TFM

Study on privacy of parental control mobile applications

Student: Feal Fajardo, Álvaro

Supervisor: Carro, Manuel

Co-Supervisor: Troncoso, Carmela

Máster Universitario en Software y Sistemas

Universidad Politécnica de Madrid

IMDEA Software Institute

Madrid - June 28, 2017

Abstract

Parental control applications are one kind of mobile software programs, which are used by parents to monitor and control the use that their kids make of their cell-phone. Parents install these type of apps on their children's phones in order to remotely set rules for what the children can do with their device and to monitor where the phone is and what their kids are using it for.

These type of applications are highly intrusive, because they gather all kinds of data that reflect private information about the users, such as their Internet history, text messages, calls, location... Furthermore, these data are often sent to servers hosted in the Internet, where the information is gathered in order to let the parent access it later from a different device than their child's phone.

Therefore, these systems make it possible (willingly or not) for third parties to access all or parts of these private data. From a privacy stand point, these applications pose a great threat to users. However, there has not yet been a study on the amount and kind of information that these apps gather and the security of their communications.

In this thesis, we study how parental control apps behave, and the features that they provide the user with, and we conduct several studies to understand their privacy implications. First, we research how well these applications explain their behavior and operation to their users. Second we gather information about the permissions that these apps request and compare them to their behavior. Third we study what information is gathered and later sent to Internet servers by these programs. Finally, we investigate if this information is sent securely.

We studied fourteen different parental control applications, anyhow we only could perform all of the studies explained above in seven of them. In each one of those seven apps we find at least one of these privacy issues: the sending of sensitive information to third parties, the leakage of private data from before the parental control app installation, the sending of private information before the user agrees with the term of usage (sometimes even the fact that the user never agrees to it), or the sending of private data via an insecure communication channel. We also find that 50% of the fourteen studied applications did not clearly explain to the user that their children's data was being sent through the Internet and stored in servers. Finally, we categorize 15% of the requested permissions in eleven of the fourteen applications as confusing and probably unnecessary.

Resumen

Las aplicaciones de control parental son un tipo de programas software para teléfonos móviles, usados por los padres para motorizar y controlar el uso que hacen sus hijos de sus teléfonos. Los padres instalan este tipo de apps en los teléfonos de los hijos, pudiendo así establecer reglas para establecer el uso que pueden hacer los hijos de sus teléfonos y monitorizar de manera remota la localización del teléfono y qué uso están haciendo los niños su teléfono.

Este tipo de aplicaciones son altamente intrusivas, ya que recogen una gran cantidad de datos que reflejan información privada sobre los usuarios, como su historial web, mensajes de texto, llamadas, localización... Además, estos datos suelen ser enviados a servidores localizados en Internet donde la información se guarda para que los padres puedan acceder después a esta información de manera remota, desde un dispositivo distinto al teléfono de su hijo.

Por lo tanto, estos sistemas comprometen (de manera intencionada o no) esta información, de modo que otros agentes podrían acceder a todos o parte de estos datos privados. Desde el punto de vista de la privacidad, estas aplicaciones representan una gran amenaza para los usuarios. Sin embargo, todavía no ha habido ningún estudio sobre la información que dichas aplicaciones recogen y la seguridad de las mismas.

En esta tesis estudiamos como se comportan este tipo de programas y las funcionalidades que le presentan al usuario. También realizamos diferentes estudios para conocer las implicaciones de privacidad de las aplicaciones de control parental. Primero, analizamos cómo de claro explican estas aplicaciones su modelo de funcionamiento, segundo recogemos información sobre los permisos que solicitan las aplicaciones que analizamos, tercero estudiamos qué información es recolectada y enviada a servidores de Internet por parte de estos programas y cuarto, investigamos si esta información es enviada de una forma segura.

Estudiamos catorce aplicaciones de control parental, en el caso de siete de estas realizando cada uno de los estudios arriba explicados. En cada una de estas siete aplicaciones descubrimos al menos uno de los siguientes problemas de privacidad: el envío de información especialmente sensible a terceros, el filtrado de datos privados anteriores a la instalación de la aplicación de control parental, el envío de información privada antes de que el usuario acepte los términos de uso (incluso el hecho de que el usuario nunca llegue a aceptar este acuerdo) y el envío de datos confidenciales a través de canales de comunicación inseguros. También encontramos que el 50% de las catorce aplicaciones analizadas no explican de una manera clara para el usuario que los datos de su hijo son enviados a través de Internet y guardados en servidores. Por último, categorizamos el 15% de los permisos solicitados por once de las catorce aplicaciones del estudio como confusos y probablemente innecesarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Parental control apps are one kind of mobile applications that are installed by parents on their kids' phone and that allow them to monitor what the kids are doing with their phone, and to dictate what their children can and can not do. These kind of applications gather several different types of data about the phone usage, such as visited webpages, contacts, location, message logs…This information is sent through the Internet to the developer's servers, where it is stored in order to allow the parent to access this info from different devices than the kid's mobile. By definition, parental control apps are, from a privacy standpoint, very intrusive applications.

These programs access and store a lot of private information about the user, which can be a threat to the user's privacy. There has been some previous work [2, 3, 4] showing how this type of information can be used to perform different privacy attacks on the user. Web history logs can be used to identify personal information about the user (such as gender, age and location) [2]. Even web browser developers had to add defenses to avoid the leakage of visited webpages. Anyhow it is still feasible to perform different type of attacks on an user, even though these take more time and gather less information than before [3]. Another dangerous piece of data is location of an user, there is previous works that shows how the visited locations of users can be used to discover, in most cases, where their home is and in a smaller number of cases the identity of the victim [4].

These apps are usually installed by parents on the kid's phone. This means that the installing user is not the one compromising its private information, but rather compromising other person's info. Furthermore, it must be noted that in many cases, children's information reflects information about the parents themselves. This leads to another aspect about these applications, which is how stealthy they are, meaning that it is important that when these programs are running on a phone, they should always advise the user of the fact that its private data is being recollected. In the case that these apps could be installed without the final user knowing, they would represent a great threat towards the person's privacy and human rights. The fact that they could be installed in stealth mode, could also make these type of applications prone to be used as a tool for domestic violence. In the following research[5] NPR surveyed 72 domestic violence shelters in the U.S. about cyberstalking. They found that in 85% of the shelters, victims were tracked by their abuser and the authors also show that in 75% of the participating centers, the employees were working with victims whose abuser eavesdropped their cellphone communications.

Nevertheless, there is not a good understanding of these programs, it seems like users do not know much about the architecture of these apps, what kind of information is gathered, how the developers treat this info, and where it goes.

For all these reasons, we settle to make a study to answer the questions presented above. In this thesis we study 14 apps from the top 100 "parental control apps" in Google Play and perform different experiments to understand the following aspects of those applications: first, how well do developers explain the app's behavior and work model; second, what permissions do these apps request; third, what information is gathered and later sent to Internet servers by these applications and fourth, whether the communications between these software programs and the Internet servers are secure. We find that each of the seven applications that we can fully test have one of the following privacy issues: sending of information to third parties, leakage of private data from before the parental control app installation, sending of private information before the user agrees with the term of usage and/or sending of private data via an insecure communication channel. We also see that 50% of fourteen the applications do not explain clearly their work model, and that 15% of the requested permissions by the studied applications is not clearly linked to any feature in the program.

1.1 Objectives

- Understand the features and behaviors of parental control apps.
- Investigate the potential differences in the understanding of the work model of this kind of programs between mid-expert and non-expert users.
- Study the set of permissions that each app requires and map them to the features that they provide.
- Analyze the privacy risks and threats that these apps pose.

Chapter 2

Preliminaries

2.1 Privacy and use of monitoring apps

Security can be defined as the fulfillment of several desirable properties, such as confidentiality, integrity, availability, non-repudiation, authentication... One of the properties under the computer security definition is privacy. Privacy can be defined as the ability of an individual or group of people to keep its personal information to itself, away from public knowledge and from undesired people. In 1970, Westin defined privacy as "The right of the individual to decide what information about himself should be communicated to others and under what circumstances". In other words, it means having control over your own information.

Children can access a lot of inappropriate content on the internet, such as violence, guns, sexual images, strong language, drugs, gambling... Parent's are concerned about this, but also about the possibility of their kids engaging in relationships with internet strangers, who can be dangerous individuals trying to take advantage kids.

According to one study conducted in 2010 in 25 EU countries, 93% of children (age 9-16) went online weekly, 33% of these kids accessed the Internet via mobile device. 41% of the kids that went online weekly had come across one of the risks that the study classified [6]. This risk were: pornography, bullying, receiving sexual messages, contact with people not known face-to-face, offline meetings with online contacts, potentially harmful user-generated content and personal data misuse. In a different study conducted by NHC in 2005 between youngster of age between 11-19, 20% had been victims of cyberbullying , 73% of those by a known person and 26% by a stranger [7].

However, these applications create an ethic debate about parent's control over their children private data. Some people say that parents should have full control over their kids cellphone usage information. Others argue that the best way to go about this issue is instructing kids on the risks of internet and cellphones, and let them learn to behave in a safe manner by themselves. According to these distinct visions of the issue, we can find different type of parental control applications. First, those that let the parents passively control their kids, alas blocking harmful content without notifying the parent. The other type of applications are active, and allow the parent to log the kid's cellphone usage.

These different ways of thinking are similar to another ongoing debate about the coexistence of security and privacy. Some experts claim that the level of security and privacy are correlated, and that in order to achieve a higher degree of security, privacy must be compromised. This thinking is perfectly described by government surveillance. But this poses some problems which are often used as discussion points by experts that claim that privacy should not be sacrificed for the sake of security[8]. First, surveillance its not always effective, a smart adversary may dodge it, by using codes, secure channels or some other techniques. Also, this model is based on the trust that must be put into the supervisor of the data, and the premise that this party will not abuse any of the knowledge acquired.

These problems translate to the parental control applications as well, creating a debate about whether the functionalities of these apps overcome the risks. It's a big risk to send all this private information from the child's phone to the developer's web servers because it could be compromised, let alone used by malicious developers for their own benefit. Parents must understand these problems upon deciding if they want to use these kind of apps, to be able to decide if it is worth to use these kind of applications. This is why we believe it's important to analyze these kind of applications, in order to help document the architecture of this programs so users can better understand what risks they might be exposed to.

2.2 Android

All the applications that we study in this work are tested in an Android emulator. Android is an operating system, developed by Google bases on the Linux Kernel. This operating system is designed for touchscreen devices, such as smartphones, tablets, smart-watches and wearables...In the 4th quarter of the year 2015, 79.6% of mobile phones had Android as its operating system [9].

These phones are based on the use of applications, which are programs that provide the user with different functionalities. These apps are available for download in the Google Play Store, which in march 2017 hosted around 2.8 million apps [10].

There are lots of different releases of the Android operating system, being Android Nougat 7.1 the latest developed and launched release. In Table 2.1 and Figure 2.1 we can see the distribution between android versions (only showing distributions greater than 0.1%). In this thesis all of the applicatons are tested using Android 4.1, because our setup requires the use of tools that work only on certain Android versions (being 4.3 the higher version that we could have used).

Table 2.1: Distribution of android versions as of May 2017 [1]

Version	Code name	Distribution
2.3.3-2.3.7	Gingerbread	1.0%
4.03.4.0.4	Ice Cream Sandwich	0.8%
4.1.x-4.2.x-4.3	JellyBean	9.1%
4.4	KitKat	18.8%
5.0-5.1	Lollipop	32.0%
6.0	Marshmallow	31.2%
7.0-7.1	Nougat	7.1%

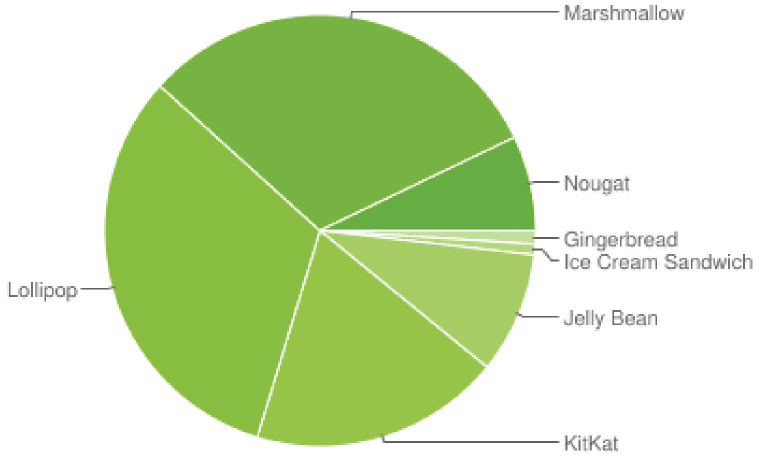


Figure 2.1: Distribution of android versions as of May 2017 [1]

2.2.1 Permissions

Android uses this feature to inform the user of the actions that they will allow an application to perform, such as accessing the phone's location, taking pictures, reading content form internal storage and many more…

In cellphones with an android version prior to Android 6.0, users are warned about the permissions that they are granting before installing the application, then they can decide if they want to cancel the installation or proceed with it. Since Android 6.0, permissions are granted on runtime. This means that when one app performs an action that requires a certain permission for the first time, the user will be prompt with the choice of granting or declining the permission. One important consequence of this policy change is that permissions can be granted individually. However some specific set of permissions may still be required for the application to work.

The Android permissions scheme has been widely studied [11, 12, 13, 14]. The android documentation is very short, the descriptions of the permissions are almost trivial, vaguely giving information about what the permission allows to do, but not mapping it to any specific functionality. There has been a lot of previous work trying to find techniques that would allow to map API calls (hence specific actions) to android permissions.

The first big work in this area was the Stowaway project [11], were the authors built a tool that detected overprivilege in compiled Android applications. They tried it with a set of 940 apps, finding that around one-third were overprivileged. More so, they published a list of common developer errors to explain why developers might ask for more permissions that needed. These common errors are:

- **Permission name**: Developers sometimes require a permission because its name sounds related to the functionality that they are building.

- **Deputies**: An application can ask a deputy to perform one action. In this case, the deputy is the one that needs the permission, not the actual application.

- **Related methods**: In some classes, there are different functions that require different permissions. Sometimes developers request both permissions when they only use functions that require one of the two.

- **Copy and Paste**: There are a lots of snippets of code available on the Internet, and sometimes these are incorrect, instructing the developer to request one permission when in fact, it is not necessary.

- **Deprecated permissions**: Android permissions change with time, and some may become outdated. Backwards compatibility or outdated information might lead developers to request for deprecated permissions.

- **Testing artifacts**: Developers might add one permission during testing and then forget to erase it.

- **Signature/System Permissions**: Standard versions of android silently refuse to grant these high privilege permissions to applications that are not signed by the manufacturer. These permissions are either requested by error or requested and then when the developers find that the code is not working, they erase the snippet of code that required this permission but forget to erase the permission request as well.

After this work, the tool PScout [14] was published, which had a more complete mapping between API calls and permissions.

The later work in this matter is the tool Axplorer [13] which allowed the authors to present a novel mapping, based on the study of the internal behaviors of the android framework. In this work, they introduce the concept of permission locality, a low locality means that a certain permission is checked by more than one service (these services could possibly be unrelated). A low locality potentially contributes to app developer's incomprehension of the permissions system.

All of this confusion and the lack of documentation are the main reasons why a lot of applications are overprivileged, and why the work towards a more complete understanding of the matter continues up to date.

One of the mentioned sources [12] study the comprehension of the permission system by phone users. The authors performed two different studies and found that only 17% of the users that took part in both studies paid attention to permissions when installing applications. They also found that only 3% of the population one of the studies users and 24% of the people that took part in the other, demonstrated competent comprehension of permissions.

These data show that users do not fully understand what they allow applications to do when they are installing them. It is important to note that there is a type of software called *grayware*, which is categorized as applications that move on the edge between malicious apps and benign apps, taking advantage of the user without doing anything outright illegal or malicious.

Since users do not fully understand the risks of over privileged apps, it is important to study the set of permissions of an app and compare it to it is functionalities to find any security threats, and to differentiate when one application is overprivileged due to bad engineering and when it can be considered *grayware*. With the literature about the topic, we are inclined to think that in most cases, overprivileged apps happen because of engineering fails by the developers.

2.3 Internet communications

One of the threats of parental control app is that, after gathering all sorts of private information about the kid's phone usage, these data are sent over the internet to the app developer's server. Then, this information is stored remotely, in order to analyze it and to be latter accessed by the parent from a different device than the child's phone.

2.3.1 Client-server model

This is a classic model where the server gathers all data and performs calculations, while the clients just connect to the server and send and receive information. Parental Control Apps scheme is some sort of server client-model, where the child sends all the information to a server, where it is stored. Then, the parent, from a webpage or another app can receive the information organized and analyzed.

Nowadays this kind of schemes have migrated towards a distributed system scheme, where the server side is formed by multiple machines that perform different tasks and balance the work and traffic load to provide the user with a better experience.

To make it easier to understand, in Figure 2.2 we will represent the server-side as one machine.

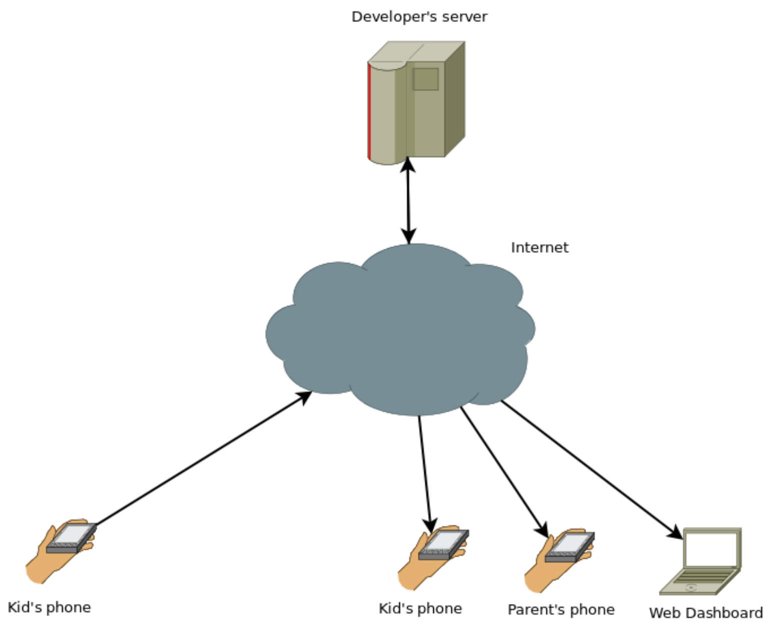


Figure 2.2: Representation of parental control app's communication model

2.3.2 Internet telecommunications stack

Internet communications are built on top of different protocols that take care of all the aspects of the communication. In Table 2.2 we can see a table with the correspondence between TCP/IP layers and OSI layers, which are two ways to organize the different actions that must take place in order to successfully establish a telecommunications network between different peers. OSI layers correspond to the IP layers but adding more granularity.

In the TCP/IP stack, the link layer is concerned about the correct physical transportation of data frames, the Internet layer makes sure that all communications in a multi-node network work properly, transport layer protocols manage the transmissions of data segments between points of a network, and finally, the application layer handles the correct presentation of data and functioning of programs.

Table 2.2: Correspondence between TCP/IP and OSI layers

TCP/IP	OSI Model	Protocol examples
Application Layer	Application Layer	DNS, DHCP, FTP, SSH
	Presentation Layer	JPEG, MIDI, TIFF
	Session Layer	PAP, SCP
Transport Layer	Transport Layer	TCP, UDP
Internet Layer	Network Layer	IPv4, IPv6, ICMP
Link Layer	Data Link Layer	ARP, Token Ring, STP
	Physical Layer	Wi-Fi, Bluetooth, Ethernet

2.3.3 SSL/TLS

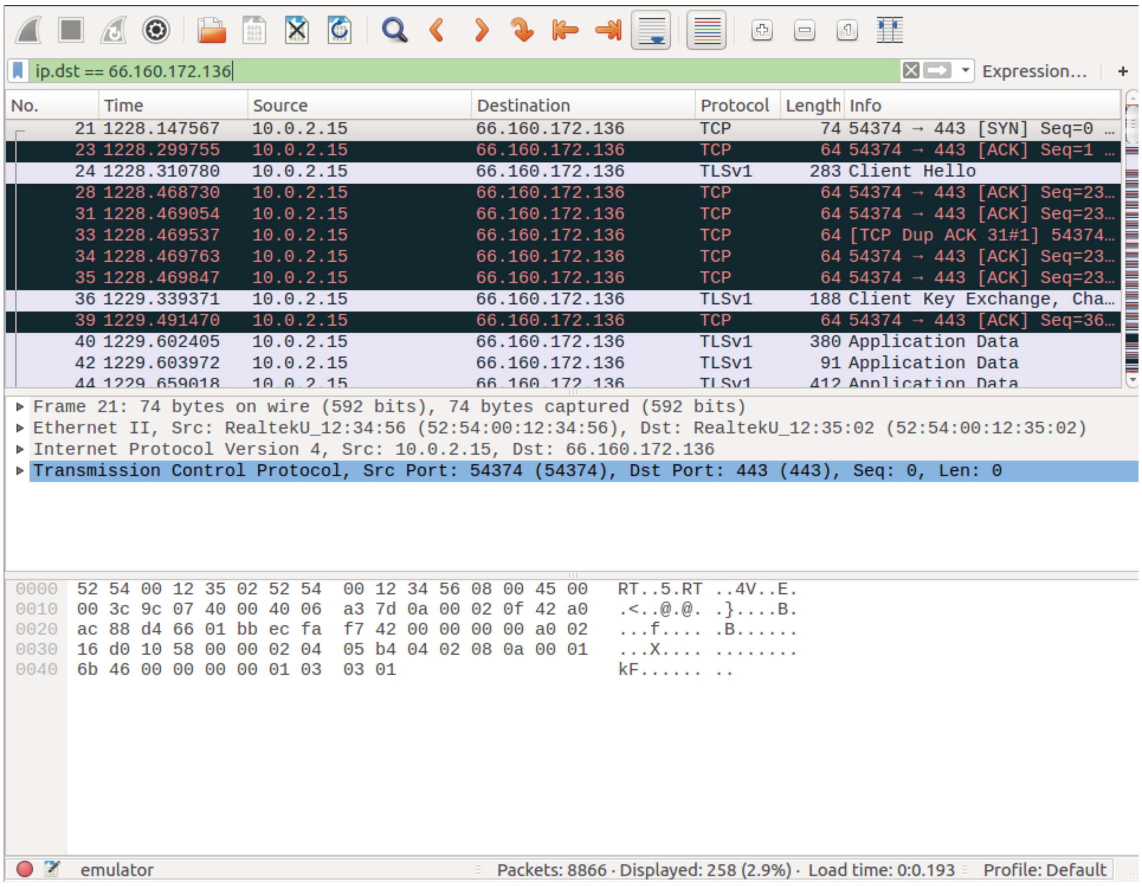
The Transport Layer Security and its predecessor Secure Socket Layer (both usually referred to as SSL), are cryptographic tools that provide an internet communication with encryption, thus making the communication secure to classic attacks like man in the middle (MITM). In these kind of attacks, when the communication is performed in plain text, an attacker intercepts all the packets sent between two peers when these packets leave the sender's device, performing some malicious action, and then sending them to the receiving peer. This gives the attacker ability to read and modify all the messages in the communication without being spotted.

All of the communications between the parental control applications and their corresponding server must happen through SSL, or else the privacy threats would be huge. A way to find out if SSL is being used in the communication, is to inspect the packages sent between the phone and the server, using a tool such as Wireshark [15], introduced in next section.

2.3.4 Wireshark

Wireshark is a network protocol analyzer, used to inspect network packets in order to understand more about the communication behavior. It is an open source tool, available free for all major operative systems.

In Figure 2.3 we show an example, with the packets captured in the communication between one of the applications and it is corresponding server. The information is presented in a vertical tripod, with the first frame showing all the packages, the second specifying the different parts of the package that correspond to each of the protocols and the final part being the data contained in the selected part (in this case, encrypted).



ip.dst == 66.160.172.136

No.	Time	Source	Destination	Protocol	Length	Info
21	1228.147567	10.0.2.15	66.160.172.136	TCP	74	54374 → 443 [SYN] Seq=0 …
23	1228.299755	10.0.2.15	66.160.172.136	TCP	64	54374 → 443 [ACK] Seq=1 …
24	1228.310780	10.0.2.15	66.160.172.136	TLSv1	283	Client Hello
28	1228.468730	10.0.2.15	66.160.172.136	TCP	64	54374 → 443 [ACK] Seq=23…
31	1228.469054	10.0.2.15	66.160.172.136	TCP	64	54374 → 443 [ACK] Seq=23…
33	1228.469537	10.0.2.15	66.160.172.136	TCP	64	[TCP Dup ACK 31#1] 54374…
34	1228.469763	10.0.2.15	66.160.172.136	TCP	64	54374 → 443 [ACK] Seq=23…
35	1228.469847	10.0.2.15	66.160.172.136	TCP	64	54374 → 443 [ACK] Seq=23…
36	1229.339371	10.0.2.15	66.160.172.136	TLSv1	188	Client Key Exchange, Cha…
39	1229.491470	10.0.2.15	66.160.172.136	TCP	64	54374 → 443 [ACK] Seq=36…
40	1229.602405	10.0.2.15	66.160.172.136	TLSv1	380	Application Data
42	1229.603972	10.0.2.15	66.160.172.136	TLSv1	91	Application Data
44	1229.659018	10.0.2.15	66.160.172.136	TLSv1	412	Application Data

```
▸ Frame 21: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▸ Ethernet II, Src: RealtekU_12:34:56 (52:54:00:12:34:56), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 66.160.172.136
▸ Transmission Control Protocol, Src Port: 54374 (54374), Dst Port: 443 (443), Seq: 0, Len: 0
```

```
0000  52 54 00 12 35 02 52 54  00 12 34 56 08 00 45 00   RT..5.RT ..4V..E.
0010  00 3c 9c 07 40 00 40 06  a3 7d 0a 00 02 0f 42 a0   .<..@.@. .}....B.
0020  ac 88 d4 66 01 bb ec fa  f7 42 00 00 00 00 a0 02   ...f.... .B......
0030  16 d0 10 58 00 00 02 04  05 b4 04 02 08 0a 00 01   ...X.... ........
0040  6b 46 00 00 00 00 01 03  03 01                     kF...... ..
```

emulator   Packets: 8866 · Displayed: 258 (2.9%) · Load time: 0:0.193   Profile: Default

Figure 2.3: Example of the Wireshark tool

2.4 Taint analysis

Nowadays, one of the challenges of mobile apps security is that cell phone applications combine data from cloud services with data that comes from various sensors in the phone. While the apps may have legitimate reasons to access this data, users would like to assure that this information is properly used, and that it does not leave the phone unless it is not necessary.

There are different ways to study what information is sent out the cellphone by one application and how this process is done. One of these different methods is taint analysis. Taint analysis consists on studying the flow of a program, marking data that comes from a sensitive source and reporting a possible leak of private information when this labeled info leaves the program scope via a network interface or external storage. In this thesis, we will use this technique to analyze which data are gathered by each of the parental control applications that we test and to find out if it is being sent through the internet.

There are two types of taint analysis: static analysis and, the one we are focusing on this project, dynamic analysis. In static analysis, applications such as Flowdroid [16] track tainted data through the binary code of android applications until they reach a given sink. They later report this information flow to human analysts or automatic malware detection tools that determine if the information leak consists of a privacy policy violation. The main problems with this kind of analysis are that, it takes a lot of computational power for big apps, that the developers can easily bypass the analysis by having boolean assertions that the program can never meet and that this technique reports a lot of false positives (this means showing an information leak when in fact there is not one).

On the other hand, dynamic analysis requires an active user to produce a flow by interacting with the application. When in this flow tainted data leaves the system through a sink, a privacy violation is reported. The problem with this type of analysis is a coverage one, it is much more difficult for an active user to reach 100% of the program branches. The benefits are that is not as easy to bypass and that it usually reports less false positives (even though it still does). The most used tool for this type of analysis, is TaintDroid [17].

2.4.1 TaintDroid

Taintdroid is a system wide dynamic taint tracking and analysis system, that can be used to find leakage of information in android applications during runtime. To summarize the way that this method works, private data is labeled in different levels (variable-level, method-level, message-level and file-level) and then this labels propagate through the program life cycle. Whenever some labeled data reaches a point where it leaves the program control flow, potentially towards an outside source (whether a network adapter or a external storage file), this event is recorded and the user is notified about what data left the system and the sink (a sink is where this data left the system).

Advantages and disadvantages

The main reason behind choosing this application is that after 7 years it is still considered the state of the art for these type of applications. The application works in real time, and has an overhead of only 14% on a CPU-bound microbenchmark run by the authors in [17], translating in negligible overhead in third-party apps.

The limitations are that, in order to work, the authors had to change some functions on the Android native code, making the application available only for certain phones and android versions.

Also, since the authors wanted to keep the application under a minimum overhead, it is possible for truly malicious apps to trick the system through control flows. For this same reason, the authors developed this tool in such a way that there's a big amount of false positives. False positives are instances where a taint is recorded and in fact, there was no actual leakage of data.

Setup

This are the steps to follow in order to start using TaintDroid in an emulator with android 4.1:

1. **Get the Android source code:**

   TaintDroid is built on top of Android, so first, it is necessary to download the correct instance of the Android source code from android's repositories. Once downloaded, the next step is to build the source code (it is best to do so without any modifications). It's recommended to launch the emulator when this is done, to check if the installation has succeed.

2. **Get the TaintDroid source code:**

   In this step, it is time to download the correct version of TaintDroid's source code into the same directory where the Android source code was downloaded. The versions must match in order to work, because, as we said before, TaintDroid works by changing some aspects of the Android source code.

3. **Build TaintDroid:**

   Now that the source code has been downloaded, it must be built in order to achieve the goal to have a working android emulator with TaintDroid installed.

4. **Obtain a kernel with YAFFS2 XATTR support:**

   Finally, since this will be installed on an emulator, this type of kernel must be used in order to have file propagation logic functioning. Now, all there's left to do is launch the emulator.

After following these steps, which are explained in much more detail in [18], we will have a fully functional emulator, with android 4.1 and TaintDroid built in. In Figure 2.4 we can see a screenshot of the working emulator.

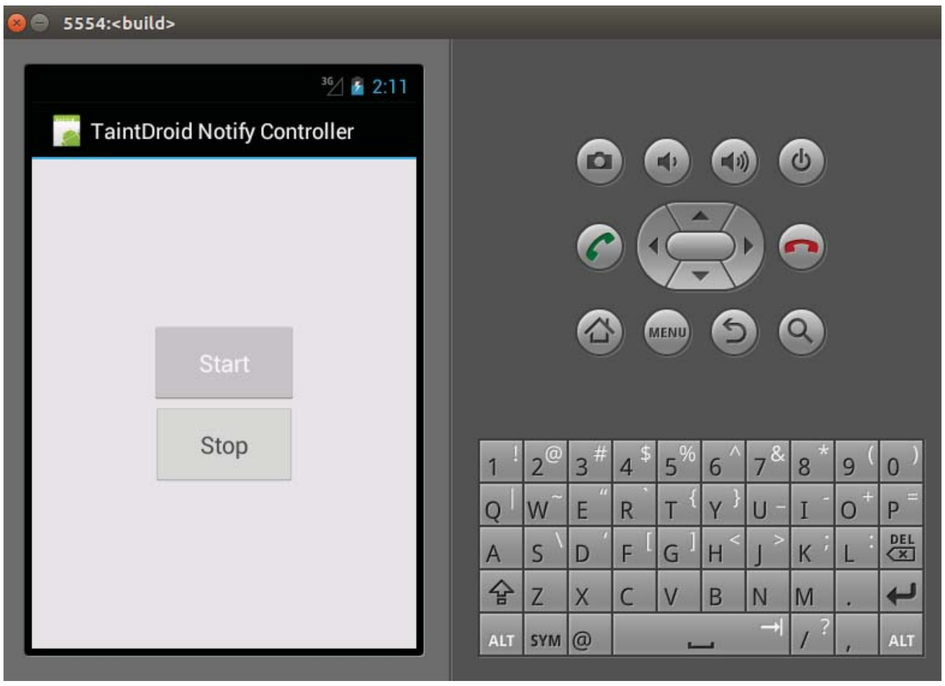


Figure 2.4: Emulator with TaintDroid and Android 4.1 installed

Chapter 3

Parental Control Apps

Parental control apps are used by parents to keep track of their child's activities on the internet. Usually, the parent installs the application in the kid's phone and can later configure a set of rules to dictate what his child can and cannot do with his phone.

The privacy threats on these type of applications are huge, not only because of the privacy levels of the data, but also because this data needs to leave the phone. In order to allow the father to remotely configure the set of rules for the children, and see the collected private data, all of this sensitive information must be sent to the developer's servers. Even under the premise that these companies never intentionally use this private data for anything else that showing it to the parent, there's still a big threat to privacy during the communication process.

3.1 Shortlist of apps

In order to analyze these applications from a privacy point of view, it is important to first understand the way they work and all the features that these programs provide to the user. We chose a small set of the applications available in the Google Play Store, and study the features and behavior of them.

The approach we took for choosing the apps consisted on searching on the Goggle Play Store for the words ”parental control apps”. Then, out of the 50 first results, we pick the ones that meet these requirements:

- Full control parental apps, with blocking and monitoring, not time control only apps, or application blockers only apps.
- More than 1.000-5.000 downloads.

- Available for Android 4.1.
- After installing and testing the apps, if any app does not work it is left out.

This is the shortlist of apps that we analyzed:

Secure Kids [19], Shield My Teen [20], ESET Parental Control [21], Parental Control Light [22], Teen Limit [23], Secure Teen [24], Norton Family [25], MM-Guardian [26], NetNanny [27], Safe Kiddo [28], Quostodio [29], Kaspersky Safe Kids[30], Boomerang Parental Control [31] and *Mobile Fence [32].*

3.2 Study on app behavior

We conducted an study on four different levels:

- App Store: Number of downloads and the ratings of the apps.
- Licensing: Type of license and pricing.
- Parent control: Devices from which the parent can control his child.
- App behavior: Features that each of the apps offers.

All the data that appears on the following tables has been collected from the Google Play Store Marketplace from the months of March 2017 to April 2017. All data is subject to change during time.

3.2.1 Google Play Store

We see in Table 3.1 that these kind of apps are widely used, since most apps have close to half a million downloads. The ratings are often low, some companies blame this fact on the kids coming back to rate the application with a low grade due to anger. In other apps, a lot of parents complain that their kid can easily bypass some of the rules, thus rating the app with a low grade.

3.2.2 License and pricing

In Table 3.2, we can see information about the type of licensing and in the apps where licensing is not free, the price. The three types of licensing that we find are:

Table 3.1: Google Play Store study on parental control apps

	APP STORE	
	Number of installs	Review
Secure Kids [33]	10.000 - 50.000	3.8
Shield my Teen [34]	100.000 - 500.000	3.5
ESET Parental Control [35]	100.000 - 500.000	3.4
Parental Control Light [36]	10.000 - 50.000	4.8
Teem limit [37]	10.000 - 50.000	2.6
Secure Teen [38]	500.000 - 1.000.000	3.4
Norton Family [39]	500.000 - 1.000.000	2.8
MMGuardian [40]	100,000 - 500,000	2.7
NetNanny [41]	100.000 - 500.000	2.5
SafeKiddo [42]	10.000 - 50.000	3.9
Quostodio [43]	500,000 - 1,000,000	3.50
MobileFence [44]	500,000 - 1,000,000	3.00
Kaspersky Safe Kids [45]	100,000 - 500,000	3.70
Boomerang Parental Control [46]	100,000 - 500,000	4.00
Average	100.000 - 500.000	3.40

- **Free**: The whole application is free to use.
- **Premium**: None of the features ca not be used unless the user pays.
- **Freemium**: Some of the features can be used without paying, but premium features can only used by those users who pay.

Most of the studied applications business' model require paying to use the application, but almost all of the apps provide a free trial so the user can first test the program functionalities, before deciding whether he likes the app enough to purchase it or not.

3.2.3 Parent control devices

In Table 3.3 we explain how the parent can control his child's activities. The columns mark the devices from where the parent can access the functionalities of the application. The options are, a web dashboard, a separate application for parents only, or the same application that the kid (usually the one installed the kid's phone). The options are not exclusive, as one application can have one to three options available.

Table 3.2: License and pricing study on parental control apps

	License			
		Pricing		
	Type	Price/Year	Devices	Free Trial
Secure Kids	Free	Free	N/A	N/A
Shield my Teen	Premium	29.99	3	Yes
ESET Parental Control	Freemium	16.52	1	Yes
Parental Control Light	Premium	?	?	Yes
Teem limit	Freemium	39.48	1	Yes
Secure Teen	Premium	39.99	3	Yes
Norton Family	Premium	38.95	5	yes
MMGuardian	Premium	34.99	1	yes
NetNanny	Premium	44.99	5	No
SafeKiddo	Premium	19.95	2	Yes
Quostodio	Premium	38.95	5	Yes
MobileFence	Premium	28.80	3	Yes
Kaspersky Safe Kids	Freemium	14.99	1	No
Boomerang Parental Control	Premium	12.00	1	Yes

Most of the applications have a web dashboard available for parents, which is another point where the security measures should be emphasized, because only the parents, with their pair of credentials should be able to see the information about the child.

On the other hand, when the parent installs the same application than the kid on its personal phone, we need to look very careful at how the set of permissions is used. It's important that, when applications that are so heavily privileged, they do not take advantage of their permissions when installed for reasons that do not involve monitoring the phone's activities.

3.2.4 Parental control features

In Tables 3.4 and 3.5 we show what features each of the applications offers to the parent. All of the results were fully tested on a real phone, except for the following apps were some features could not be tested. In those cases, the approach was to decide whether the functionality was supported or not through the application's description on Google Play, the features described in the publisher's web page, and the options that may have shown in the web dashboard as premium features.

Table 3.3: Study on the devices that can be used to manage parental control apps

	Parent Control		
	Web	App	
	Web Control	Child	Parent
Secure Kids	x	x	x
Shield my Teen	x	x	x
ESET Parental Control	x		
Parental Control Light	x		
Teem limit		x	
Secure Teen	x	x	x
Norton Family	x	x	
MMGuardian	x	x	x
NetNanny	x	x	
SafeKiddo	x		
Quostodio	x	x	
MobileFence	x	x	x
Kaspersky Safe Kids	x		
Boomerang Parental Control	x		x

For ESET Parental Control, Kaspersky Safe Kids and Teen Limit, we could not test all of the features because they were premium only features and not included in the free trial. In Parental Control Light, for signing up to a free-trial, a payment method was required, therefore, we did not test it. Finally Net Nanny did not provide a free trial.

We divided the results on two tables, for clarity and visibility reasons.

In the table above, all the features are divided in active or passive. Active means that the parent can perform any action (such as blocking) and passive means that it can log the child's activity. As can be inferred from the table, one option does not take away the other.

Looking at both tables we see that, all the apps can block webpages and applications, while the rest of the features are only supported by some applications. The most common features are time supervision, which is setting schedules in which the child can use the phone, and monitoring the children's location.

By studying the results, we can see that the level of intrusiveness can vary a lot, from one app to other.

Table 3.4: Parental control features study on parental control apps

	App behaviour									
	App usage		Web Usage		Calls		SMSs		Contacts	
	Pasive	Active	Pasive	Active	Pasive	Active	Pasive	Active	Pasive	Active
Secure Kids		x	x	x		x			x	x
Shield my Teen		x	x	x	x	x	x	x	x	x
ESET Parental Control	x	x	x	x						
Parental Control Light	x	x		x						
Teem limit		x		x						
Secure Teen		x	x	x	x		x			
Norton Family	x	x	x	x			x	x	x	x
MMGuardian	x	x	x	x	x	x	x	x	x	x
NetNanny	x	x	x	x						
SafeKiddo		x	x	x						
Quostodio	x	x	x	x	x	x	x	x		x
MobileFence	x	x	x	x	x	x	x			
Kaspersky Safe Kids	x	x	x	x	x		x			
Boomerang Parental Control	x	x	x	x	x	x	x			

Table 3.5: Parental control features study on parental control apps

	App behaviour					
	Time supervision	Social media	Block the phone	Location	Alarms	Emergency mode
Secure Kids	x		x	x	x	x
Shield my Teen				x		
ESET Parental Control	x			x		
Parental Control Light	x			x		
Teem limit	x			x		
Secure Teen	x	x	x	x		
Norton Family	x	x	x	x		
MMGuardian	x		x	x		
NetNanny	x					
SafeKiddo	x			x		
Quostodio	x	x		x		x
MobileFence	x		x	x		
Kaspersky Safe Kids	x	x	x	x		
Boomerang Parental Control	x			x		

3.3 Other kind of parental control apps

In the study above we have only included applications that monitor the child's phone, because these are the type of apps that are more intrusive and, therefore, pose a bigger risk to the privacy of users. Nevertheless, there are other type of parental control apps in the market, which can also be dangerous for the users, even thpugh they are less intrusive by definition.

There are mainly, two other types of parental control programs:

- **Internet browsers**: In order to protect the kids from the internet risks, parents use these type of applications that replace other browsers and filter the contents of web pages, blocking the sites that belong to inappropriate categories.
- **Application blockers**: This type of apps substitute the regular Android launcher, substituting the home screen with a custom one where only the applications that the parent allows will be accessible for the child.

In this study, we test some of these applications, in order to provide completeness to this work.

3.4 Approach

We take the first three applications of this kind that were found when constructing the shortlist of parental control applications ans test them in an Android 4.1 emulator with the software TaintDroid installed. We install and use the applications and then explain all the possible leaks that we find.

The applications that we will test are *Protect Kid, ChildLock* and *SafeBrowser*, being the first two applications application blockers and the last one an internet browser.

KidsZone

This application blocks the phone and lets the user access only accepted apps. The developer's explain that this app is thought to be installed on the parent's app, and that when the kid uses the phone, the parent can easily lock it so the child does not access dangerous apps. We expect this application to work locally, because there's no need to send any kind of information through the internet, locking the device locally should be enough.

ChildLock

This app is an application blocker that replaces the normal Android home screen with a new one where only the apps that are approved by the parent appear and can therefore be used. This application could work perfectly without sending any kind of information through the internet (except login information which is considered normal and not a privacy threat).

Safe Browser

This application is just a web browser which categorizes webpage depending on their contents, in order to be able to block certain categories so the user wo not be able to access them. This type of application should work locally because, if the parent does not have to ability to remotely monitor the web usage, then there's no reason to send data to a server in order to store it.

Chapter 4

Privacy Analysis

In order to meet all of the objectives of this work, we performed four different studies. For each one of them, we explain the setup, actions performed to get to the results, and what results we find.

In the following sections, we describe each one of the four different studies that we performed, ordered depending on at what stage the study tasks can be performed: before, during or after installation. The first study, which helps us understand how clearly app publishers explain the way their product works to the users before installing is therefore done prior to app installation. The second one,which gathers information about what permissions each application requests and tries to map them to functionalities is performed when visiting the Google Play Store for installation of the program. Finally, the last two experiments, which study what information gathers each of the tested apps and if they are sent to Internet servers in a secure manner ot not, are conducted once the apps are installed on the phone.

4.1 Potential differences in the understanding of the applications behavior between users

4.1.1 Setup

For the first study, we take into consideration all of the apps that are present in the shortlist, no matter if they work on the emulator or not.

One problem with this type of applications is that they target a class of users who may not have a lot of knowledge about computer engineering or computer security topics. Therefore, if this applications do not specifically explain that all of the child's data is going through the network and being stored in a server in order to later be consulted by the parents, they will most likely not know about this behavior. It is possible that many parents would prefer to choose using these kind os apps knowing that their child's data is being stored in third party servers.

The approach we follow consists on gathering the descriptions in the applications' Google Play entry, and manually inspect and study it in order to determine whether it is clear that these applications operate on a server level. We consider that a non-expert user depends on this clarification in order to understand the behavior of this applications. On the other hand we assume that a mid-expert user will know that this type of applications can not work locally, without a server side storying the information.

4.1.2 Actions performed

In order to perform the study, first we create a dataset with all the information corresponding to each of the apps, including name and Google Play description. Then, for each of the dataset entries we read it and look for a clear explanation about the fact that these kind of applications do not operate locally. If this is the case, then the application it is marked as equally clear to a mid-expert and to a non-expert user. On the other hand, if there is not a clear indication about the locality of the application, it is marked as confusing for a non-expert users.

The descriptions that specifically talk about sending data to servers or use terms like "remotely manage" are marked as clearly descriptive. Nevertheless, if the description does not come close to explaining the locality, the application is marked as unclear. Also, in the applications with an unclear description, we install the application on a phone, in order to discover if before the app starts gathering information, the user is notified about data being sent through the network to a server.

4.1.3 Results

The results can be seen in the Table 4.1. Only seven out of fifteen applications clearly explained that the app was not local, and that it would indeed work remotely (as in sending data to servers).

There is not a correlation between the apps that do and the ones that do not, but, it is noticeable that most applications from known security companies do not talk about this issue in their descriptions (even though some inform about it after installation). It is probable that parents will trust this kind of developers more than "unknown" ones, therefore parents will probably not look as closely to the app's descriptions in comparison.

Table 4.1: Manual study of app's descriptions

	Clear description about locality
Secure Kids	YES
Shield my Teen	YES
ESET Parental Control	YES
Parental Control Light	YES
Teem limit	NO
Secure Teen	YES
Norton Family	NO
MMGuardian	NO
NetNanny	YES
SafeKiddo	NO
Quostodio	NO
MobileFence	NO
Kaspersky Safe Kids	YES
Boomerang Parental Control	NO

The fact that applications do not state this on their descriptions is nor a privacy threat, neither do we say that it is done with malicious intents of fooling the user. What we want to prove is that it is usual for this kind of apps to be focused on explaining features that parents will consider useful while not talking about some behaviors that the parents should know, understand and accept upon installing the app.

4.2 Study on permissions requested by parental control applications

4.2.1 Setup

In this study, we analyze the set of permissions that each of the applications request. We take this information from the Google Play Store's application entry and divide it in different groups depending on the nature of the permissions. Google already categorizes the permissions in different groups, and we work with this same group classification.

The categories that we use and the Google groups that are included in each one of them, are the following:

- **Monitoring information**: Device and app history.
- **Calls and messaging**: SMS, Phone, Device and call information.
- **Accounts and contacts**: Identity, Contacts.
- **Sensors and storage**: Photos/Media/Files, Storage, Camera, Microphone, Location
- **Wi-Fi connections**: Wi-fi connection information.
- **Others**: Others.

The permissions that one application requests mapped to the functionalities that it provides can be a good way to find out if this software can be categorized as grayware. It is important to note that, in many cases, the excess of permission request are due to bad programming and not to bad intentions, but anyhow, it is good to know when one app is doing something wrong, whether it is done accidentally or intentionally. As it is demonstrated in many works [11, 14, 13], overprivileged applications are very usual because of the difficulty that understanding when one Android permission is needed in order to perform one action and when it is not poses.

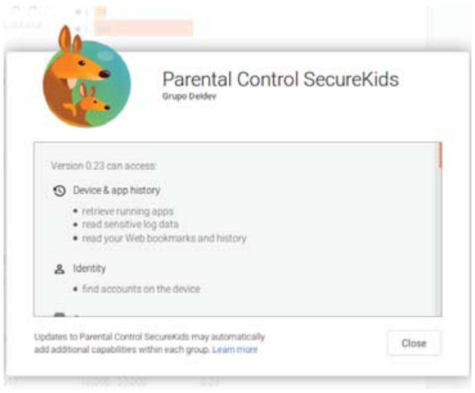
4.2.2 Actions performed

To get the results, we take each of the apps Google Play Store entry and in the additional information section, we find the details about the permissions (an example can be seen in Figure 4.1). These details consist of a list, explaining what actions the app is requesting to have permission to perform. Each one of these actions can be maped to one or more permissions, which can be found on the android manifest of the application. Since it is more understandable, we will use the Google Play Store description of the permissions, instead of the actual permissions that appear in the android manifest of each application.

Finally, we prepare several tables, where each one of the permissions that appear on at least one app, becomes a column of one table. With the subset of all the permissions that are granted to at least one app, we proceed to complete the table adding one application per row and marking down if the application request each permission or not.

4.2.3 Results

For this study, we show the results categorized in the groups that we described in the setup subsection. For some of the groups, we have divided the results in more


Parental Control SecureKids
Grupo Deldev

Version 0.23 can access:

Device & app history
- retrieve running apps
- read sensitive log data
- read your Web bookmarks and history

Identity
- find accounts on the device

Updates to Parental Control SecureKids may automatically add additional capabilities within each group. Learn more

Figure 4.1: Example of the detailed permission view from the Google Play Store

than one table, in order to keep the number of columns small enough to fit on the page width.

For each of the groups, we explain what the permissions requested are usually needed for, and we try to map it to any functionality of the application. In the case of permissions that we consider to be harder to explain, we will mark them as a probable misuse of the permission privileges.

Monitoring information

In Table 4.2, we can see that all of the tested applications can retrieve other programs running on the phone at any given moment, also, eleven of the thirteen applications can access the web history and bookmarks. These permissions make sense when we compare them to the functions that the applications have (3.2.4). We find two apps that do not have access to the web history, but they are the ones that only block and not monitor Internet navigation.

The read sensitive log data is an old permission that stopped working in Android Jelly Bean. It means that the application can read the application logs, keeping track of crashes and running activities.

Calls and messaging

In the Table 4.3, we show the permissions related to text messages. They are divided in SMS, MMS and WAP. SMS are text-only messages, while MMS supports multimedia data as well. WAP messages contain links to media allocated on the web, and are usually received when a user purchases a game, wallpaper, or ringtone via paid sms.

All the apps that request the permission to *Receive, send, modify and/or read*

Table 4.2: Monitoring information permissions study

	Device & app history		
	Retrieve running apps	Read sensitive log data	Read your Web bookmarks and history
Secure Kids	✓	?	✓
Shield my Teen	✓		✓
ESET Parental Control	✓	?	✓
Parental Control Light	✓		
Teem limit	✓		
Secure Teen	✓		✓
Norton Family	✓		✓
MMGuardian	✓		✓
NetNanny	✓	?	✓
SafeKiddo	✓		✓
Quostodio	✓		✓
MobileFence	✓		✓
Kaspersky Safe Kids	✓	?	✓
Boomerang Parental Control	✓		✓

text messages (except ESET Parental Control) are the ones that support the monitoring of text messages. In the case of ESET Parental Control, we do not see any feature that should require these permissions.

Table 4.3: Messaging permissions study

	SMS					
	Read your text messages (SMS or MMS)	Receive text messages (SMS)	Receive text messages (MMS)	Receive text messages (WAP)	Send text messages (SMS)	Edit your text messages (SMS ot MMS)
Secure Kids						
Shield my Teen	✓	✓				
ESET Parental Control	?	?			?	?
Parental Control Light						
Teem limit						
Secure Teen	✓	✓			✓	
Norton Family	✓	✓	✓	✓	✓	✓
MMGuardian	✓	✓	✓		✓	✓
NetNanny						
SafeKiddo						
Quostodio	✓	✓			✓	✓
MobileFence	✓					
Kaspersky Safe kids	✓					
Boomerang Parental Control	✓	✓			✓	✓

In Table 4.4, we see the permissions related to the calls features. The *Read phone status and identity* is used by almost al of the applications, because these permission is necessary to receive the unique identifier of the phone (IMEI) which is used for unique communication purposes and to link the data received in the server to the right user.

Of the eight applications with any of the other call permissions, six of them have some feature that backs up the need to request it. In the case of ESET Parental Control and Teen Limit there is not a feature that clearly explains the need for this permission.

Table 4.4: Call permissions study

	Phone					Device
	Directly call phone numbers	Reroute outgoing calls	Read phone status and identity	Read call logs	Write call logs	Read phone status and identity
Secure Kids	✓	✓	✓			✓
Shield my Teen	✓		✓	✓	✓	✓
ESET Parental Control	?	?	✓	?		✓
Parental Control Light			✓			✓
Teem limit	?			?	?	
Secure Teen	✓	✓	✓			✓
Norton Family	✓	✓	✓	✓	✓	✓
MMGuardian			✓			✓
NetNanny			✓			✓
SafeKiddo			✓			✓
Quostodio	✓	✓	✓	✓	✓	✓
MobileFence	✓	✓	✓			✓
Kaspersky Safe Kids			✓	✓		✓
Boomerang Parental Control	✓	✓	✓	✓	✓	✓

Accounts and contacts

In Table 4.5 we show the permissions related with user accounts and contacts and the ability to read, modify, add and delete contacts and user accounts.

Almost all of the apps have the *Find accounts on the device* permissions, which is usually used in the login process, to autocomplete the fields with data from existing accounts on the device. Also, two application request *Adding or removing accounts*, which is used to create new accounts on the device's account manager to facilitate the process of login back in. For NetNanny, we could not test if this is the case. In the ESET Parental Control app, after installing and creating and account, a new entry does not appear on the account manager, therefore we ca not be sure if this permissions is needed by the application.

The applications that request the ability to *Read and/or modify contacts* should do so if they provide call/SMS features. This is the case for all, except for ESET Parental Control. In the case of *Read your own contact card* we could not find any documentation as to what this is needed for.

Table 4.5: Accounts and contacts permission study

	Identity			Contacts		
	Find accounts on the device	Add or remove accounts	Read your own contact card	Find accounts on the device	Read your contacts	Modify your contacts
Secure Kids	✓			✓	✓	✓
Shield my Teen					✓	
ESET Parental Control	✓	x	?	✓	?	
Parental Control Light	✓			✓		
Teem limit	✓			✓		
Secure Teen	✓			✓	✓	
Norton Family	✓			✓	✓	
MMGuardian	✓		?	✓	✓	✓
NetNanny	✓	x		✓		
SafeKiddo	✓			✓		
Quostodio	✓			✓	✓	✓
MobileFence	✓			✓	✓	
Kaspersky Safe Kids	✓			✓	✓	
Boomerang Parental Control	✓	x		✓	✓	✓

Sensors and storage

In Tables 4.6 and 4.7 we present, divided in two categories, all permissions related to camera, microphone and location in the sensors table, while in the storage category we find the permissions related to internal and external storage access.

All the permissions make sense, *Access to storage* is required by most applications that need to save any kind of data on the phone. Also, the camera permissions are usually requested when there is a possibility to set a profile picture. The other sensor permissions make sense, because some applications have their own messaging app that requires access to these sensors.

Table 4.6: Storage permissions study

	Photos/Media/Files		Storage	
	Read the contents of your USB storage	Modify or delete the contents of your USB storage	Read the contents of your USB storage	Modify or delete the contents of your USB storage
Secure Kids	✓	✓	✓	✓
Shield my Teen				
ESET Parental Control	✓	✓	✓	✓
Parental Control Light	✓	✓	✓	✓
Teem limit				
Secure Teen				
Norton Family	✓	✓	✓	✓
MMGuardian	✓	✓	✓	✓
NetNanny	✓	✓	✓	✓
SafeKiddo	✓	✓	✓	✓
Quostodio	✓	✓	✓	✓
MobileFence	✓	✓	✓	✓
Kaspersky Safe kids	✓	✓	✓	✓
Boomerang Parental Control	✓	✓	✓	✓

Also, all of the applications request access to the location information, which corresponds to the fact that all the applications provide the feature to track the child's whereabouts.

Wi-fi Connection

In Table 4.8 we show which of the studied applications request the permission required to read information about configured wi-fi connections and current connections. Since all of these applications need the Internet to work, it is expected that they request this permission.

Others

This Google Play category contains most of the permissions that this applications request, and we will show the results of this study in many different tables, in order to make it easier to read and understand.

Table 4.7: Sensors permissions study

	Camera	Microphone	Location	
	Take pictures and videos	Record audio	Precise location (GPS and network-based)	Approximate location (network-based)
Secure Kids	✓		✓	
Shield my Teen			✓	✓
ESET Parental Control	✓		✓	
Parental Control Light			✓	
Teem limit			✓	✓
Secure Teen			✓	✓
Norton Family			✓	✓
MMGuardian	✓	✓	✓	
NetNanny	✓		✓	
SafeKiddo			✓	✓
Quostodio			✓	
MobileFence	✓		✓	✓
Kaspersky Safe Kids			✓	
Boomerang Parental Control	✓		✓	✓

Table 4.8: Wi-Fi connection permission study

	Wi-Fi Connection Information	Cellular data settings
	View Wi-Fi connections	Change/intercept network settings and traffic
Secure Kids	✓	
Shield my Teen	✓	
ESET Parental Control	✓	
Parental Control Light	✓	
Teem limit		
Secure Teen	✓	
Norton Family	✓	
MMGuardian	✓	
NetNanny	✓	
SafeKiddo	✓	
Quostodio	✓	
MobileFence	✓	
Kaspersky Safe Kids	✓	✓
Boomerang Parental Control	✓	✓

There is not a given order to the way we are presenting these tables, other than the order of appearance in the Google Play Store permission information. Tables will be numerated from 1-10 in the caption, to improve the order and clarity of the study.

Even though we will explain with detail what each permissions and why we think it is needed or not, to improve the self explanatory capacity of the tables we will use three values per column. We use ✓when we could find a clear explanation for the need of that permission, we use **x** when we find a possible explanation for the permission and, finally, we use **?** when we do not find an explanation for the given permission.

Table 4.9: Other permissions study

	Choose widgets	Bind to an accesibility service	Download files without notification	Interact with a device admin	Enable or disable app components
Secure Kids	?				
Shield my Teen		✓			
ESET Parental Control		✓			
Parental Control Light		✓	x		
Teem limit		✓			
Secure Teen		✓			
Norton Family		✓			
MMGuardian		✓	✓		
NetNanny					
SafeKiddo	?				
Quostodio					
MobileFence		✓			
Kaspersky Safe Kids					
Boomerang Parental Control				x	x

In Table 4.9, there are several actions that do not present a threat. *Bind to an accessibility service* is requested so the app can use services that help users with disabilities in using the phone and the application. *Enable or disable app components* could be used to save battery life, by turning off unnecessary components. The *Choose widgets* permission, allows the application to create a widget to easily access data related to this app, even though none of the apps that we tested created any widget.

The *Download files without notification* permission is a dangerous one, because an application could download malware into the user's phone. MMGuardian installs a browser on the phone, and that could possibly be the reason to request this permission; in the case of Parental Control Light we could not test the app enough to see a reason to request this permission.

The request for *Interact with a device admin* allows the application to have rich control over the device, which can be dangerous, but could make sense that an application requests it, given the kind of applications we are talking about.

In Table 4.10 all the permissions are requested only by one application, Boomerang

Table 4.10: Other permissions study

	Delete other app's data	Delete apps	Power device on or off	Force stop other apps	Directly install apps
Secure Kids					
Shield my Teen					
ESET Parental Control					
Parental Control Light					
Teem limit					
Secure Teen					
Norton Family					
MMGuardian					
NetNanny					
SafeKiddo					
Quostodio					
MobileFence					
Kaspersky Safe Kids					
Boomerang Parental Control	x	x	?	?	x

parental Control. The permission to *Delete other app's data* and *Delete apps*, could be needed to manage a browser application that is installed along with the parental control software. This could also be the reason to request the *Directly install apps* permission, but Boomerang does not directly install, recommending to do so instead.

For the need to request permission to perform the actions *Power device off* and *Force stop other apps* we do not find an explanation.

Table 4.11: Other permissions study

	Interact across users	Full license to interact across users	Add or remove a device admin	Manage users	Update component usage statistics
Secure Kids					✓
Shield my Teen					✓
ESET Parental Control					
Parental Control Light				?	✓
Teem limit					✓
Secure Teen					✓
Norton Family					✓
MMGuardian					✓
NetNanny					✓
SafeKiddo					✓
Quostodio					✓
MobileFence					✓
Kaspersky Safe Kids					
Boomerang Parental Control	?	?	?	?	✓

In table 4.11, we find the permission to *Update component usage statistics*, which most applications request in accordance to their features. In this table, we also see *Interact across users* and *Full license to interact across users*, which are dangerous permissions that allow the application to perform actions across different users and that we do not find any reason for the apps to request.

For requesting the permissions *Manage users* and *Add or remove a device admin* we do not find a particular reason, and these permissions, specially the latter can be dangerous.

Table 4.12: Other permissions study

	Read sync statistics	Close other apps	Adjust your wallpaper size	Read Home settings and shortcuts	Write Home settings and shortcuts
C0C0C0Secure Kids		✓	?		
Shield my Teen					
ESET Parental Control					
Parental Control Light					
Teem limit					
Secure Teen					
Norton Family					
MMGuardian					
NetNanny	✓	✓			
SafeKiddo		✓	?	?	?
Quostodio		✓			
MobileFence					
Kaspersky Safe Kids		✓			
Boomerang Parental Control		✓			

In Table 4.12, we find three permission request that do not seem necessary, *Adjust your wallpaper size*, *Read Home settings and shortcuts*, *Write Home settings shortcuts*. The permission to *Read sync statistics* could be used to improve communications with the server. Finally, the permission to *Close other apps* could be related to applications that install other programs, but some of the apps that request these permissions do not install other software. Nevertheless, this request could be also related to the feature of blocking other apps.

Table 4.13: Other permissions study

	Status bar	Modify app ops statistics	Modify secure system settings	Receive data from Internet	View network connections
Secure Kids				✓	✓
Shield my Teen					✓
ESET Parental Control				✓	✓
Parental Control Light				✓	✓
Teem limit				✓	✓
Secure Teen				✓	✓
Norton Family				✓	✓
MMGuardian				✓	✓
NetNanny					✓
SafeKiddo				✓	✓
Quostodio				✓	✓
MobileFence				✓	✓
Kaspersky Safe Kids				✓	✓
Boomerang Parental Control	?	?	?	✓	✓

In Table 4.13 we find the permissions *Receive data from Internet* and *View network connections*, which are related to the constant communications between the phone and the developer's servers.

The Boomerang Parental Control application request permissions to *Status Bar* (open, close, or disable the status bar and its icons), *Modify app ops statistics* (aps ops was a somehow hidden android tool that allowed the user to manage the permission for apps, only worked in Android 4.3 and 4.4.1, and now available only for rooted devices) and *Modify secure system settings* (allows the app to modify secure settings that are usually managed only by pre-installed app, we consider this a dangerous permission). We did not find any clear explanation to why these might be needed.

Table 4.14: Other permissions study

	Pair with Bluetooth devices	Access Bluetooth settings	Read battery statistics	Change network connectivity	Connect and disconnect from Wi-Fi
Secure Kids			?	✓	✓
Shield my Teen					✓
ESET Parental Control				✓	✓
Parental Control Light				✓	✓
Teen limit					
Secure Teen					✓
Norton Family					
MMGuardian				✓	✓
NetNanny					
SafeKiddo					
Quostodio					
MobileFence	✓	✓			✓
Kaspersky Safe Kids				✓	✓
Boomerang Parental Control	?				✓

In Table 4.14, we see two bluetooth related permissions which are, *Pair with Bluetooth devices* and *Access Bluetooth settings*. In the case of MobileFence the bluetooth related permissions are needed because it can block Bluetooth settings. In the case of Boomerang Parental Control, we could not find any feature Bluetooth related. We did not find any reason to ask for the *Read battery status* permission.

Secure Kids request the permission to *Read battery status*, whose objective is not clear. Nevertheless, this permission should not be dangerous. Finally, most of the applications request the permissions *Change network connectivity*, *Connect and disconnect from Wifi* which is related to the fact that all of this apps work constantly with a network connection to send all data to remote servers.

Table 4.15: Other permissions study

	Disable your screen lock	Expand/collapse status bar	Create accounts and set passwords	Full network access	Control Near Field Comunnication
Secure Kids	x	?		✓	
Shield my Teen				✓	
ESET Parental Control	x			✓	
Parental Control Light	x			✓	?
Teem limit				✓	
Secure Teen				✓	
Norton Family				✓	
MMGuardian				✓	
NetNanny			?	✓	
SafeKiddo				✓	
Quostodio				✓	
MobileFence	x	?		✓	
Kaspersky Safe Kids				✓	
Boomerang Parental Control		?		✓	

In Table 4.15, we find three apps that can not be mapped to any feature or need of the apps. These are *Expand/collapse status bar*, *Create account and set passwords* (which is specially dangerous) and *Control Near Field Communication.*

In the case of *Full network access*, the need for this permission is again explained by the continuous communication between these apps and the developer's servers. Finally, the request for *Disable your screen lock*, could be explained with the feature of sending custom alarms and notifications to the children that some applications provide.

Table 4.16: Other permissions study

	Read Google service configuration	Read sync settings	Run at startup	Set wallpaper	Draw over other apps
Secure Kids			✓	?	
Shield my Teen			✓		
ESET Parental Control			✓		✓
Parental Control Light			✓	?	✓
Teem limit			✓		✓
Secure Teen			✓		
Norton Family	✓		✓		✓
MMGuardian			✓		✓
NetNanny		✓	✓		
SafeKiddo			✓	?	✓
Quostodio			✓		
MobileFence			✓		✓
Kaspersky Safe Kids			✓		✓
Boomerang Parental Control		✓	✓		

In tTable 4.16, we can see that the permission *Run at startup* is requested by all the applications in the list, which is needed in order to prevent the child to bypass the app's features by rebooting the phone.

The permission to *Read syn settings* is most likely used for improving communications with the server. We could not find any explanation to why the permission *Set wallpaper* would be necessary for these kind of applications. Also, the permission that grants access to *Read Google Service configuration* is needed to use many of the new features in Google Play Services, such as better location, routing, and maps.

Finally, *Draw over other apps* is needed to show notifications above all other apps, which links directly to the feature that many applications have to let the parent send a critical alert or message to the child.

Between the permissions that we can see in Table 4.17 we find *Prevent device from sleeping*, which is needed for applications to be able to gather information about cellphone usages at all times. The permission to *Control vibration* is most likely used for the features of setting alarms, making an emergency call to the children and sending emergency messages to the phone.

The permission *Set preferred apps* is deprecated and may not bo longer be used, it used to work so that one application could decide to be the default app for some action without asking the user. The permission *Use accounts on the device* means that the app might ask permission to access some online services (such as Google, Facebook or Twitter). Finally, the permission to *Reorder running apps* means that the app can move applications between the background and foreground, which we ca not map to any features.

Table 4.17: Other permissions study

	Set preferred apps	Reorder running apps	Use accounts on the device	Control vibration	Prevent device from sleeping
Secure Kids				✓	✓
Shield my Teen					✓
ESET Parental Control					✓
Parental Control Light			x		✓
Teem limit				✓	✓
Secure Teen					✓
Norton Family					✓
MMGuardian				✓	✓
NetNanny		?	x	✓	✓
SafeKiddo				✓	✓
Quostodio				✓	✓
MobileFence				✓	✓
Kaspersky Safe Kids	?				✓
Boomerang Parental Control				✓	✓

Table 4.18: Other permissions study

	Modify system settings	Write web bookmarks and history	Toggle sync on and off	Install shortcuts	Uninstall shortcuts
Secure Kids	✓	?		?	
Shield my Teen					
ESET Parental Control		?			
Parental Control Light	✓		✓	?	
Teem limit					
Secure Teen					
Norton Family		?			
MMGuardian	✓	?		?	?
NetNanny	✓		✓	?	
SafeKiddo				?	
Quostodio		?			
MobileFence		?			
Kaspersky Safe Kids	✓	?			
Boomerang Parental Control	✓				

In Table 4.18, we see that six apps request the permission to *Modify system settings*, which can be dangerous but is used by the applications to change settings such as brightness or WiFi between others. The requested permission to *Toggle sync on and off* is used to improve communications between the apps and the servers.

Finally, the permissions to *Write web bookmarks and history*, *Install shortcuts* and *Uninstall shortcuts* do not obviously map to any functionalities of the applications, and we can not state for what purpose they might be needed.

4.3 Privacy information disclosure in parental control apps

4.3.1 Approach Setup

We study each of the apps in 3.1 that work in the android 4.1 emulator with TaintDroid installed. Each app is isolated, because due to their features and nature, running alongside other parental control apps will cause blocking of features and unexpected behaviors. This means that we will install one of the apps, proceed to fully test it and then remove it in order to continue to do this experiment with the next application.

In order to study each of the apps, the approach is to manually test the app, performing a certain set of actions that could possible trigger the sending of private info to the developer's server. Then, for each application, we note if any of the actions resulted in an unexpected behavior or leak of information.

The goal of the experiment is to find any unexpected leakage of information, or any action that will result in a privacy violation or threat. It is important to note that given the behavior of these kind of apps, most of the information leaks are expected. For example, if an application sends the browser history to the developer's server and this corresponds to one of the app features, we wo not consider it a threat or violation towards privacy.

4.3.2 Actions performed

1. **Installation and setup**:

   Taints that may occur while installing the app for the first time and signing into an account to activate the parental control features. It is important to check if this information is sent before or after the user agrees with the privacy policy, and also when the application shows this terms to the user and how. In some of the applications the user is prompted with this policy and needs to actively accept it before login in, in other cases the developers warn the user that when creating an account the user is accepting the terms of use and privacy policy. Whenever the user is not notified about this policies, we consider it a privacy issue.

2. **Browsing the Internet**:

   Browsing the Internet using the default Android browser and, in the case of applications that install their own browser, using both. We visit allowed and blocked webpages and check if these actions produce a leak of information.

3. **Using apps**:

   We open different applications, blocked and allowed, and use them for a while to check if this produces any taint report.

4. **Calling a phone number**:

   Even though the emulator does not have service, we use the Android dialer app to perform a call which will be saved in the log even when there is no service. We then check if info related to this call is leaked.

5. **Sending a text message**:

   The case is the same that with a call, the phone does not have service but allows us to send a text message, registering it in the text SMS history.After sending the message, we check if there is any information leakage.

6. **Using a fake location**:

   The emulator does not have location services available for use, but we downloaded an application that sets a fake location. Using this app we can see if the studied applications gather information about the phone's whereabouts.

4.3.3 Results

These are the results of the experiment for each of the individual apps.

Kaspersky Sake Kids

Even though this app does not work in our emulator, we tested the application in a real phone to see if we could find any useful information. What we found is that, when installing this app, the user must read and accept the terms of use. When looking through these terms we found that this application sends data to third parties. These third parties are two analytics sites (Appsflyer and Facebook, which will also appear later on in other applications) and Google AdWords, which is an online advertising service. Also, in this terms the application developers explain that the communication with Google AdWords is not secure.

Privacy problem: This app sends private information to third parties and, while the user is notified of this behavior and must accept it upon using the application, we consider a privacy issue the fact that one of these communications (with Google AdWords) is not secure.

Shield My Teen

Destination of data: When testing applications, we analyze the messages sent to the servers looking for the hostname of these machines. This information is recorded by the app TaintDroid. Nevertheless, in this case, the hostname does not appear.

Installation and setup: We install this application, an open it up for the first time creating a new account. When we finish the whole process and the application starts monitoring the device, there are a lot of messages sent to the developer's server. With this communication, the phone application sends information about the configuration options that the parent chose for this phone, the IMEI code which is an unique identifier of each mobile phone, used by these type of apps to let the server know to which user the information received belongs to and finally, after a while, the SMS history of the phone is sent to the developer's server in order to let the parent access it remotely.

Privacy problem 1 : One of the issues that we find in the installation of this app is that the user is never notified about the terms of use or the privacy policy, therefore the application starts gathering data without the active approval of the user.

Privacy problem 2 : The expected behavior when using one monitoring application is that the app should start gathering data in the moment of installation. In the case of Shield My Teen this is not true, because the complete SMS history is sent to the server upon installation.

Browsing the Internet: This application does not install a browser app, therefore we use the default Android application to browse the Internet. We access two different webpages, a sports newspaper (which is allowed) and a gambling site (which is blocked). After waiting for around 10 minutes, we see that, in both cases, the browsing history is sent to the server, as expected.

Using apps: When we first installed the application, there is a message sent to the server with the list of all the installed applications. This message is sent in order to let parents know what applications the kid has installed and be able to decide which ones to block. We try blocking one application to see if the behavior when using blocked and allowed apps is different. We open and use one application that is not blocked and also try to access the one that we have previously blocked. None of these options result in an information leak.

Sending a text message: When the application is installed the whole text message history is sent to the developer's server, including messages from before the application was installed. Also, we tried to send a text message and found that around 10 minutes after sending it, the content and receiver information are sent to the server.

Calling a phone number: In this application, this feature did not work. When installing the app, the call history is not sent to the server, and when we performed a call there was not any leak.

Setting a fake location: We use one application that allows us to set a fake location, and we trick the system into thinking that the phone is located in the Madrid city center. Around 10 minutes after setting the location, this information is sent to the server and then we check in the web dashboard that the fake location has been recorded.

Parental control Light

This app does not work on emulators, therefore we could only perform some of the actions in our study, because it is possible to install the application and check if there is any information leakage even if the user is not able to set up an account.

Data destination: The information is not only sent to their own server, but also to a domain belonging to appsflyer. This is an analytics product, that tracks user activities within the app, to show to the developers and help them make decisions based on these analytics [47].

Installation and set-up: When the app is first opened, it prompts the user with a message saying that the IMEI code will be sent out to be used for unique communication issues. Nevertheless, before accepting this, the IMEI has already been sent to one address that belongs to an analytics software. Once we accept the message, this same code is sent to a different address belonging to the developers.

Privacy problem: This application sends information to a third party before informing the user of the privacy policy. Furthermore, after sending the IMEI code to a third party, the user is prompted with a message that claims that no information (including the IMEI) will be sent other parties. This is a great privacy issue, because not only the info is sent before approval but also the user is deceived about the app's behavior.

Secure Teen

Destination info: When we start trying out this application, we see several images on the web dashboard that also appeared on the app Shield My Teen. Due to this fact, we decide to check the developers of both applications and realize that they are developed by the same company. We believe that probably this group decided to do a renovated application after releasing the first one. We use TaintDroid to check if they share the same server and we can confirm that they do, therefore, the destination server is the same for both apps.

Installation and set-up: When we start the app for the first time and finish the whole registration process, accepting the privacy policy when doing so, the application starts monitoring the phone usage. Also, there is an initial communication between the app and the server where the Address Book and the SMS history are sent through the Internet.

Privacy problem: When this application sends the SMS history to the server, information from before the app had been installed is also sent. This is a problem because the expected behavior is that these type of applications only monitor text messages from the moment they have been installed by the user.

Browsing the Internet: For this application we use the default Android browser, because it does not install a new browser. We access a safe page (sports newspaper) and one that would be considered as dangerous for a child (gambling). After a couple minutes, this information is sent to the server.

Using a fake location: We use a fake location in Madrid city center to see if this information is sent to the server. Furthermore, we see that in the web dashboard of this app, you can set how often you want to gather location info (between 8 and 2880 minutes). Once we set this time, after that given time, the phone sends this information to the server.

Sending a text message and calling a phone number: We merge the result of both of these actions together, because the application behavior is the same. We call a phone number, and also send a text message and around five to ten minutes later, we can observe a leak with the information regarding the text message or call being sent to the server.

Norton Family

This application can be installed in the parent as well as in the child's phone. The expected behavior when installing the app on the parent's device is that there is no leakage of any kind and that when we perform all actions below, there is not any taint reported by the utilized tool. We perform this analysis checking that the expected behavior is met and that there is not an information leakage of any kind.

After trying out the parent mode, we reinstall the application ins the same device, but this time we configure it in child mode, allowing the application to monitor the phone's usage information. The results of the testing of this application in child mode are below.

Destination info: We use the tool TaintDroid to discover where the information is sent, and we see that the app sends requests to two different domains: urlcat.norton.com which is owned by Norton Cloud and eventlog-rd.norton.com that belongs to the Microsoft Corporation IP range. Given these domains, it seems like they belong to two different Norton services, which does not indicate any privacy threat.

Installation and setup: When we install the app and open it for the first time, the terms of use must be accepted before continuing to use the application. Then, we proceed to set up a new account and after this process is finished, the application starts monitoring the phone's info. Also, at the beginning, a list with the installed apps is sent to the server, in order to let the parent see this list on the web dashboard and block applications if necessary.

Browsing the Internet: Norton Family provides the user with its own safe browser application, therefore, we browse the Internet with this app in order to perform this experiment. In the same manner as before, we browse one application that would be considered as safe, which is a sports newspaper, and one that would be considered as inappropriate for kids, which is a betting webpage. As soon as we access either of the webpages, we see that this information is sent to the server. Specifically, we see that one of the services (urlcat) receives only information when this action is performed.

Using apps: We access the web dashboard and block one of the installed applications in order to then access this blocked app. We also open and use one of the applications that is not in the blacklist of blocked apps. None of these actions result in any information leakage.

Using a fake location: Even though the location feature is advertised on the application, we did not find any reference to it in the web dashboard. Furthermore, when we set a fake location on the emulator the information about this location is never sent to the server.

Calling a phone number: The calls feature is not supported by this application. Nevertheless, we try to call a phone number and confirm that there is no any taint report.

Sending a text message: When we try to access this feature from the parent web dashboard, the website advises us that this feature is only supported in Canada and the United States. Anyways, we try it in our emulator and confirm that there is not any leakage of info, but we believe that this is because there is no service and therefore, the developer's have no way of placing us either in Canada or the United States.

MMGuardian

Destination of data: This application shows a communication problem with the phone, this probably due to the fact that it was not possible to functionally install Google Play services on an emulator and apparently, this service is needed in some applications for communication matters. Nevertheless, we proceeded to test it, and found that even though the communication is nor working, the app sends messages to three different addresses: one belonging to the developers, another one from the Facebook analytics API and the last one from thor.komodia which is a webpage categorizer [48].

Setup and installation: We install the application and open it for the first time, then we create a new user and accept to install the developer's own web browser, thus accepting the terms of use and privacy policy. After this process is done, the address book, SMS and Call history and the IMEI code are leaked to the address that belongs to the developers server. The IMEI is also leaked to the Facebook analytics server, nevertheless, the user has accepted this behavior when creating an account and therefore this is not a privacy problem.

Browsing the web: This application installs its own browser, therefore we use it to visit URLs that are considered safe as well as unsafe webpages. We find that everytime we try to access one URL, the app sends a request to the browser categorizer to decide if the content is safe for the child or not. Also, after accessing Internet sites, the renovated web history is sent.

Privacy problem: It is dangerous to send a request to a third party everytime a web site is accessed, but in this case the request does not clearly say what webpage is being visited. Further analysis should be conducted in order to decide if the visited webpage could be inferred from this request.

Using a fake location: Since there are communication issues with the application and the web dashboard, we are not able to get this information on demand. Also, it looks like the location info is not sent by the child's phone unless the parent asks for it.

Sending a text message: This application also installs its own messaging application, which the father can configure to monitor the message contents and even filter inappropriate words. We try sending a SMS with this application and we observe that as expected the information relating the sender, receiver, and content is sent to the server.

Calling a phone number: We try to call a phone number using the build in dialer application and there is not any leakage of information when we perform this action.

Boomerang Parental Control

Destination of data: When analyzing this application with TaintDroid we are able to learn that there are communications with three different domains, two of them belong to the Facebook analytics API, while the other belongs to the developer's server and is app.useboomerang.com.

Installation and setup: After installing the application, when we open it for the first time there is a taint record that tells us that the IMEI code is sent to two different addresses that belong to the Facebook analytics API. After properly creating a new account (thus accepting the privacy policy) configuration and device related information (such as battery life, memory status. . .) is sent to the app's developers server.

Privacy problem: The sending of private information before the user has agreed with the terms of use is a privacy violation. In this particular case, the user agrees to it when creating a new account, and before this action has been performed, there has already been a leakage of information to an analytics program Internet server. Any leak that happens after creating an account is accepted, because the privacy policy accepted by the user explains that this sending of information to third parties may happen.

Sending SMS: This application does not install a messaging app, therefore we send a message using the built in Android app and find that after a time of the order of thirty minutes, the information regarding this SMS is (sender, receiver and content) sent to the server.

Using a fake location: The communication between the dashboard and the child's phone does not work in our setup, due to the fact that we could not install Google Play services in our emulator. Because of this issue we can not actively request the location, and the phone does not leak this information if it is not required by the parent at the web dashboard.

4.3.4 Other kind of Parental Control Apps

We followed the same course of action that in Section 4.3, performing for each application every one of the actions explained in our approach for the applications identified in Section 3.3.

We will explain the results of our study for each one of the apps, focusing on the applications where we were able to find privacy problems.

Kids Zone

Our study shows that there is not any kind of tainted data leaving the system, and that this application is safe to use without any risk of the private data of the phone being compromised by being sent through the network and stored in servers.

Child Lock

We find that every time we set up the lock and switch to the special launcher with only allowed applications, the application sends information to two different addresses. One of them corresponds to a server in the domain ad.leadboltapps.net, which is an advertising network to which these apps send the IMEI of the phone in plain text, in order to play ads for the user.

The second address belongs to an analytics application that gathers usage data in order to take marketing decisions. The application sends the location data to this analytics app through a non-secure communication channel. Therefore, we find two *privacy problems*, the sending of location data to third parties without telling the user, and also that some parts of these data are sent in the clear.

Safe Browser

We throughly test this application, installing it and browsing safe web pages as well as sites that fall into a blacklisted category. After the study there is not any sign of leakage of information or private data.

4.4 Usage of SSL in the client-server communication

4.4.1 Setup

The applications in our study are constantly sending information to the developer's server, since all the information sent is private, and should therefore be confidential, these communication channels should use the SSL/TLS protocol. It is highly important that the communications are encrypted, or else the security threats will greatly augment, because anyone could read the data sent by the phone.

In this experiment, using the tool Wireshark, we gather data sent by all the applications that we were able to get to work in the emulator, and then we study if the communication with the different servers is encrypted or not.

4.4.2 Actions performed

In order to perform this study, we launch the emulator, with the option to save all traffic data sent by the phone in a file that we will later analyze. Then, we install one by one each of the applications that work in an emulator, and perform the actions that will result in a leak to each of the different servers that the app may communicate with. Following the same approach that in the previous experiment described in 4.3 and learn the specific IP addresses to where the information is being sent. Then, once we know the specific IP addresses for each of the apps, we analyze the data traffic with the tool Wireshark. For this, we filter the data by destination address, using the list of IPs that we have gathered. Finally, for each of the addresses we mark if the data is sent on the clear or encrypted.

4.4.3 Results

Shield My Teen and Secure Teen

Both of theses apps belong to the same developer, and have almost the same features. Also, they communicate with the same server, which belongs to the app developers company. We gather packets sent to this server and confirm that in both cases, the data is sent through a secure communication channel using the SSL/TLS protocol.

Parental Control Light

In the case of this application we only have access to the messages that are sent when opening the application, because the rest of the features do not work in our emulator. Nevertheless, we are able to check that when communicating with either the developer's server or the analytics site appsflyer, the information is sent encrypted, thus not presenting a security threat.

Boomerang Parental Control

During the previous study we found that this app sends data to three different addresses, even though two of those belong to the Facebook analytics API. When we analyze them with the tool Wireshark, we can see that each one of the communications use the SSL/TLS protocol, this being secure.

Norton Family

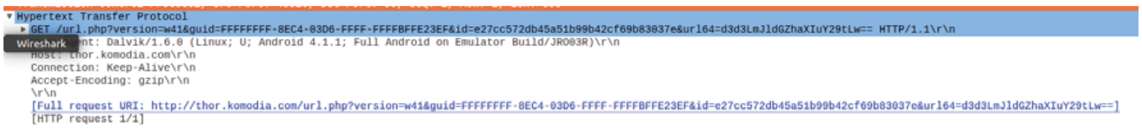
When analyzing this application, we find that every event is logged in the server by sending one message to the eventlog-norton domain. Nevertheless, when the user accesses one webpage, there is also a request to the urlcar-norton domain. When we analyze the communication with these domains we find that, while the event login messages are encrypted, the request that is sent when accessing one webpage is sent in plain text.

Privacy problem: The information about what webpages the user is accessing is always sent without encryption, making it possible for an attacker to gather information about the whole web history of the user.

MMGuardian

In the case of this app, after the analysis we find that the application communicates with three different servers. One of these belongs to the application developers and the second one is an analytics program from Facebook and the last one is a web categorizer, that receives a request and responds with the webpage category. Using the tool Wireshark, we learn that the communications with the first two servers happen under the SSL/TLS protocol, with all data being encrypted. Nevertheless, the information sent to the last server, the URL categorizer, is sent in the clear, even though this info does not leak the exact URL of the visited webpage (as can be seen in Figure 4.2).

Privacy problem: Even if we need further analysis to probe that the web history could be inferred from the communication with the web categorizer, sending information in the clear is enough reason to categorize this as an privacy issue.



```
▾ Hypertext Transfer Protocol
  ▸ GET /url.php?version=w41&guid=FFFFFFFF-8EC4-03D6-FFFF-FFFFBFFE23EF&id=e27cc572db45a51b99b42cf69b83037e&url64=d3d3LmJldGZhaXIuY29tLw== HTTP/1.1\r\n
    ent: Dalvik/1.6.0 (Linux; U; Android 4.1.1; Full Android on Emulator Build/JRO03R)\r\n
    Host: thor.komodia.com\r\n
    Connection: Keep-Alive\r\n
    Accept-Encoding: gzip\r\n
    \r\n
    [Full request URI: http://thor.komodia.com/url.php?version=w41&guid=FFFFFFFF-8EC4-03D6-FFFF-FFFFBFFE23EF&id=e27cc572db45a51b99b42cf69b83037e&url64=d3d3LmJldGZhaXIuY29tLw==]
    [HTTP request 1/1]
```

Figure 4.2: Image of the request to a web categorizer

Chapter 5

Conclusions

In this thesis we carried out the first in depth study of the privacy of parental control apps. We studied 14 parental control apps, as well as 3 blocking apps and safe browsers. In the case of parental control apps: first, we checked if the publishers were clear when explaining to the user the behavior of these applications; second, we studied the set of permissions that these apps requested and categorized them as necessary or doubtful; third, we gained information about what data parental control apps gather about the user and fourth, we analyzed is the gathered data is sent to Internet servers in a secure manner. In the case of the three other applications, since they should work locally, we checked if there was any kind of information being sent to Internet addresses.

5.1 Findings

The primary goal of this study is to find any kind of privacy threats or private information misuse, in order to understand the privacy risks of using parental control apps. In Table 5.1, we summarize all of our findings related to this objective.

We found privacy risks and/or mishandle of private information in each one of the seven apps that we tested to some extent. It is important to remark that in some applications full testing could not be achieved, leaving room to think that we might have found even more threats.

The most usual privacy problems are the sending of private information before the user accepts the privacy policy and the sending of private information unencrypted, which happen in 43% of the analyzed applications. Furthermore, in one application the user does not approve the privacy at all, finishing installation and setup without seeing this policy. Also, 28% of applications leaked information about phone usage prior to the moment when the monitoring application was installed.

Finally, one of the application sends sensitive information (visited webpages) to a third party server. It is interesting to note that two of the seven apps present more than one problem.

In Section 4.3.4 we also analyzed other type of parental control apps, which are used for blocking applications or as a safe Internet browser. We expected these apps to be local, but found that one of the three studied applications sent information to third party servers. Furthermore this private info is sent through the Internet to third-party companies without encryption.

As can be seen in Table 4.1 , we also found that only 50% of the studied applications were clear about the fact that private information is being sent to their servers and stored there, in order to let the parent access it from other device. While this is probably not a matter of maliciousness, but transparency instead, it is still important to notice this behavior.

Table 5.1: Finding of privacy threats on the analyzed apps

	Parental Control Light	MMGuardian	Norton Family	Shield My teen	Secure Teen	Boomerang parental Control	Kaspersky Safe Kids
Sensitive information sent to a 3rd party		x					
Leakeage of monitoring information from before app installation				x	x		
Leakeage of info before policy acceptance	x			x		x	
Information sent to servers in the clear		x	x				x
User does not accept privacy policy				x			

Finally, in our permission study, we found that 17% of the permissions that the applications requested were not clearly mapped to any functionality, more specifically, 13% of the requested permissions could not be marked as necessary for either of the actions that the application performs, and the other 4% were ambiguous permissions which we could not categorize as neither of the other two provabilities. In Figures 5.1 and 5.2 we can see the total and individual results respectively.

All the permissions categorized as *Unnecessary (?)* in these charts, are the permissions to which we could not find a clear to reason to why they are needed, we decided to categorize them as unnecessary to make it easier to understand, even though in order to categorically assure that they are unnecesary we would have to perform further analysis and probably contact the developers.

The results distributed by application show that only 21% of the apps request only clearly necessary permissions. Out of the 14 studied programs, the ones that request the bigger number of doubtful permissions were Eset Parental Control with 13 and Boomerang Parental Control with 17, accounting for 41% of the total permission request that are not easily explained.

These results allow us to say that parents must be aware of the risks that they pose before installing this kind of software on their children telephone.

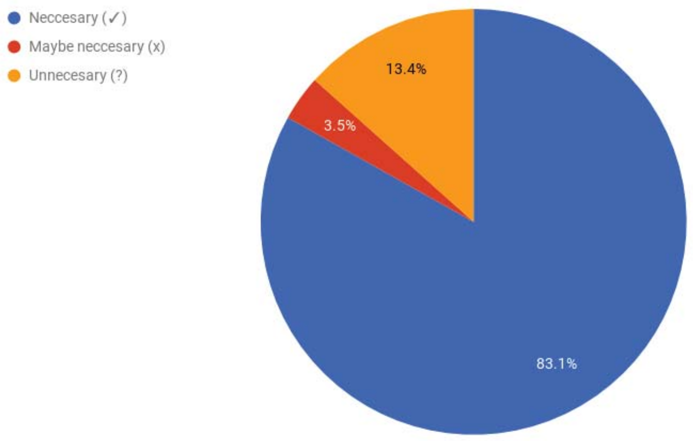


Figure 5.1: Percentage of over privilege in the studied apps

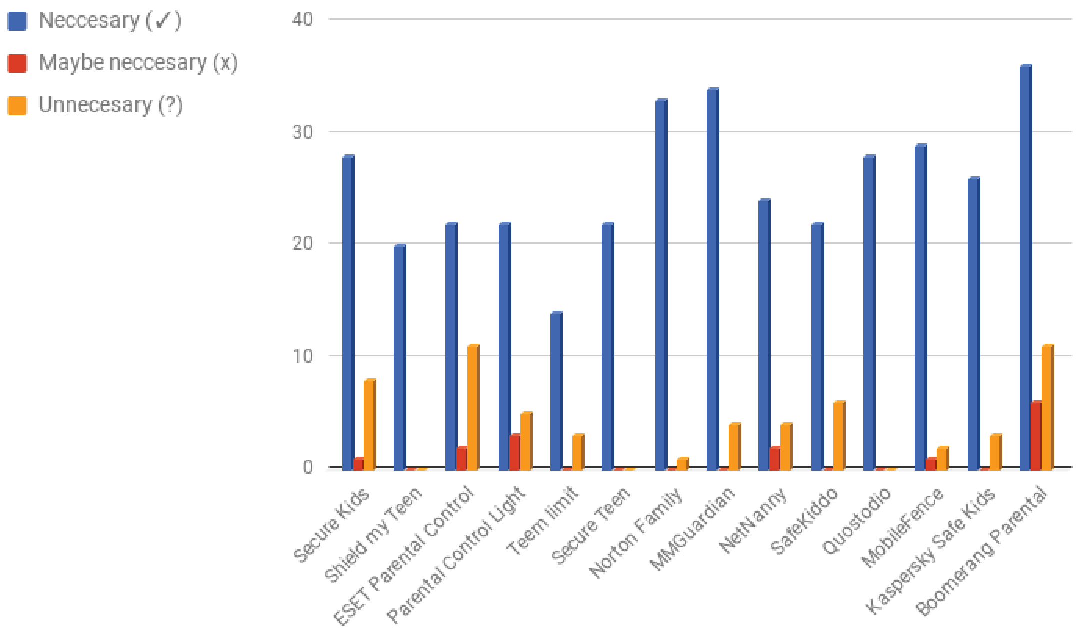


Figure 5.2: Results of study on permissions

5.2 Limitations and future work

The biggest limitation of our study is that, because of using an emulator, most of the apps could not be fully tested. Out of 13 apps that are in our shortlist, only six of them can be run on an emulator and even then, two of them had features that did not work in our setup. One solution for this problem would be to install the TaintDroid application in a real phone. We tried to do this, but the developers only affirm that it will work in a small number of phone models to which we did not have access. We attempted to install it in other models but did not succeed and when searching for precedence we did not find any. In the future, a solution would be to have access to a phone where TaintDroid can be installed, and then repeat the study with more apps.

Another limitation is that the software TaintDroid, depends on some tweaks to the Android system that the the developers had to implement. Because of this reason, the researchers have only made the tool available for up to Android 4.3. It would be interesting to be able to study all these applications in other versions of Android, specially above Android 6.0 where the permission system changed.

Because of this same reason, the applications could not be tested in the other big operative system, iOS, which is the exclusively built for Apple devices. It would be interesting to study the possible differences between the versions of one application for each operative system.

Bibliography

[1] "Android Dashboards," 2017. Online; accessed May 29, 2017.

[2] R. Jones, R. Kumar, B. Pang, and A. Tomkins, ""i know what you did last summer": Query logs and user privacy," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, (New York, NY, USA), pp. 909–914, ACM, 2007.

[3] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson, ""i still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks"," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, (Washington, DC, USA), pp. 147–161, IEEE Computer Society, 2011.

[4] J. Krumm, *"Inference Attacks on Location Tracks"*, pp. 127–143. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[5] "Smartphones Are Used To Stalk, Control Domestic Abuse Victims," Online; accessed June 6, 2017.

[6] S. Livingstone, L. Haddon, A. Görzig, and K. Ólafsson, "Risks and safety on the internet: the perspective of European children: full findings and policy implications from the EU Kids Online survey of 9-16 year olds and their parents in 25 countries," *LSE Research Online*, 2012.

[7] "Cyberbullying," 2006. Online; accessed May 10, 2017.

[8] D. J. Solove, ""I've Got Nothing to Hide" and Other Misunderstandings of Privacy," *San Diego Law Review, Vol. 44, 2007*, 2007.

[9] "Smartphone OS Market Share," 2016. Online; accessed May 29, 2017.

[10] "Number of apps available in leading app stores as of March 2017," 2017. Online; accessed May 29, 2017.

[11] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android Permissions Demystified," *Conference on Computer and Communications Security (CCS)*, 2011.

[12] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android Permissions: User Attention, Comprehension, and Behavior," *Symposium on Usable Privacy and Security (SOUPS)*, 2012.

[13] M. Backes, S. Bugiel, E. Derr, P. McDaniel, D. Octeau, and S. Weisgerber, "On Demystifying the Android Application Framework: Re-Visiting Android Permission Specification Analysis," in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 1101–1118, USENIX Association, 2016.

[14] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "PScout: Analyzing the Android Permission Specification," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), pp. 217–228, ACM, 2012.

[15] "Android Dashboards," Online; accessed June 1, 2017.

[16] S. Arzt, S. Rasthofer, C. Fritz, E. Bodde, A. Bartel, J. Klein, Y. L. Traon, D. Octeau, and P. McDaniel, "FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps," *PLDI*, 2014.

[17] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," *OSDI*, 2010.

[18] "TaintDroid Build Instructions for Android 4.1." `http://fs.fish.govt.nz/Page.aspx?pk=7&sc=SUR`, 2012. Online; accessed April 24, 2017.

[19] "Secure Kids." `https://securekids.es/`. Online; accessed March 12, 2017.

[20] "Shiled My Teen." `http://www.shieldmyteen.com/`. Online; accessed March 12, 2017.

[21] "ESET Parental Control." `https://parentalcontrol.eset.com`. Online; accessed March 12, 2017.

[22] "Parental Control Light." `https://www.netsparkmobile.com/en/`. Online; accessed March 12, 2017.

[23] "Teen Limit." `https://www.teenlimit.com/`. Online; accessed March 12, 2017.

[24] "Secure Teen." `http://www.secureteen.com/`. Online; accessed March 12, 2017.

[25] "Norton Family." `https://family.norton.com/web/`. Online; accessed March 12, 2017.

[26] "MMGuardian." `http://www.mmguardian.com/`. Online; accessed March 12, 2017.

[27] “Net Nanny.” https://www.netnanny.com. Online; accessed March 12, 2017.

[28] “Safe Kiddo.” http://safekiddo.com/en_us/. Online; accessed March 12, 2017.

[29] “Quostodio.” https://www.qustodio.com/en/. Online; accessed March 12, 2017.

[30] “Kaspersky Safe Kids.” https://usa.kaspersky.com/safe-kids. Online; accessed March 12, 2017.

[31] “Boomerang Parental Control.” https://useboomerang.com/. Online; accessed March 12, 2017.

[32] “Mobile Fence.” https://www.mobilefence.com/. Online; accessed March 12, 2017.

[33] “Secure Kids on Google Play.” https://play.google.com/store/apps/details?id=com.securekids.launcher_reloaded&hl=en. Online; accessed March 12, 2017.

[34] “Shield My Teen on Google Play.” https://play.google.com/store/apps/details?id=com.infoweise.parentalcontrol.shieldmyteen&hl=en. Online; accessed March 12, 2017.

[35] “ESET Parental Control on Google Play.” https://play.google.com/store/apps/details?id=com.eset.parental&hl=en. Online; accessed March 12, 2017.

[36] “Parental Control Light on Google Play.” https://play.google.com/store/apps/details?id=con.netspark.mobile&hl=en. Online; accessed March 12, 2017.

[37] “Teen Limit on Google Play.” https://play.google.com/store/apps/details?id=com.teenlimit.android.child&hl=en. Online; accessed March 12, 2017.

[38] “Secure Teen on Google Play.” https://play.google.com/store/apps/details?id=com.infoweise.parentalcontrol.secureteen&hl=en. Online; accessed March 12, 2017.

[39] “Norton Family on Google Play.” https://play.google.com/store/apps/details?id=com.symantec.familysafety&hl=en. Online; accessed March 12, 2017.

[40] “MMGuardian on Google Play.” https://play.google.com/store/apps/details?id=com.mmguardian.childapp&hl=en. Online; accessed March 12, 2017.

[41] "Net Nanny on Google Play." https://play.google.com/store/apps/details?id=com.contentwatch.ghoti.cp.browser&hl=en. Online; accessed March 12, 2017.

[42] "Safe Kiddo on Google Play." https://play.google.com/store/apps/details?id=com.safekiddo.kid&hl=en. Online; accessed March 12, 2017.

[43] "Quostodio on Google Play." https://play.google.com/store/apps/details?id=com.qustodio.qustodioapp&hl=en. Online; accessed March 12, 2017.

[44] "Mobile Fence on Google Play." https://play.google.com/store/apps/details?id=com.mobilefence.family&hl=en. Online; accessed March 12, 2017.

[45] "Kaspersky Safe Kids on Google Play." https://play.google.com/store/apps/details?id=com.kaspersky.safekids&hl=en. Online; accessed March 12, 2017.

[46] "Boomerang Parental Control on Google Play." https://play.google.com/store/apps/details?id=com.nationaledtech.Boomerang&hl=en. Online; accessed March 12, 2017.

[47] "Appsflywer, mobile app tracking and attribution." https://www.appsflyer.com/, 2012. Online; accessed May 23, 2017.

[48] "URL server protocol - Komodia." http://www.komodia.com/newwiki/index.php/URL_server_protocol. Online; accessed May 10, 2017.

[49] D. E. Denning and P. J. Denning, "Certification of Programs for Secure Information Flow," *Communications of the ACM*, 1977.

[50] "Android Open Source Project. Android Security Overview," 2012.

[51] B. Andow, A. Nadkarni, B. Bassett, W. Enck, and T. Xie, "A Study of Grayware on Google Play," 2014.