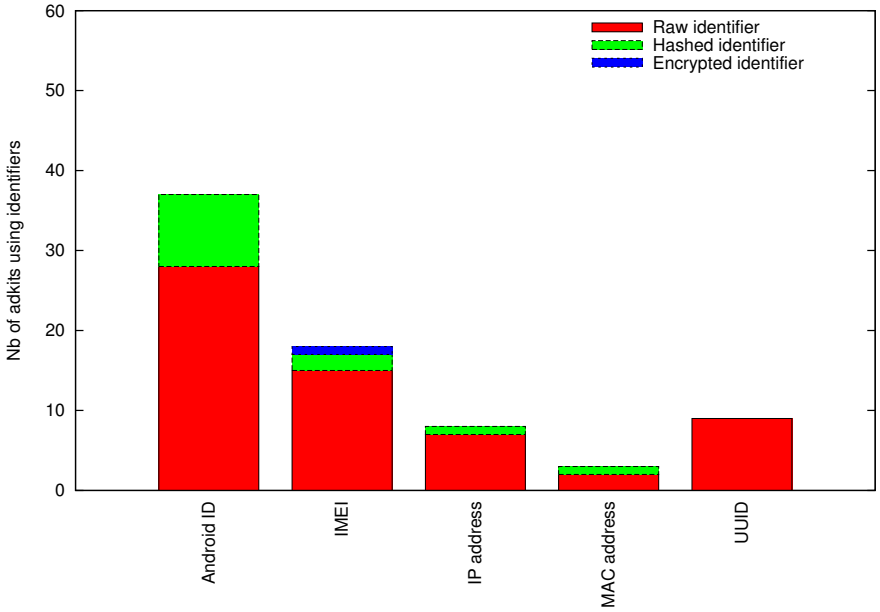

Figure 8: Different identifiers used in ad-kits

Our reverse engineering also reveals borderline implementation style. Like many other applications - genuine or not - ad-kits like to obfuscate their code. Around 40% of ad-kits we analyzed used some level of obfuscation. Obfuscating one's code is not reprehensible though, yet obfuscating one's namespace is strange (Airpush - `com.airpush.android` - has been seen to use namespace `com.klYv.TsrC111182`), deleting logs arouses suspiciousness (Mobclick - see Figure 9 line 19) and using reflection to silently retrieve account emails (Pontiflex - see Figure 10 lines 10, 15, 19 and 22) is no more than an attempt to hide one's behaviour.

```
1  private static String d(android.content.Context p12)    {
2    v4 = 0;
3    v0 = p12.getPackageName();
4    v1 = new java.util.ArrayList();
5    v1.add("logcat");
6    v1.add("-d");
7    v1.add("-v");
8    v1.add("raw");
9    v1.add("-s");
10   v1.add("AndroidRuntime:E");
11   v1.add("-p");
12   v1.add(v0);
13   v3 = new String[v1.size()];
14   v2 = new java.io.BufferedReader(
15               new java.io.InputStreamReader(
16                       Runtime.getRuntime().exec(
17                           v1.toArray(v3)).getInputStream()),
18                           1024);
19 ...
20   Runtime.getRuntime().exec("logcat -c");
```

Figure 9: System log manipulation in Mobclick Agent 2.1.1

An operational email, i.e an email which is read by a real person, is obviously valuable to ad-kits and Pontiflex SDK are not the only ones to try and retrieve them: so does (at least) Applovin SDK 3.4.4 and Airpush. In our sense, the problem lies in the fact that end-users are not aware this is happening. The application requests the permission `android.permission.GET_ACCOUNTS` whose documentation says:

"Allows access to the list of accounts in the Accounts Service"

The documentation does not explain the permission allows an application to retrieve data tied with accounts, i.e names, or email addresses.

```
public static String getDeviceSetupEmail(Context p26) {
  if (p26 != 0) {
     v9 = AdManager.getAdManagerInstance(p26.getApplicationContext());
     if (v9.isAllowReadDeviceData() != 0) {
        v11 = new StringBuilder();
        if (p26.getPackageManager().checkPermission(
                  "android.permission.GET_ACCOUNTS",
                   p26.getPackageName()) == 0) {
            v5 = Class.forName("android.accounts.AccountManager");
            if (v5 != 0) {
                v0 = new Class[1];
                v21 = v0;
                v21[0] = android.content.Context;
                v16 = v5.getMethod("get", v21);
                v0 = new Object[1];
                v23 = v0;
                v23[0] = p26;
                v19 = v16.invoke(v5, v23);
                if (v19 != 0) {
                    v0 = new Class[0];
                    v15 = v19.getClass().getMethod("getAccounts", v0);
```

Figure 10: Decompiled code of Pontiflex SDK. The getAccounts() method of the AccountManager class is not called directly, but via Java reflection

Among strange to suspicious code we observed as analysts, there are yet 2 other issues:

- We noticed Mobclix 4.0.1 checks whether the device is rooted or not. The information is sent out (in clear text) to their servers, we do not know what they use it for.

```
public boolean isDeviceRooted()    {
  boolean v2 = 1;
  if (this.rooted == -1) {
      Runtime.getRuntime().exec("su");
      this.rooted = 1;
      if (this.rooted != 1) { v2 = 0; }
  } else {
      if (this.rooted != 1) { v2 = 0; }
  }
  return v2;
}
```

Figure 11: Code in Mobclix 4.0.1 which tests whether the device is rooted or not

- Several ad-kits also take great care to detect Android emulators. While this could be a genuine action (so as not to display ads within the emulator?), it can also be seen as an attempt to hide from analysts.

- AdsMOGO SDK 1.0.3 tests if the IMEI is = 000000000000000
- Google Ads 4.3.1 tests if Build.BOARD = unknown, Build.DEVICE = generic and Build.BRAND = unknown
- Mobfox 1.4 tests the value of android_id. The value 0000000000000000 corresponds to Android emulators, while 9774d56d682e549c is a common android_id which was shared by several devices by error.
- Chartboost 2.0.1 tests if Build.PRODUCT.equals("sdk")

Finally, in Applovin 3.4.4, we uncovered potentially dangerous coding. The ad-kit extends the DexClassLoader class which allows the loading of .apk or .jar files that contain a classes.dex entry *without triggering a formal application install*. This means that the APK or JAR gets executed silently, without any warning for the end-user - see Figure 12.

```
v2 = new java.io.File(p9.getDir("al_sdk", 0), v1);
if((v2.exists() == 0) || (v2.length() <= 0.0)) {
   ...
} else {
   this.d = new SdkClassLoader(v2,
                               p9.getDir("al_outdex", 0),
                               SdkBootstrap.getClassLoader());
}
```

Figure 12: Applovin SDK 3.4.4 silently downloads its updated SDK to a directory named al_sdk. SdkClassLoader is a class which extends the powerful (and dangerous) DexClassLoader class

5 Tracking Users

Given the methods implemented to identify devices and track them (see Figure 8), it was fairly simple to set-up an environment to demonstrate how ad-providers can track users across applications using the same ad-kit, and easily build long-term user profiles.

Indeed, all ad-providers use the same identifiers, whether they are hashed or raw, regardless of the application in which the ad-kit is embedded. This means ad-providers can correlate data coming from different applications on the same device.

In order to illustrate this point, we developed 2 applications:

- Foo: with hard-coded personal information and access to all permissions but geo-location ones.
- Bar: with permissions to access GPS location (READ_COARSE/FINE_LOCATION)

We then integrated the Mobclix 4.1.0 ad-kit in both applications, and installed both applications on our Android Emulator. Both applications use the Mobclix *sendDemographics()* method to send information to the Mobclix ad-servers in order to receive targeted ads

- Foo: The information sent includes the user's age, which is harcoded into the application.
- Bar: The information sent includes the user's GPS coordinates which are forced onto the Android emulator.

Both applications were run into the same emulator instance, and we set-up Wireshark to control HTTP requests sent to the Mobclix ad-servers. Both applications sent their respective data to the ad-servers. Surely enough however, the ads sent back proved to be targeted to the same profile.

The *Foo* application was 1st launched with a hard-coded user age of 15 years old. The ad shown in the *Bar* was an ad for a mobile gaming application. We then changed the age in *Foo* indicating the user was 70 years old. The ad that popped up on the *Bar* application was an ad for a golf resort.

A network sniffer can easily do the same correlations, across different ad-providers (see Figure 13). Because of how ad-providers use the same consistent identifiers (see Figure 8), and because all requests are sent through clear text, a sniffer can easily track requests coming from the same device. This means not only can an ad-provider build long-term user profiles based on a user's personal habits, but it also means ad-providers can also gather personal information from other ad-providers by cross-referencing data.

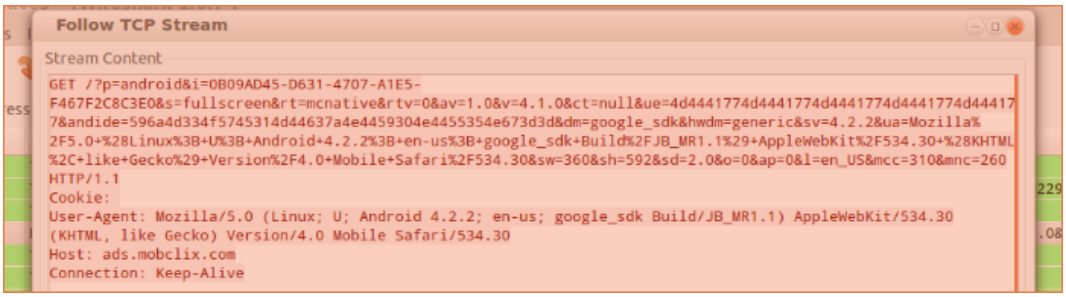

Figure 13: Wireshark traffic capture. The *andide* field is the device's Android ID

Moreover, with clear-text requests/responses, a man-in-the-middle attack is also pretty straightforward. Not only can the attacker listen and intercept requests, but he can easily impersonate the ad-server and send whatever he wants instead of the ad, potentially able to send exploits and execute malware onto the mobile device.

6 Conclusion

In this paper we examined security and privacy issues raised by in-app advertisement libraries (ad-kits). We analyzed 120,000 malware samples from our malware collection. Although data collection is a known issue with ad-kits, our study shows to which extent this is done. We discover the level of details collected by ad-kits, the most sensitive being cell ID, GPS coordinates, IMSI, whether the phone is rooted, income, marital status, politics, religion, sexual orientation. This is information most people might share based on intimate relationships, but most probably wouldn't want exposed publicly.

Our research also shows how some ad-kits silently download remote applications. We found how anonymity is seriously compromised by the methods implemented to track users, using consistent data such as raw

Android ID and IMEI. Experiments show how ad-providers can effectively build long-term user profiles correlating data from different applications. Security and anonymity are further compromised by clear-text data transmission. Among all ad-kits that we reverse-engineered, only 1 used a secure connection to transmit collected data. Clear-text transmissions allow anyone listening on the network to collect and correlate information between different applications, and across different ad-providers.

Although people are now aware of some security and privacy breaches on their computers, they hardly understand to what extent. In reality, lives are totally exposed. As very few realise the same breaches exist on smartphones, the issue is even worst. This paper shows the full extent as to how both advertisers and ad-providers can build very rich and accurate user-profiles. And in this era where technology prevails, data is the only relevant currency.

Appendix A

List of fields collected by 60 different versions of ad-kits:

- accelerometer
- android_id
- age
- birthday or birthdate
- have children or not
- city
- company
- country
- country code,
- **cell ID**
- device model and brand
- education
- emails
- ethnicity
- facebook ID
- first name, last name
- gender
- **GPS longitude and latitude** (not altitude)
- IMEI
- **IMSI**, MCC or MNC
- IP address
- **income**
- interests
- kernel version
- keywords
- LAC

- language
- locale
- MAC address
- **marital status**
- network or SIM operator
- OS version or name
- package version or name
- phone number
- **politics**
- presence of an accelerometer
- presence of a GPS
- presence of a memory card
- **religion**
- ro.serialno
- **rooted indicator**
- SDK version
- **sexual orientation or dating gender**
- SIM serial number
- state
- street address
- timezone
- twitter ID
- UUID
- zip code or area code

References

[1] Gartner. Gartner says worldwide mobile phone sales declined 1.7 percent in 2012. `http://www.gartner.com/newsroom/id/2335616`, 2013. [Online; accessed 15-March-2013].

[2] Donald Melanson. Eric schmidt: Google now at 1.5 million android activations per day. `http://www.engadget.com/2013/04/16/eric-schmidt-google-now-at-1-5-million-android-activations-per/`, 2013. [Online; accessed 18-April-2013].

[3] Harry McCracken. Whos Winning, iOS or Android? All the Numbers, All in One Place. `http://techland.time.com/2013/04/16/ios-vs-android/`, 2013. [Online; accessed 18-April-2013].

[4] Michael C. Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.

[5] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen. Investigating user privacy in android ad libraries. In *Workshop on Mobile Security Technologies (MoST)*, 2012.

[6] Theodore Book, Adam Pridgen, and Dan S. Wallach. Longitudinal analysis of android ad library permissions. *CoRR*, abs/1303.0857, 2013.

[7] Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. Breaking for commercials: characterizing mobile advertising. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, IMC '12, pages 343–356, New York, NY, USA, 2012. ACM.

[8] Tim Bray. Identifying app installations, March 2011. `http://android-developers.blogspot.fr/2011/03/identifying-app-installations.html`.