Android malware situation
Author

Asier Martínez Retenaga

This study has been carried out with the collaboration of Jesús Díaz Vico and Héctor R. Suárez on the behalf of INCIBE and Francisco López Hernández and Javier Rascón Mesa on behalf of HISPASEC.

February 2015

This publication belongs to INCIBE (Spanish National Cybersecurity Institute) and is under an Attribution-NonCommercial 3.0 Spain license from Creative Commons. For this reason, it is allowed to copy, distribute and publically communicate this work under the following conditions:

• Attribution. The contents of this report may be totally or partially reproduced by third parties, citing its origin and making explicit reference both to either INCIBE or CERTSI and its website: http://www.incibe.es. Such acknowledgement may never suggest in any case that INCIBE supports this third party or the usage it makes of the work.

• NonComercial use. The original material and its derivatives may be distributed, copied and exhibited as long as it is not with commercial purposes.

By reusing or distributing this work, the terms of the license of the work must be made clear. Some of these terms may not be applied if CERTSI grants permission as holder of the author rights. Complete license text: http://creativecommons.org/licenses/by-nc-sa/3.0/
# INDEX

1. **CURRENT SITUATION** 5  
2. **ABOUT THE STUDY** 8  
3. **ANDROID SECURITY MODEL** 9  
4. **APK FORMAT** 12  
5. **ANALYSIS OF SAMPLES** 14

5.1. Types of files 14  
5.2. Signed APKs 15  
5.3. Monetized APKs 16  
5.4. SDKVersion 22  
5.5. THREAT Exposure time 25  
5.6. Development platform 27  
5.7. Package Name 27  
5.8. Existing Package Names in Google Play 29  
5.9. Permissions 31  
5.10. Services 36  
5.11. Receivers 37  
5.12. Providers 38  
5.13. Activities 38  
5.14. Complexity 39  
5.15. Dangerousness 40  
5.16. Default language 41  
5.17. Alternative languages 42  
5.18. Most used words 44  
5.19. Established connections 45  
5.20. Antivirus detections 47  
5.21. Types of malware 48
5.22. Privacy

6 Conclusions

Annex 1 – Comparison of Monetization Libraries

Annex 2 – List of Antivirus Engines

Annex 3 – List of Methods That Affect Privacy

List of Figures

References
Android has become the most attacked mobile platform, as shown in the following examples.

- Kaspersky Lab’s malware report, corresponding to the second quarter of 2014, indicates that Android attracts 99% of malicious mobile applications.
- «Cisco’s 2014 Annual Security Report» also indicates that 99% of malware on mobile platforms is targeting Android.
- Kindsight Security Labs estimates that there are 15 million mobile devices infected with malware around the planet, almost over four million more than registered in the final stages of 2013, 60% of which use Android.
- Arxan’s «State of Mobile App Security» report reveals that there are malicious versions of 97% of the TOP 100 paid apps and of 80% of the TOP 100 free apps on Android. These are distributed through a large amount of alternative markets.
- «Cheetah Mobile’s security report» indicates that it has identified over 2.2 million malicious applications which correspond to Android.
Quick Heal’s «Security Threat Report 2014» reveals that malware for Android has increased by 600% since 2012.

This is mainly down to six factors:

1. The choice of Android as a mobile platform by most users: there are already over 900 million devices.
2. This platform has a number of flaws as shown in the «Android Open Source Project - Issue Tracker». Likewise, the security controls it incorporates are not 100% trustworthy. Some of the most relevant news on the subject are the following:

   - BlueBox investigators published a vulnerability known as «MasterKey», which was subsequently explained in detail at the «Android: One Root to Own Them All» conference at Black Hat USA 2013. This security flaw enabled the injection of arbitrary code in any signed APK without altering the signature. Two files with the same name have to be included in the APK for this to happen, because Android’s verifications are only applied on the first file, with the second file being executed subsequently.

   - A Chinese researcher discovered a vulnerability which enabled the modification of an APK’s content without Android’s validation mechanisms detecting it. Despite the fact that this vulnerability is difficult to exploit and only occurs on certain occasions (by modifying classes.dex files smaller than 64kb), it could allow an attacker to obtain sensitive information belonging to the affected user.

   - Bluebo researchers published a vulnerability known as «FakeID», which was later detailed at the «Android Fake ID Vulnerability» conference at Black Hat USA 2014. This vulnerability, which has affected users since January of 2010, consists of Android’s installation package not verifying the certificate chain, meaning that if the certificate issuer of an application is falsified, Android will not detect it and will accept it as a valid certificate.

   - Researcher Rafay Baloch published a vulnerability that affected the default browser in around 75% of Android devices. With this vulnerability the SOP «Same Origin Policy» can be avoided, meaning that it is possible to obtain information from the other pages open on the browser and access cookies, and if they are not protected, they are used to steal the user’s identity or hijack their sessions.

3. Google Bouncer’s security control is not infallible when accepting new applications on Google Play.
4. The exploitation of these flaws results in a great economic benefit with a minimal investment. A huge amount of sensitive information can be obtained such as photographs, emails, accounts and passwords, SMS, recordings, etc. which can then...
be sold on underground markets for identity thefts, blackmails, etc. Likewise, subscriptions to SMS Premium services generate a lot of earnings for cybercriminals, or even hiring specific botnet services such as denial-of-service attacks, or mass mailings belonging to phishing companies, or 0days on Android which people pay up to between 30 million and 60 million dollars, according to Forbes magazine.

5. A large percentage of Android applications are not developed using methods that emphasize on «security by design». Likewise, others such as OWASP or OASAM aimed at application security and geared towards identifying threats corresponding to security in mobile devices and providing the necessary controls to reduce their impact and the probability of exploitation, are not used either. According to Gartner’s «Predicts 2014: Mobile Security Won’t Just Be About the Device» report, over 75% of the applications developed in 2015 will fail the basic security tests. In this aspect, certain default options in some components of Android applications do not help too much. For example, unless configured otherwise, a provider enables the information that an application shares to be accessible both to the system and third parties, unless if the android:exported="false" attribute is established.

Likewise, it is important to acknowledge that when an application is being developed, its code tends to be reused; we are not going to reinvent the wheel. According to a study published at RSA Conference USA 2014 by Codenomicon, between 80% and 90% of software developed for mobile devices is done using reused code. The security errors that the reused code could have are not kept in mind in the majority of cases.

6. Users are not aware of the real threat and, in many cases, do not take the necessary measures to maintain their security and privacy. The «Mobile Security 2015: The Enterprise at Risk» report indicates that 34% of users do not take any kind of security related measures in their devices.

All this leads to cybercriminals consistently generating applications that look to exploit both the inexperience and credulity of users and the security errors of the system itself. These applications are mainly aimed at carrying out the following activities:

- Subscribing to «SMS Premium» services.
- Downloading and installing unwanted applications, mainly coming from alternative markets or unofficial locations.
- Tracking the device’s location and recording audio files or videos to monitor the user.
- Monitoring SMS from banking services.
- Stealing personal information such as contacts, images, videos.
- Pretending to be useful applications when in reality they are not, such as false antivirus software.
- Hijacking of the mobile device.
2 ABOUT THE STUDY

Given the scenario described in the Current situation section, it is necessary to analyse the technical features of malware in Android in depth: «If you know your enemies and know yourself; you will not be imperiled in a hundred battles, you will never be defeated». (The art of war – Sun Tzu). Therefore, it is possible to know in depth certain aspects that are common in these types of threats and therefore apply the necessary measures to reduce their impact and exposition.

It has been possible to carry out this report thanks to a collaboration between the Spanish National Cybersecurity Institute (Instituto Nacional de Ciberseguridad), INCIBE and HISPASEC, in a joint effort to offer a concrete vision from an analytical point of view. For this purpose, in the second half of 2014, a total of 76,000 samples of malicious applications have been obtained, and analysed with the collaboration of HISPASEC.

The analysed samples are detected by antivirus with large malware detection rates in Android, with the aim of giving the study an important degree of reliability and avoid including false positives. The selected antivirus were: ESET-NOD32, BitDefender, Commtouch, Gdata and MicroWorld-eScan.

To carry out the analysis, on one hand, reports from Virustotal have been used. This service not only analyses the samples with a multitude of antivirus engines, but also uses Androguard to perform a static analysis of the samples. When decompiling the applications, this utility enables access to great amounts of information such as permissions, services, receivers or providers, the publicity libraries they used, the list of activities or significant chains, etc. Besides, other tools, also based on Androguard, have been adapted for extracting additional information from the samples, all being used to generate this report.
3 ANDROID SECURITY MODEL

In order to understand the Android malware features it is important to have a clear picture of this mobile operating system’s security model to be able to assess its strengths and weaknesses, and therefore have an idea of the deficiencies or aspects that need improving that are currently being exploited by malware developers. The architecture of Android is based on a multilayer model as shown in Figure 1:

- **Applications** – corresponds to the highest level of the model and is composed by applications installed by default and user-installed applications.
- **Framework** – is the layer that contains the Android system’s key functions such as Package Manager which enables installing/deleting Android applications, ActivityManager controlling the life cycle of every activity in every application, etc.
- **Libraries** – corresponds to the libraries that use the different components in the Android system. Some of the most distinguished libraries are multimedia, graphic engines or SQLITE database engines.
- **Android Runtime** – is made up of two components: CORE libraries and Dalvik virtual machine or its successor, ART. All the applications are executed in the virtual machine with the aim of providing the model with secure surroundings.
• **Kernel** – corresponds to the abstraction layer between the hardware and software. Includes essential services such as memory or process management, or the drivers enabling an interaction with the camera, audio, Wi-Fi, etc.

The security model is designed to prevent regular attacks such as social engineering, which aim to install third party applications. To prevent these attacks, it incorporates a number of protections that reduce the probability of being affected by them, as well as the impact in the case of an infection occurring. Some of the most important measures that the platform includes to protect itself from different types of threats are the following:

• Google Play incorporates a number of controls, such as not enabling various applications to have the same avatar image or same name. Likewise, it blocks the publication of applications in the following cases:
  o Applications that include illegal content.
  o Applications that facilitate real betting games.
  o Applications that include hate-promoting content.
  o Applications that include pornography.
  o Applications that include real violence for free.

Moreover, it includes a service known as «**Bouncer**», which is responsible for verifying the applications that are uploaded to Google’s official repository. To carry this out, it performs a dynamic analysis of the applications and signature verifications with regards to malware lists.

• The virtual machine isolates applications when they are executed, preventing an application from gaining access to another.

• The framework for application development incorporates different functions such as permission models and cryptographic functions.

• It incorporates different methods such as ASLR, **ProPolice** or **safe-iop** with the aim of reducing risks associated to the memory management.

• Encryption for both the mobile device and the external SD card.

• By default, the Android platform does not enable the installation of applications from «unknown sources».

• **Application verification systems** warning users when applications that are external to Android’s official market are being installed.

• «Request password for purchases» with the aim of preventing accidental or unwanted purchases.

• **Google Play Services gathers the applications it still does not know about to improve their verification.**
Figure 2: Android incorporates multiple controls to protect its users. Source: Adrian Ludwig – Android’s Chief of Security.

Despite all of these layers, the model presents some loopholes or aspects that need improving:

- Applications are usually signed with self-signed certificates. Therefore, they do not require any certification authority to assure that the application does not present any risks for the user.
- The possibility of creating customized permissions may present a privacy risk.
- The Bouncer’s security controls must be more exhaustive.
- With Google Play Store’s latest changes to its permissions policy, users are no longer notified regarding certain changes in the permissions requested by the applications.
4 APK FORMAT

All the applications for Android devices, including malware, are encapsulated in a specific way known as APK «Application Package File». This format is used for the installation and distribution of applications for this mobile platform.

The structure of APK files is shown in Figure 3:

- **AndroidManifest.xml** – it is the application’s configuration file. Different aspects are identified in it such as the application’s unique identifier, its components («activities», «receivers», «content providers», etc.) or the permissions that the application requires to work properly.
- **classes.dex** – this contains the application’s compiled code in DEX format so that it can be interpreted by virtual machines Dalvik or ART.
- **resources.arsc** – it is the file corresponding to the compiled resources.
- **META-INF** – it is the directory that stores information corresponding to the application’s digital signature and contains the following files:
  - **MANIFEST.MF** – it contains a complete list of the APK’s files along with its respective SHA-1 hash.
  - **CERT.SF** – it contains the SHA-1 hash of every 3 lines that appear in the MANIFEST.MF.
  - **CERT.RSA** – it stores the CERT.SF file’s signature, therefore, it contains the APK’s signature.

Figure 3: Structure of an APK file.
- **res** – it is the directory that stores the resources (images, texts, XML files, etc.) used by the application.
- **lib** – it is the directory that contains the compiled code for different architectures: armeabi, armeabi-v7a, x86 or mips.
ANALYSIS OF SAMPLES

Following are some of the aspects and components of the malicious APKs that have been analysed along with the results obtained from the analysis.

5.1. TYPES OF FILES

All the files included in the analysed APKs have been identified with the aim of determining which types of files repeat themselves the most.

Overall, over 8 million files have been registered, which is an average of 113.44 files per application. Out of all of them, there are 1,300 different types. The most repeated file formats are shown in Figure 5.
Besides the most repeated types of files, there are a number of noteworthy information:

- Odd extensions have been identified such as .VIAGRA or .ACTOR.
- There are a total of 11,925 files with the .APK extension, in other words, APKs that also contain APKs, which leads to consider that they correspond to APK installers that take advantage to perform malicious activity or to droppers belonging to malicious applications.
- Only 26 files with .SQLITE or .SQLITE3 have been identified corresponding to databases.
- Over 200,000 correspond to files with audio extensions: .MP3, .MP4, .OGG, .WAV, etc.
- Over 10,000 correspond to compressed files: .RAR, .ZIP.
- Only 19 files with the .PDF extension have been identified.

5.2. SIGNED APKS

When publishing a Google Play application, it is necessary to have signed it previously as that is one of the multiple verifications that the Bouncer performs to determine whether the applications is malicious or not.

According to Android’s documentation in this [this matter](#): «**Android requires that all apps be digitally signed with a certificate before they can be installed**». The integrity of the APK can be verified through this signature, given that it contains the hash of each of its files. Therefore, when the installation is being performed, Android’s application manager verifies that these hashes are correct and therefore proves that the APK has not been modified.
After analysing all the applications, it has been identified that 100% of them are digitally signed. This comes as no surprise, given that when an APK is being generated, the IDEs generate a debug mode certificate with the tools that Android SDK incorporates.

However, there is an aspect that refers to APK certificates that is significant: as indicated previously, the certificates do not compulsorily pass a control by a certification authority or CA, but they are self-signed, meaning that it is not a specially safe security mechanism.

Continuing with Android’s documentation: «Android uses this certificate to identify the author of an app, and the certificate does not need to be signed by a certificate authority. Android apps often use self-signed certificates».

In this case, the immense majority of the applications analysed possess a self-signed certificate. Only 191 of them possess different information in the issuer from the end user who the «Subject» certificate is issued for, which corresponds to 0.02% of the applications. This also does not guarantee that the certificate is trusted because the information from both the certificate issuer and end user can be easily forged when generating it using Jarsigner or similar utilities. Some of the names that have been used as the common name of the certificate issuer (CN) when generating the certificates are Bill Gates or Steve Jobs.

Google does not encourage the generation of certificates that are validated using certification authorities. Specifically, it request the following: «If you plan to publish your apps on Google Play, the key you use to sign these apps must have a validity period ending after October 22nd, 2033»

The majority of the certification authorities do not issue certificates with such long validity periods, with 5 years being the usual validity period for end users. This policy contradicts the requirements established in the Certification Authority Browser Forum (CAB Forum) in 2012, of which Google is a member and where it is established that as of April of 2015 certificates issued by CAs to end users will have a maximum validity of 39 months instead of 60 months.

5.3. MONETIZED APKS

Many developers include advertising in their applications with the aim of making them profitable and thereby obtain economic benefits. Monetization is something normal and in the right proportions does not result in a negative experience for users [1].

There are different business models for monetizing an application, the following being the most popular:

- **Paid apps**: this model is falling into disuse over time because users tend to prefer installing free apps. Despite this, they still generate large amounts of earnings.
- **Free apps with advertising**: including third party advertising.
- **«Freemium» apps**: they offer an additional function in exchange for money. Normally they do not include advertising because they are funded by the users’
purchases. This business model generates most earnings for developers and is geared towards applications that have large volumes of users, normally games.

- **Subscription**: applications that periodically charge amounts of money and in exchange allow subscribed users to use them.

The most commonly used model both for normal and fraudulent apps is «Free apps with advertising», which works as indicated in Figure 6.

![Figure 6: Free apps with advertising model.](image)

Following is an example of communications between an APK and a server corresponding to an advertising network:

- The app performs the following request:

  http://ads.wapx.cn/action/miniad/ad?app_id=106185f96d1a4106166a4f14ade5d9f&udid=XXXXXXXXXXXXXX&imsi=XXXXXXXXXXXXXX&net=internet&base=wapx.cn&app_version=2.8.9&sdk_version=1.8.8&device_name=Nexus%20S&device_brand=google&y=c98b7c327e6f623890da4b9f5a26de9&device_type=android&os_version=4.0.4&country_code=US&language=en&act=com.windrey.cuteroot=true&channel=mumayi&device_width=480&device_height=800&at=134984907891

  As can be observed, the request includes a series of parameters that correspond to information from the terminal for customizing the ads that are going to be shown: `device_name`, `device_brand`, `os_version`, `country_code`, `language`, etc.

- The server responds with information, in an XML format in this case. Amongst other information, it includes the URLs of the images corresponding to the customized advertising:
The problem comes when a developer floods its app with advertising, when apps of no use are monetized only with the aim of enriching the app developer, or even when the system is used to spread malware through ads that redirect users to these types of threats and therefore compromised the security of the device.

It is important to keep in mind that advertising networks generate a volume of thousands of millions of dollars annually. That is why they are very important target for cybercriminals, as reflected in White Ops’s report «The Bot Baseline: Fraud in Digital Advertising» which indicated that 11% of website ads are visited by bots and 23% of the visits that advertising on a video receives come from botnets, with 1% coming from mobile networks. This will lead to an estimated loss of 6,000 million dollars for advertisers in 2015.

Likewise, all monetization libraries directly affect the privacy of users to a larger or smaller extent, since they usually request a number of permissions with the aim of gaining access to personal information and therefore establish customized adverts. In addition, as demonstrated by FireEye investigators, a large part of these monetization libraries are vulnerable and directly expose users’ information in «Man In The Middle» attacks.

Different methods exist to automatically identify monetization libraries that incorporate APKs, such as «AdDetect: Automated detection of Android ad libraries using semantic analysis» [2]. To carry out the study, the chosen applications have been decompiled and all the packages integrated in them have been compared with a list of around 150 libraries that have previously been identified in a research effort as the most commonly used. In other words, the frequency of use of all the identified monetization libraries has been analysed. Figure 8 shows the data extracted from the analysis of the applications:
13% of the analysed apps incorporate some monetization library.

Of all the libraries identified, only 55 are different, the most repeated ones being the following:

- **waps**: 16%
- **admob**: 12%
- **adwo**: 8%
- **domob**: 7%
- **youmi**: 6%
- **smartmad**: 6%
- **vpon**: 5%
- **adsmogo**: 5%
- **airad**: 4%
- **airpush**: 3%
- **mobwin**: 3%
- **greystripe**: 3%
As reflected in the data, creators of malicious apps do not specifically resort to one service; instead, they tend to use different advertising networks to obtain economic benefits. Therefore, they usually integrate various monetization libraries in one APK. The 9,840 monetized APKs incorporate a total of 26,894 libraries, giving an average of 2.73 libraries per APK.

There are comparisons of these types of libraries from different points of view: «A Comprehensive Guide To All Mobile Ad Networks» [3], «Unsafe Exposure Analysis of Mobile In-App Advertisements» [4] or «Investigating User Privacy in Android Ad Libraries» [5].

Following is a comparative table in terms of the potential risk for users based on the required permissions that the latest versions of the TOP 10 most used libraries request. This is directly related to their privacy risk level, therefore affecting users’ privacy:

<table>
<thead>
<tr>
<th>Library</th>
<th>Privacy Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>admogo</td>
<td>High</td>
</tr>
<tr>
<td>smartmad</td>
<td></td>
</tr>
<tr>
<td>airad</td>
<td></td>
</tr>
<tr>
<td>vpon</td>
<td></td>
</tr>
<tr>
<td>domob</td>
<td></td>
</tr>
<tr>
<td>youmi</td>
<td>Medium</td>
</tr>
<tr>
<td>airpush</td>
<td></td>
</tr>
<tr>
<td>waps</td>
<td></td>
</tr>
<tr>
<td>adwo</td>
<td>Low</td>
</tr>
<tr>
<td>admob</td>
<td></td>
</tr>
</tbody>
</table>

The analysis of the permissions can be seen in depth in «ANNEX 1 – of monetization libraries». 
As indicated previously, 55 different monetization libraries have been identified, although some of them include various versions. Those that are most repeated are shown in Figure 9:

**Figure 9: Top 20 versions of monetization libraries.**

These apps have been analysed with the aim of determining the privacy risk that using apps that integrate these types of libraries could represent for users. For this purpose, their corresponding SDKs «Software Development Kits» have been analysed to identify what information they collect. Generally, monetization libraries obtain the following data from the device and the user:

- **Data concerning the mobile device:** telephone number, language, manufacturer, model, device’s serial number, resolution, Android version, MAC, IP, IMEI or MEID, location, GPS information, type of connection, network operator, connection preferences, active functions (NFC, camera, Bluetooth), list of applications downloaded or installed in the device along with their respective versions, how and when the installed application are used, browsing history.

- **Data concerning the user:** name of the user, date of birth, gender, studies, religious beliefs, marital status, sexual orientation, ethnic group, political ideology, post code, age, email address, hobbies, etc. On some occasions, they even acquire the contact list with their corresponding name, phone number, email and photograph, list of calls, calendar, history of text messages, identifiers for other services such as social networks, etc.

A large part of this data is requested from the user through methods such as surveys, or taking part in prize draws, etc. However, in some cases access is gained directly without interaction or the user’s knowledge.
Through these practices over 50 aspects that enable the generation of the user’s profile are collected. Following is an example of a basic profile with false information that can be obtained through the information collected in these types of libraries:

Josines Segurola, born on the 13/01/1981 (34 years), living in Guardo, married with two children, catholic, email: josinessegurola@gmail.com, with a degree in mathematics, works at an institute, likes sport in general and Chuck Norris films, owns an unrooted Nexus 5, with Android 5.0 and his phone number is 69XXXXXX0, he uses different cloud services and social networks, mainly facebook and his user name is josinessegurola, he has a data rate contract with XXXXX, he regularly visits sports websites and national newspapers, he usually arrives at work on time and always walks the same route, his address is XXXXX road, number 5. He doesn’t spend weekends in his regular home; he goes to a town 20km away from Guardo.

On many occasions, all this information can be shared with third parties, so users lose their control over it. Besides, the information is stored in the servers of the corresponding companies (normally located outside Spain and even in countries that don’t have a solid legislation regarding data protection), and if they suffer a security incident, the information could be exposed and used directly by delinquents to perform criminal acts such as scams or extortion.

The information they collect and what they use it for can be consulted in the privacy policy available on their respective websites, although they tend to only give superficial information and don not go into details.

In some cases, antivirus companies automatically detect all the applications that include some monetization libraries through generic detection because of the high intrusion level.

Android currently incorporates a permissions system at application level, so that when an application is granted a permission, this same permissions are inherited by the libraries included in the application. That is, there is no separation of privileges. On occasions, the requested permissions are necessary for a correct functioning of the library, but not for the application, leading to over-privileged apps [6]. As described, this implies a great risk for users, mainly related to privacy. There are different approximations that aim to reduce or solve the problem, such as «AdSplit: Separating smartphone advertising from applications» [7], «AdDroid: Privilege Separation for Applications and Advertisers in Android» [8] or «AFrame: Isolating Advertisements from Mobile Applications in Android» [9]. However, it is Google’s responsibility to implement the necessary improvements for setting an adequate distinction of the permissions required by the application, without this supposing a problem for its developers.

5.4. SDKVERSION

Android is an operating system that is fragmented in various versions. Throughout the years Google has worked to reduce this problem with measures such as «Google Play Services»,
which enables upgrading Google and Google Play’s apps without the need to execute a complete upgrade of the operating system. Still, as seen in Graphic 10, Android users are still divided into 5 of the operating system’s versions, without acknowledging the stats referring to Android 5.0 «Lolipop», which still is not available on Android’s website.

This means that the performance of the app is different depending on the Android version where it is executed, or even the access to different characteristics or functions of the device. That is why apps in Android are developed establishing different parameters such as «MinSDKVersion», «MaxSDKVersion» and «TargetSDKVersion».

These values correspond to the minimum version of Android that is compatible with the app, the maximum version of Android that is compatible with the app and the optimum version, which the app has been tested with normally.

Figure 11 shows the data extracted from the analysis of the apps:

**Figure 10: Distribution of Android’s versions.**

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>0.4%</td>
</tr>
<tr>
<td>2.3.3 - 2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>7.8%</td>
</tr>
<tr>
<td>4.0.3 - 4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>6.7%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>19.2%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>20.3%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>6.5%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>39.1%</td>
</tr>
</tbody>
</table>

**Android malware situation**
Generally, malware creators for Android aim for their apps to be compatible with the greatest amount of devices possible, so they establish low values in «MinSDKVersion». 68.19% of the analysed applications are compatible with versions of Android that are no longer in use., and On the other hand, only two apps have been identified whose minimum compatible version is KitKat, which means that they do not affect to roughly a 70% of the market share.

Regarding the largest version of Android that is compatible with apps or «MaxSDKVersion», insufficient data has been obtained to reflect it. This is logical since Android’s documentation recommends the following: «Warning: Declaring this attribute is not recommended».

Regarding the optimum version of Android that the apps have been designed for, «TargetSDKVersion», the data is quite revealing, as can be observed in Figure 12:  

---

**Figure 11: MinSDKVersion.**

Generally, malware creators for Android aim for their apps to be compatible with the greatest amount of devices possible, so they establish low values in «MinSDKVersion». 68.19% of the analysed applications are compatible with versions of Android that are no longer in use., and On the other hand, only two apps have been identified whose minimum compatible version is KitKat, which means that they do not affect to roughly a 70% of the market share.

Regarding the largest version of Android that is compatible with apps or «MaxSDKVersion», insufficient data has been obtained to reflect it. This is logical since Android’s documentation recommends the following: «Warning: Declaring this attribute is not recommended».

Regarding the optimum version of Android that the apps have been designed for, «TargetSDKVersion», the data is quite revealing, as can be observed in Figure 12:
34.96% of the analysed apps have «TargetSDKVersion» as obsolete Android versions. Moreover, 8.34% is focused on Froyo, a relatively old version of Android with a market share of 0.6% and that will probably disappear soon. All this leads to the idea that the samples are not very new. However, all of them have arrived on Virustotal’s analysis system for the first time in the second half of 2014, meaning they are probably quite recent, at least in terms of their identification on behalf of the system. In the «Exposure time to the threat» this aspect is analysed in depth.

5.5. THREAT EXPOSURE TIME

In this report, we have defined the threat exposure time as the time that passes since the app is created until it has been sent to VirusTotal.

In this aspect, the possible heuristics or generic detections made by antiviruses are not acknowledged; those types of detections and protections are not much extended in mobile platforms antivirus software. It is not considered either whether the samples have previously reached companies through their clients or in any other way, since it is not possible to determine the inclusion date of the corresponding signatures in their detection systems. To add values to this data, it is important to keep in mind that many of the antivirus companies use the service to feed their detection systems (signature files, cloud detection, etc.), so it can serve as an initial approximation to the elapsed time since its creation and until it is detected by a large amount of antivirus engines.
With the aim of trying to determine this time, the apps have been analysed, verifying the `classes.dex` file’s latest modification date, which as indicated previously corresponds to the source code of the compiled app. It is important to keep in mind that this date corresponds to the local date when it was compiled and therefore may have been altered by the system’s administrator, despite this not being a very common practice.

Once the app’s latest modification date has been obtained, they have been compared with the first time they were sent to Virustotal. Attending to this criteria, the following risk levels have been identified depending on the exposure time:

- Applications where the difference between the compilation date and the date of the first delivery to Virustotal is negative → Error (it indicates that the date in the system where the APK was compiled is incorrect, it cannot be determined whether this is accidental or deliberate).
- Applications where the difference between the compilation date and the date of the first delivery to Virustotal is below 1 day → Very low.
- Applications where the difference between the compilation date and the date of the first delivery to Virustotal is below 7 days → Low
- Applications where the difference between the compilation date and the date of the first delivery to Virustotal is below 31 days → Medium.
- Applications where the difference between the compilation date and the date of the first delivery to Virustotal is below 93 days → High.
- Applications where the difference between the compilation date and the date of the first delivery to Virustotal is above or equal to 93 days → Very high.

The data in Figure 13 depicts this classification:

![Figure 13: Exposure time.](image-url)
• Almost half of the apps have a very high exposure time of over 3 months.
• 13% of apps have a low or very low exposure time below 1 week.
• The system’s compilation date has been falsified in 19% of the apps (14,384).

5.6. DEVELOPMENT PLATFORM

Likewise, the system where the APKs have developed has also been verified. The obtained results are shown in Figure 14:

![Figure 14: Development platform.](image)

The immense majority of the analysed apps have been developed in Windows platform, only 2% have been developed in Linux. Seemingly, OSX is not the development platform of choice for malicious app developers.

5.7. PACKAGE NAME

Using package names of the analysed apps as a source, the most commonly used ones have been verified. This roughly allows determining whether there are especially active malware families for Android and if malware developers make slight modifications on the APKs to avoid hash detection from antiviruses. The obtained data is shown in Figure 15.
Figure 15: Package names.

Besides, the number of times that the words that form the package names have been repeated has been verified through NLTK, a library used for processing natural language, leading to the results obtained in Figure 16:

Figure 16: Most used chains in Package names.
It is logical for some of the terms to be common, since they are regularly used by convention. However, others are not common and can be used to detect malicious apps.

Likewise, the terms that give an idea of the topic of the apps have been identified, such as: showvideo, game, livewallpaper, playporn, pornow, smsreg, etc, and others which are more familiar have are identified in the article «Are some Android apps used for scams?».

5.8. EXISTING PACKAGE NAMES IN GOOGLE PLAY

As mentioned at the beginning of the document, Arxan’s «State of Mobile App Security» report reveals that 97% of the TOP 100 pay apps and 80% of the TOP 100 free apps for Android have modified versions corresponding to malware, distributed by a great number of alternative markets.

Taking the package names of the analysed apps as an origin, their existence has been verified on Google Play with the objective of checking how many of the analysed apps are probably original apps that have been modified to be malicious. There can be accidental coincidences, in other words, different apps that have the same package name. However, they are normally sufficiently characteristic for this not to happen. The obtained data can be observed in Figure 17.

Only 3% of the analysed apps are predictably apps stored in Google Play that have been downloaded and modified to be malicious.

The type of app has also been verified. In other words, what category they are belong to within Google Play. Therefore, a small approximation can be made as to which are the types of apps that are plagiarized most and are modified to include a malicious code. Apps belonging to 40 different categories have been identified, the top 10 being shown in Figure 18.
Finally, the amount of times these apps have been downloaded has been obtained with the aim of measuring the potential impact malicious apps have. The obtained data can be seen in Figure 19.

**Figure 18: Top 10 categories.**

**Figure 19: Number of downloads of Google Play.**
The potential impact of apps varies greatly: there are malicious apps with the same package name as others that are available on Google Play that have only been downloaded between 1 and 5 times and others that have been downloaded over 1,000,000,000 times, such as Google Maps and Youtube. Mainly for apps with few downloads, this fact can lead to analysing in depth whether they are malicious apps that have avoided the bouncer’s controls and are available on the market or if they are legitimate apps that have been used as a base to modify them, including malicious components and spreading them on other markets.

5.9. PERMISSIONS

Android incorporates a permissions model thanks to which the operations that an app can make on a mobile device can be limited. When it is installed, the user is informed of the permissions that it requests and have been previously established by the developer in the AndroidManifest.xml file. It is necessary to accept all the requested permissions in the case of wishing to install the app, not being possible to perform an installation with reduced functions so that only specific permissions are accepted.

This represents a risk because in many cases the developers include additional permissions which are not necessary for the functioning of the app but serve to obtain information such as the contacts list or places visited through the GPS tracker. As reflected in the report by the Agencia Española de Protección de Datos 31% of apps request excessive permissions for the functionality they offer.

With the aim of trying to improve the permissions model and for enabling the user to have a greater control over the privacy of their personal data, Google developed a native tool known as «App Ops», available from Android 4.3 onwards, which enables a user to revoke the permissions he deems necessary in an app. However, this app was deleted with Android’s 4.4.2 upgrade. It is important to note that the revocation of a permission can be the difference between the app working or not, so it is important to be especially careful in this aspect.

Permissions have different danger levels depending on the functions they allow the app to perform and are consequently classified in four «Protection Level» groups. Likewise, through this attribute, it is possible to determine which apps have access to the permission:

- **Normal**: in theory they do not represent a real threat to the device or the user. That is why they are automatically accepted. They are not shown by default when the installation is being performed, it is necessary to expand the displayed information to be able to see them. Some of the permissions located within this category are: vibrate or allowing the state of the network to be known.

- **Dangerous**: they represent a real threat to the user, such as subscribing to SMS Premium services or accessing to personal information. That is why when the installation of the app takes place, they are shown to the user and must be accepted.
Some of the permissions placed in this category are: SMS submission, SMS reception or making calls.

- **Signature**: the permission is only accessible by those apps signed with the same certificate than the app that declares the permission.

- **Signature / System**: it is like «Signature», but it can also be used by the system.

Therefore, the permissions are classified in various «permission_groups» depending on what their use is, with the following being the most notorious:

- **android.permission-group.ACCOUNTS**: it groups together the permissions that request direct access to Google’s accounts.

- **android.permission-group.COST_MONEY**: it groups together the permissions that can be used to make the user spend money without direct participation.

- **android.permission-group.LOCATION**: it groups together the permissions that grant access to the actual position of the user.

- **android.permission-group.PERSONAL_INFO**: it groups together the permissions that facilitate access to the user’s private information such as contacts, calendar events, emails, etc.

A total of 1,221,350 permissions have been identified, equating to an average of 16.1 permissions per app.

The most requested permissions by apps can be seen in Figure 20:

![Figure 20: Top 20 most requested permissions.](Google's image)
<table>
<thead>
<tr>
<th>Permission</th>
<th>Percentage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>android.permission.INTERNET</td>
<td>97%</td>
<td>Offers apps complete access to Internet.</td>
</tr>
<tr>
<td>android.permission.READ_PHONE_STATE</td>
<td>94%</td>
<td>Enables read-only access to phone state.</td>
</tr>
<tr>
<td>android.permission.WRITE_EXTERNAL_STORAGE</td>
<td>92%</td>
<td>Allows an app to write to external storage.</td>
</tr>
<tr>
<td>android.permission.ACCESS_NETWORK_STATE</td>
<td>88%</td>
<td>Allows an app to access information about networks.</td>
</tr>
<tr>
<td>android.permission.SEND_SMS</td>
<td>79%</td>
<td>Allows an app to send SMS messages.</td>
</tr>
<tr>
<td>android.permission.RECEIVE_SMS</td>
<td>73%</td>
<td>Allows an app to monitor incoming SMS messages, to record or perform processing on them.</td>
</tr>
<tr>
<td>android.permission.WAKE_LOCK</td>
<td>58%</td>
<td>Allows PowerManager WakeLocks to keep processor from sleeping or screen from dimming.</td>
</tr>
<tr>
<td>android.permission.RECEIVE_BOOT_COMPLETED</td>
<td>58%</td>
<td>Allows an app to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting.</td>
</tr>
<tr>
<td>Permission</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>com.android.launcher.permission.INSTALL_SHORTCUT</td>
<td>Allows an app to install a shortcut in Launcher. signatureOrSystem</td>
<td></td>
</tr>
<tr>
<td>android.permission.READ_SMS</td>
<td>Allows an app to read SMS messages. Dangerous</td>
<td></td>
</tr>
<tr>
<td>android.permission.CALL_PHONE</td>
<td>Allows an app to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed. Dangerous</td>
<td></td>
</tr>
<tr>
<td>android.permission.GET_TASKS</td>
<td>Allows an app to obtain information about its current or recent execution tasks. Dangerous</td>
<td></td>
</tr>
<tr>
<td>android.permission.ACCESS_COARSE_LOCATION</td>
<td>Allows an app to access approximate location derived from network location sources such as cell towers and Wi-Fi. Dangerous</td>
<td></td>
</tr>
<tr>
<td>android.permission.ACCESS_FINE_LOCATION</td>
<td>Allows an app to access precise location from location sources such as GPS, cell towers and Wi-Fi. Dangerous</td>
<td></td>
</tr>
<tr>
<td>android.permission.ACCESS_WIFI_STATE</td>
<td>Allows apps to access information about Wi-Fi networks. Normal</td>
<td></td>
</tr>
</tbody>
</table>
Based on the statistics, the following conclusions regarding the used permissions are obtained:

- 99.93% of the apps request at least one permission catalogued as dangerous.
- 97% request unlimited access to Internet.
- 94% request access to sensitive information.
- At least 79% can lead to additional costs in a user’s bill. Keeping in mind that, as reported by SMS PREMIUM services, each received SMS costs approximately 1.45€ «Max 25 SMS/month», this could result in roughly 36.25€ per month and user. Accordingly, the analysed apps can generate an economical benefit of over 2 million Euros a month, without including other aspects such as selling confidential information, etc.
- 58% are automatically loaded when the system is switched on.
Among the 20 most requested permissions, 15% correspond to permissions with «normal» «ProtectionLevel», 75% to «dangerous» and 10% to «signatureOrSystem». That is why it is very important to focus on the permissions that an applications requests when it is installed, as it could be quite significant. However, this control falls upon the user and therefore mainly represents a risk because of the credulity, confidence and lack of awareness of the vast majority of users.

The new permissions system in Google Play notably increases the risk derived from accepting them. With this system, when the upgrade is performed, the user implicitly accepts permissions that belong to the same group as the permissions that have been accepted before, without any interaction being required. Moreover, details are not given when the installation is made from Google Play. Instead, a generic explanation of the group they belong to is shown.

Therefore, the validity of the permissions have been verified, comparing them with the ones indicated in Android’s documentation with the aim of making an approximation to the poorly written permissions, which mainly occur because of an incorrect syntax, and which are customized.

- 5.19% of all the permissions declared correspond to customized or bad declared permissions.
- 43.49% of the analysed apps contain customized or bad declared permissions.
- 13.7% of these permissions are directly related to the Google Cloud Messaging for Android service through which a developer can establish bidirectional communication between a server and the app’s client.

### 5.10. SERVICES

Services are components of an app that can perform long duration operations on a secondary level without the need for user interaction. A total of 201,826 services have been identified, which equates to an average of 2.66 per application. Figure 21 shows the most frequently declared services:
5.11. RECEIVERS

Receivers are components of an app that enable the reception of actions that are sent through the system or other apps, even when other components in the app are not working. Figure 22 shows the percentage of analysed apps that include this type of mechanisms.

A total of 228,951 receivers have been identified, which equates to an average of 3.01 receivers per app. Among those, those related to received text messages stand out, which are normally directly related to subscriptions to SMS Premium services. For these, the process is as follows:
1. A user installs an app, previously accepting the requested permissions, one of them being text messages delivery.
2. Without the user’s knowledge, the app sends a subscription message to an SMS Premium service.
3. These types of services require a confirmation code that the user also receives through a text message.
4. A receiver that is explicitly designed for this situation identifies the reception code, reads it, deletes the message and sends a confirmation message to the service. Therefore, a user is subscribed to a special service without giving his consent.

5.12. PROVIDERS

Providers are components of an app that enable sharing information with other apps. Figure 23 shows the percentage of apps that incorporate these mechanisms.

![Figure 23: Percentage of the analysed apps that use providers.](image)

A total of 2,010 providers have been identified, equating to an average of 0.03, in other words, its use is not too extended in malicious apps. This is reasonable, since it does not tend to be necessary for these types of apps to implement mechanisms to share their information. Instead, in some cases, they look to exploit an incorrect configuration in other apps to obtain sensitive information.

5.13. ACTIVITIES

Activities correspond to the user interface of an app, so that the app can interact with the user.

0.03% of the analysed apps do not include any activity, in other words, they do not have a graphic interface. It is not too clear if this fact enables users to realise that something is not right with the app, or if on the contrary it makes them think that it has not been installed...
correctly and they do not give it importance as occurs with malware for other computing environments such as Windows. Of the remaining apps, 99.7% include a total of 409,819 activities, which equates to an average of 5.39 activities per app.

5.14. COMPLEXITY

The complexity of an APK is calculated from the number of permissions, services, receivers, providers, activities and ads that the APKs have.

The following ranges have been used for the calculation:

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0 – 10</td>
</tr>
<tr>
<td>Medium</td>
<td>11 – 25</td>
</tr>
<tr>
<td>High</td>
<td>26 – 50</td>
</tr>
<tr>
<td>Very high</td>
<td>&gt; 50</td>
</tr>
</tbody>
</table>

The obtained information obtained, classified according to the indicated ranges, is shown in Figure 24.
8.11% of the apps have a low complexity.
52.89% of the apps have a medium complexity.
29.81% of the apps have a high complexity.
9.19% of the apps have a very high complexity.

5.15. DANGEROUSNESS

Androguard implements a series of methods that enable to establish the dangerousness levels of an APK. For this, it takes as reference various aspects such as the permission types requested by the app. The information shown in Figure 25 has been obtained from the analysed apps.
Figure 25: Danger levels of the analysed apps.

Less than 1% of the malicious apps analysed have a danger level below 50%

5.16. DEFAULT LANGUAGE

The `strings.xml` file contains the text chains that are used by an app. The default language of each of the analysed apps is identified parting from the aforementioned file after processing it to exclude its characteristic tags and using Python’s `guessLanguage` library. The results obtained are depicted in Figure 26.
The most common languages are English, Chinese and Russian, which is logical given that they are the target audience with the largest market share.

Usually, it is not possible to determine the language of an app if the `strings.xml` file does not contain many literals or if the text chains are directly defined in a static way through the properties of the app’s elements and not through the file. That is the reason why it was not possible to establish the language for 30.19% of the analysed apps.

**5.17. ALTERNATIVE LANGUAGES**

The data shown in Figure 27 has been obtained by extracting the values corresponding to the languages established in the apps from the app’s resources. This is done by analysing and comparing them with those included in ISO 639-1 which harvests the main languages in the world and enables them to be identified through two letters.
Figure 27: Multi-language support in the analysed apps.

Of the 11% of apps with alternative languages, the most commonly used languages are shown in Figure 28.

Figure 28: Alternative languages.

The data in Figure 29 is obtained by grouping together the APKs by the number of alternative languages they include.
74.09% of the multi-language apps have only one alternative language and 14.48% have over 20 languages configured.

5.18. MOST USED WORDS

Following a similar procedure to the ones used in the «Package name», the most used words have been obtained from the chains of the strings.xml file, once it has been processed to exclude the characteristic tags from its definition. Therefore, the punctuation, exclamation, interrogation marks, articles, prepositions, etc, are excluded. The most used words can be seen in Figure 30.
5.19. ESTABLISHED CONNECTIONS

From the URLs associated with the connections established by the apps, the location for their corresponding domains have been verified, obtaining the data from Figure 31 and Figure 32.

Figure 30: Most used words.

Figure 31: World map of the connections established by malicious apps.
Likewise, which domains/subdomains the malicious apps establish most connections with has been verified, the results of which appear in Figure 33.

It is important to keep in mind that the fact that a malicious app established a connection with a URL does not imply that this URL is malicious: in many cases they correspond to the
previously mentioned monetization services. That is why the domains of the obtained URLs have crossed with different reputation lists.

The complete URLs have not been verified because, usually, they include parameters that vary in each case and therefore it is very complicated to classify them as malware. In most cases, the domains were not classified, but in some cases they were identified as malicious, as shown in Figure 34.

5.20. ANTIVIRUS DETECTIONS

The number of antiviruses that detect each of the samples, out of a total of 55 engines, has been verified (the complete list can be seen in «ANNEX 2 engines»).

The results obtained are displayed in Figure 35.
There are only 6 samples from 76,000 that are detected by 10 antiviruses or less, which confirms the reliability of the sampling selected for the report.

5.21. TYPES OF MALWARE

Using the detections made by the engines selected for the acquisition of the samples analysed for the report and keeping in mind the format of the naming convention of the alias assigned to each of them, the most extended types of Android malware have been verified, obtaining the data in Figure 36.
Like in other platforms, such as Windows, the most extended type of malware are Trojans.

Concerning the most active malware families, the classification illustrated in Figure 37 has been obtained.

Figure 37: Most active malware families.
The most active families are those corresponding to FakeInst, in other words false installers, corresponding to supposed installers of other apps, but which are apps that subscribe users to SMS premium services.

5.22. PRIVACY

All APKs methods have been analysed and compared with a list of potentially dangerous methods, available in «ANNEX 3 – of methods that affect privacy», and that can directly threaten a user’s privacy. Following is the obtained data:

- 46,083 apps with codes that directly affect the privacy of users that install it, which equates to 60.64% of the analysed apps, have been identified.
- 181,305 methods that affect the privacy of users, which equates to an average of 2.38 methods per app, have been identified.

Figure 38 illustrates the most repeated potentially dangerous methods.

Many of these methods are directly related to the monetization libraries, given that they are used to obtain the necessary information to customize advertising on devices.
CONCLUSIONS

As the data reflects, Android malware continues to grow day by day and a trend change is not anticipated in the upcoming years. Instead the exact opposite is most likely to occur.

An irrefutable fact is that, as happened in developments for other types of platforms such as Windows, Android malware has acquired a very important level of sophistication, which denotes that the level of professionalization of this type of service is high, mainly as a result of the great economic benefit that their developers obtain with a very small investment.

Despite this notable increase and the achieved level of satisfaction, users are not aware of the dangers they are exposed to and, on many occasions, do not take the necessary measures to protect themselves. It must be noted that security starts with oneself and that knowledge and common sense are the best protection measures.

With relation to the study, a great amount of significant data has been obtained, the most outstanding data being illustrated in Figure 39.
Figure 39: Some of the relevant data from the study.
• Android malware developers prefer **Windows as a development platform** as demonstrated by the fact that 98% of the analysed apps have been compiled in this operating system.

• 99.98% of the apps are **self-signed**, an aspect that is common to legitimate apps. This means that their legitimacy is not backed by a certification authority or CA, something which is completely logical in the case of malicious apps, but which should be worked on to add another security layer to apps and therefore to users of the platform. It is important to find a balance between effectiveness and security, necessarily pondering whether to apply restrictive policies on occasions.

• The apps possess 16.1 permissions on average, some of the most significant data being the following:
  o 99.93% requested at least one permission catalogued as **dangerous**.
  o 97% request indiscriminate access to internet, which they make the most of to **exfiltrate information**.
  o 94% request **access to sensitive information**.
  o At least 79% of the apps have permissions that derive in **additional costs in the user’s bill**. This could lead to an economic benefit of over 2 million Euros a month, without keeping in mind other aspects such as the sale of confidential information, etc.

• It can be determined that 39% of the apps have a **high or very high complexity** based on the amount of permissions, services, receivers, providers, activities and ads.

• From the languages included in the apps, it can be concluded that they are mainly geared towards **Chinese, Russian and English-speaking markets**.

• They establish connections mainly with **China** and the **United States**.

• They possess and average of 2.38 methods per application that affect the privacy of users.

• Evaluating the impact that they have on users that install them, 99.72% of the apps have a **medium / high dangerousness level**.

• Keeping in mind the number of downloads of apps with package names in Google Play, it seems that Android malware creators indiscriminately select which apps to clone and modify adding malicious functions or, at least, they do not limit themselves exclusively to those that have a high ratio of downloads.

Against this situation, it is necessary for companies with antivirus solutions in this case for the Android operating system, to make a notable effort and implement mechanisms that can offer users a greater level of security.

Using the information obtained by this study as reference, it is possible to improve the systems responsible for performing the identification and categorization of apps that are potentially dangerous or malicious.
With all the obtained information, signatures that are based on vast criteria can be generated: package names, names of activity components, services, receivers, providers, hashes from app components, established connections, a combination of the above, etc., or even use other study routes such as hashes from the app’s dangerous methods, signatures based on the AndroidManifest.xml file, etc.

This way, generic detections can be made for different malware families and thereby identify new variants that appear. As shown, and as happened on other platforms such as Windows, developers use «constructor» generators on many occasions which slightly alter the apps with the purpose of avoiding signature detection systems, which means that implementing these signatures would be very useful.

On the other hand, it is necessary to take measures like those taken for Android One, for which Google assumes the responsibility of sending upgrades directly to mobile devices without the need for the mobile manufacturers to interact, thereby ensuring that the spectrum of terminals that include it do not suffer the important issue of fragmentation.

It is important to keep in mind that it is the responsibility of all the implied agents to take the convenient measures to reduce the threats that we are exposed to on a daily basis.
## ANNEX 1 – COMPARISON OF MONETIZATION LIBRARIES

<table>
<thead>
<tr>
<th>Feature</th>
<th>waps</th>
<th>admob</th>
<th>applo</th>
<th>domob</th>
<th>youmi</th>
<th>smartmad</th>
<th>vpon</th>
<th>adsmogo</th>
<th>airad</th>
<th>airpush</th>
</tr>
</thead>
<tbody>
<tr>
<td>internet</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>access_network_state</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>read_phone_state</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>write_external_storage</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>access_coarse_location</td>
<td>X</td>
<td></td>
<td></td>
<td>O</td>
<td></td>
<td>X</td>
<td>O</td>
<td>X</td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>access_wifi_state</td>
<td>X</td>
<td></td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>vibrate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>install_shortcut</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>access_fine_location</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>O</td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>camera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>record_audio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>send_sms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>call_phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read_calendar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>write_calendar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read_external_storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get_tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mountUnmount_filesystems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>system_alert_window</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read_logs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>write_settings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>change_configuration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>broadcast_sticky</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>receive_boot_completed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read_sms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read_contacts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(X) Compulsory (O) Optional
<table>
<thead>
<tr>
<th>Antivirus Engine</th>
<th>Web</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-Aware</td>
<td><a href="http://lavasoft.com">http://lavasoft.com</a></td>
<td>Canada</td>
</tr>
<tr>
<td>AegisLab</td>
<td><a href="http://www.aegislab.com">http://www.aegislab.com</a></td>
<td>Taiwan</td>
</tr>
<tr>
<td>Agnitum</td>
<td><a href="http://www.agnitum.com">http://www.agnitum.com</a></td>
<td>Russia</td>
</tr>
<tr>
<td>AhnLab-V3</td>
<td><a href="http://www.ahnlab.com">http://www.ahnlab.com</a></td>
<td>Korea</td>
</tr>
<tr>
<td>Antiy-AVL</td>
<td><a href="http://www.antiy.net">http://www.antiy.net</a></td>
<td>China</td>
</tr>
<tr>
<td>Avast</td>
<td><a href="http://www.avast.com">http://www.avast.com</a></td>
<td>Czech Republic</td>
</tr>
<tr>
<td>AVG</td>
<td><a href="http://www.avg.com">http://www.avg.com</a></td>
<td>Czech Republic</td>
</tr>
<tr>
<td>Avira</td>
<td><a href="http://www.avira.com">http://www.avira.com</a></td>
<td>Germany</td>
</tr>
<tr>
<td>AVware</td>
<td><a href="http://www.avware.com">http://www.avware.com</a></td>
<td>Brazil</td>
</tr>
<tr>
<td>Baidu-International</td>
<td><a href="http://antivirus.baidu.com">http://antivirus.baidu.com</a></td>
<td>China</td>
</tr>
<tr>
<td>BitDefender</td>
<td><a href="http://www.bitdefender.com">http://www.bitdefender.com</a></td>
<td>Romania</td>
</tr>
<tr>
<td>Bkav</td>
<td><a href="http://www.bkav.com">http://www.bkav.com</a></td>
<td>Vietnam</td>
</tr>
<tr>
<td>ByteHero</td>
<td><a href="http://www.bytehero.com">http://www.bytehero.com</a></td>
<td>China</td>
</tr>
<tr>
<td>CAT-QuickHeal</td>
<td><a href="http://www.quickheal.com">http://www.quickheal.com</a></td>
<td>India</td>
</tr>
<tr>
<td>ClamAV</td>
<td><a href="http://www.clamav.net">http://www.clamav.net</a></td>
<td>USA</td>
</tr>
<tr>
<td>CMC</td>
<td><a href="http://www3.cmcinfosec.com">http://www3.cmcinfosec.com</a></td>
<td>Vietnam</td>
</tr>
<tr>
<td>Comodo</td>
<td><a href="http://www.comodo.com">http://www.comodo.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>Commtouch</td>
<td><a href="http://www.cyren.com">http://www.cyren.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>DrWeb</td>
<td><a href="http://www.drweb.com">http://www.drweb.com</a></td>
<td>Russia</td>
</tr>
<tr>
<td>Emsisoft</td>
<td><a href="http://www.emsisoft.com">http://www.emsisoft.com</a></td>
<td>New Zealand</td>
</tr>
<tr>
<td>ESET-NOD32</td>
<td><a href="http://www.eset.com">http://www.eset.com</a></td>
<td>Slovakia</td>
</tr>
<tr>
<td>Fortinet</td>
<td><a href="http://www.fortinet.com">http://www.fortinet.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>F-Prot</td>
<td><a href="http://www.f-prot.com">http://www.f-prot.com</a></td>
<td>Iceland</td>
</tr>
<tr>
<td>F-Secure</td>
<td><a href="http://www.f-secure.com">http://www.f-secure.com</a></td>
<td>Finland</td>
</tr>
<tr>
<td>GData</td>
<td><a href="http://www.gdata.de">http://www.gdata.de</a></td>
<td>Germany</td>
</tr>
<tr>
<td>Ikarus</td>
<td><a href="http://www.ikarus.at">http://www.ikarus.at</a></td>
<td>Austria</td>
</tr>
<tr>
<td>Jiangmin</td>
<td><a href="http://www.jiangmin.com">http://www.jiangmin.com</a></td>
<td>China</td>
</tr>
<tr>
<td>K7AntiVirus</td>
<td><a href="http://www.k7computing.com">http://www.k7computing.com</a></td>
<td>India</td>
</tr>
<tr>
<td>K7GW</td>
<td><a href="http://www.k7computing.com">http://www.k7computing.com</a></td>
<td>India</td>
</tr>
<tr>
<td>Kaspersky</td>
<td><a href="http://www.kaspersky.com">http://www.kaspersky.com</a></td>
<td>Russia</td>
</tr>
<tr>
<td>Kingsoft</td>
<td><a href="http://www.kingsoft.com">http://www.kingsoft.com</a></td>
<td>China</td>
</tr>
<tr>
<td>Malwarebytes</td>
<td><a href="http://www.malwarebytes.org">http://www.malwarebytes.org</a></td>
<td>USA</td>
</tr>
<tr>
<td>McAfee</td>
<td><a href="http://www.mcafee.com">http://www.mcafee.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>McAfee-GW-Edition</td>
<td><a href="http://www.mcafee.com">http://www.mcafee.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>Microsoft</td>
<td><a href="http://windows.microsoft.com/MSE">http://windows.microsoft.com/MSE</a></td>
<td>USA</td>
</tr>
<tr>
<td>MicroWorld-eScan</td>
<td><a href="http://www.escanav.com">http://www.escanav.com</a></td>
<td>India</td>
</tr>
<tr>
<td>NANO-Antivirus</td>
<td><a href="http://www.nanoav.ru">http://www.nanoav.ru</a></td>
<td>Russia</td>
</tr>
<tr>
<td>Antivirus Name</td>
<td>Website URL</td>
<td>Country</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Norman</td>
<td><a href="http://www.norman.com">http://www.norman.com</a></td>
<td>Norway</td>
</tr>
<tr>
<td>nProtect</td>
<td><a href="http://www.nprotect.com">http://www.nprotect.com</a></td>
<td>Korea</td>
</tr>
<tr>
<td>Panda</td>
<td><a href="http://www.cloudfantivirus.com">http://www.cloudfantivirus.com</a></td>
<td>Spain</td>
</tr>
<tr>
<td>Qihoo-360</td>
<td><a href="http://360safe.com">http://360safe.com</a></td>
<td>China</td>
</tr>
<tr>
<td>Rising</td>
<td><a href="http://www.rising.com.cn">http://www.rising.com.cn</a></td>
<td>China</td>
</tr>
<tr>
<td>Sophos</td>
<td><a href="http://www.sophos.com">http://www.sophos.com</a></td>
<td>UK</td>
</tr>
<tr>
<td>SUPERAntiSpyware</td>
<td><a href="http://www.superantispyware.com">http://www.superantispyware.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>Symantec</td>
<td><a href="http://www.symantec.com">http://www.symantec.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>Tencent</td>
<td><a href="http://guanjia.qq.com">http://guanjia.qq.com</a></td>
<td>China</td>
</tr>
<tr>
<td>TheHacker</td>
<td><a href="http://www.hacksoft.com.pe">http://www.hacksoft.com.pe</a></td>
<td>Peru</td>
</tr>
<tr>
<td>TotalDefense</td>
<td><a href="http://www.totaldefense.com">http://www.totaldefense.com</a></td>
<td>USA</td>
</tr>
<tr>
<td>TrendMicro</td>
<td><a href="http://www.trendmicro.com">http://www.trendmicro.com</a></td>
<td>Japan</td>
</tr>
<tr>
<td>TrendMicro-HouseCall</td>
<td><a href="http://www.trendmicro.com">http://www.trendmicro.com</a></td>
<td>Japan</td>
</tr>
<tr>
<td>VBA32</td>
<td><a href="http://anti-virus.by">http://anti-virus.by</a></td>
<td>Belarus</td>
</tr>
<tr>
<td>VIPRE</td>
<td><a href="http://www.vipreantivirus.com">http://www.vipreantivirus.com</a></td>
<td>Malta</td>
</tr>
<tr>
<td>ViRobot</td>
<td><a href="http://www.hauri.net">http://www.hauri.net</a></td>
<td>Korea</td>
</tr>
<tr>
<td>Zillya</td>
<td><a href="http://zillya.com">http://zillya.com</a></td>
<td>Ukraine</td>
</tr>
<tr>
<td>Zoner</td>
<td><a href="http://www.zonerantivirus.com">http://www.zonerantivirus.com</a></td>
<td>Czech Republic</td>
</tr>
</tbody>
</table>
ANNEX 3 – LIST OF METHODS THAT AFFECT PRIVACY

getDeviceId
getLine1Number
getDeviceSoftwareVersion
getNetworkOperator
getNetworkOperatorName
getSimSerialNumber
getActiveNetworkInfo
getNetworkPreference
getDisplayLanguage
getDisplayCountry
getSubscriberId
getLongitude
getLatitude
getCellLocation
getPhoneType
getAccounts
setPassword
getNetworkInfo
getAllNetworkInfo
getPackageName
getApiKey
getManufacturer
getAboutMe
getBirthday
getCircledByCount
getCover
getCurrentLocation
getDisplayName
getGender
getImage
getld
getLanguage
getNickname
getOrganization
getPlacesLived
getPlusOneCount
getRelationshipStatus
getTagline
getUrl
getUserName
getPostalCode
LIST OF FIGURES

Figure 1: Android architecture. ................................................................. 9
Figure 2: Android incorporates multiple controls to protect its users. Source: Adrian Ludwig – Android’s Chief of Security......................................................... 11
Figure 3: Structure of an APK file. .............................................................. 12
Figure 4: Icons of some of the analysed applications. ................................ 14
Figure 5: Top 10 types of files. ................................................................ 15
Figure 6: Free apps with advertising model. ............................................. 17
Figure 7: Example of advertising in Android. ........................................... 18
Figure 8: Monetization. .............................................................................. 19
Figure 9: Top 20 versions of monetization libraries. ............................... 21
Figure 10: Distribution of Android’s versions. .......................................... 23
Figure 11: MinSDKVersion. ........................................................................ 24
Figure 12: TargetSDKVersion. .................................................................. 25
Figure 13: Exposure time. ......................................................................... 26
Figure 14: Development platform.............................................................. 27
Figure 15: Package names. ........................................................................ 28
Figure 16: Most used chains in Package names........................................ 28
Figure 17: Existing Package Names in Google Play. ............................... 29
Figure 18: Top 10 categories....................................................................... 30
Figure 19: Number of downloads of Google Play. .................................... 30

Android malware situation
Figure 20: Top 20 most requested permissions................................................................. 32
Figure 21: Most declared services in the analysed apps.............................................. 37
Figure 22: Percentage of analysed apps that contain receivers............................... 37
Figure 23: Percentage of the analysed apps that use providers............................... 38
Figure 24: Complexity of the analysed apps............................................................... 40
Figure 25: Danger levels of the analysed apps............................................................. 41
Figure 26: Default languages of the analysed apps...................................................... 42
Figure 27: Multi-language support in the analysed apps............................................ 43
Figure 28: Alternative languages.................................................................................. 43
Figure 29: Amount of alternative languages............................................................... 44
Figure 30: Most used words.......................................................................................... 45
Figure 31: World map of the connections established by malicious apps............... 45
Figure 32: Location of domains................................................................................... 46
Figure 33: Domains / Subdomains............................................................................. 46
Figure 34: Analysed domains...................................................................................... 47
Figure 35: Number of APK detections......................................................................... 48
Figure 36: Types of malware...................................................................................... 49
Figure 37: Most active malware families................................................................. 49
Figure 38: Potentially dangerous methods............................................................... 50
Figure 39: Some of the relevant data from the study.................................................. 52
REFERENCES


