Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Análisis de la seguridad y privacidad ofrecida por dispositivos Android

Tesis que presenta

Laiphel Marco Gómez Trujillo

para obtener el Grado de

Maestro en Ciencias en Computación

Directores de la Tesis:

Dr. Francisco José Rambó Rodríguez Henríquez
Dr. Luis Julián Domínguez Pérez

Ciudad de México **Octubre, 2017**

Dedicatoria

A mi mamá, a mi hermano y a mi abuela.

Agradecimientos

Gracias a mi familia, por su amor, su apoyo incondicional, sin el cuál, no habría llegado hasta aquí, su estóica paciencia y por impulsarme a superarme cada día.

A mi asesor, el Dr. Francisco Rodríguez-Henríquez, por aceptarme como su tesista, por sus consejos, sus enseñanzas, su paciencia y porque nunca dejó de creer en mí. A mi asesor, el Dr. Luis Julián Domínguez Pérez, por sus consejos, su ayuda y por haberme apoyado durante el desarrollo de este trabajo.

Al Cinvestav, mi *alma mater*, por haberme permitido ser parte de esta prestigiosa institución y realizar un posgrado. Al CONACyT, por el apoyo económico brindado durante mis estudios de la maestría.

A mis amigos y compañeros del Cinvestav, por su amistad, su ayuda, sus consejos y por las experiencias que vivimos juntos, desde las más gratas hasta las más adversas.

A Sofi, por su apoyo durante la realización de mis trámites, así como por su amabilidad y paciencia.

A la Dra. Sonia Mendoza Chapa y al Dr. Luis Gerardo de la Fraga, mis sinodales, por sus comentarios para enriquecer este trabajo.

A los profesores del Departamento de Computación, por compartirme sus concimientos durante sus clases.

Al CIMAT Zacatecas, por permitirme realizar una estancia en esta institución, para poder progresar en la realización de este proyecto y por hacer de la misma, una amena experiencia.

Resumen

El sistema operativo Android ofrece una gama de funcionalidades que facilitan al usuario la realización de cuantiosas actividades de su vida cotidiana y la mayoría de los dispositivos con este sistema se venden a precios que están al alcance de casi cualquier persona. No obstante, el entorno Android alberga distintos riesgos de seguridad en mecanismos como los servicios de comunicación y su procedimiento de borrado de información. Por lo tanto, la información personal del usuario queda expuesta y así un atacante puede obtener esa información incluso si ésta fue eliminada.

En esta tesis se pretende encontrar cuáles son las fugas de privacidad que tiene un dispositivo Android, para recuperar información del usuario; realizando ataques de husmeo, para interceptar la comunicación del dispositivo; recuperando información eliminada del modo normal y del modo seguro de restablecimiento de fábrica. También se hace un análisis de los mecanismos de WiFi y Bluetooth, de los algoritmos de cifrado en el medio de almacenamiento, de la generación y almacenamiento de las llaves de cifrado de algunos fabricantes y de la seguridad de algunas aplicaciones presintaladas por los fabricantes, del navegador Chrome, del navegador nativo de la AOSP y de la API de navegación (Webview).

Este trabajo se realiza con la finalidad de descubrir nuevos agujeros de seguridad y vulnerabilidades de Android y probar la eficacia de su servicio de confidencialidad de algunos mecanismos del dispositivo. Así mismo se busca concientizar a los usuarios lo que implica no tomar las medidas de seguridad necesarias en los dispositivos móviles y sensibilizar a los fabricantes y vendedores de dispositivos a que tomen en mayor consideración la privacidad del usuario.

Abstract

The Android operating system offers a set of functionalities that makes easier the performing of many personal and work user activities. However, the Android environment has many different security issues as a comunication services and its information deleting procedure. Therefore, the user's personal information is exposed and thus an attacker can obtain that information even if it was deleted.

This thesis aims to find out what are the leaks of privacy that has an Android device, to retrieve information from the user; performing sniffing attacks, to intercept device communication; retrieving deleted information from normal mode and safe factory reset mode. Also, an analysis of the WiFi and Bluetooth is made, also as well as the encryption algorithms in the storage medium, the generation and storage of the encryption keys of some manufacturers and the security of some preintalled applications by the manufacturers, of the Chrome browser, AOSP's native browser, and the navigation API (Webview).

This work is done in order to discover new security holes and vulnerabilities of Android and test the effectiveness of its service of confidentiality of some mechanisms of the device. It also seeks to raise the awareness of users, which implies not taking the necessary security measures on mobile devices and making device manufacturers and vendors aware of the privacy of the user.

Índice general

Índice de tablas

Índice de figuras

Capítulo

1

Introducción

"Predecir es muy difícil, y sobre todo el futuro"

Niels Bohr

Android es el sistema operativo móvil dominante para dispositivos móviles y tabletas (*tablets*), ocupando un 87.6 % de la cuota de mercado mundial en el segundo trimestre de 2016[1]. En Agosto de 2017 la versión más utilizada a nivel mundial es la 6 (*Marshmallow*)[2], y en un cercano segundo lugar, está la versión 5 (*Lolipop*). En México, en el 2015 el sistema de Google dominaba aproximadamente el 82.5 % del mercado mexicano[3].

Los dispositivos móviles o teléfonos inteligentes y las tabletas (*tablets*) poseen casi las mismas capacidades que las computadoras personales de hace algunos años, pero a diferencia de éstos, tienen mayor portabilidad y son más pequeños.

También, la variedad de aplicaciones que ofrece, sus componentes en hardware como la cámara, el giróscopo y el acelerómetro, su capacidad para insertar medios de almacenamiento extraíbles como la tarjeta SD y sus servicios de comunicación que permiten conectarse a internet y comunicarse con otros dispositivos, ha dado lugar a que la popularidad de los dispositivos móviles con Android haya crecido notablemente, en los últimos años, porque ha ayudado a los usuarios a realizar varias

[1] http://www.idc.com/prodserv/smartphone-os-market-share.jsp, consultado el 15 de octubre de 2016

[2] https://developer.android.com/about/dashboards/index.html, consultado el 09 de septiembre de 2017

[3] http://www.luisgyg.com/blog/2015/11/21/android-domina-mexico-2015-comscore/ consultado el 20 de octubre de 2016

actividades personales y laborales de diversa índole como entretenimiento, mensajería instantánea, compras en línea y transacciones bancarias a través de su gran variedad de aplicaciones.

A la fecha, la popularidad de Android ha aumentado y ha encabezado la preferencia de los usuarios en los últimos años. Los factores que han contribuído a este acontecimiento son los precios accesibles de los dispositivos, ya que no es necesario pagar una licencia para obtener este sistema operativo y gran parte del mismo es de código abierto. Adicionalmente, el conjunto de herramientas que proporciona el *kit* de Desarrollo de Software (*Software Development Kit, SDK*) de Android de forma gratuita, permite que los desarrolladores puedan programar sus aplicaciones sin necesidad de tener mucha experiencia en programación. Esto ha traído como consecuencia que el desarrollo de aplicaciones para móviles esté principalmente enfocado a Android y es por eso que actualmente hay una enorme cantidad de aplicaciones que se pueden obtener fácilmente en la tienda de aplicaciones de Android, *Play Store* o en sitios de terceras entidades.

La plataforma de Android es compleja, Drake et al. [3] mencionan que, debido a esta complejidad, es difícul realizar una explotación (programa o *script* que toma ventaja de un *bug* o una vulnerabilidad en un sistema o equipo computacional, provocando un comportamiento anormal en el mismo para ganar su control de manera parcial o total)[4] para Android. Eso implica que la seguridad en los dispositivos móviles con Android es más compleja que en las computadoras personales; sin embargo, los primeros están expuestos a casi los mismos riesgos que los segundos [4].

Asimismo, a diferencia del sistema iOS que solamente permite la descarga de las aplicaciones exclusivamente desde la *Apple Store*, lo cual en teoría conlleva a un incremento en la seguridad, en Android las aplicaciones se pueden obtener de forma legal tanto de la tienda oficial (*Google Play*) como de terceras fuentes. Las aplicaciones que provienen de terceras fuentes no están aprobadas por Google, lo que implica que esa aplicación puede no ser confiable; sin embargo, aunque la aplicación sea legítima no implica que sea completamente segura [5].

Del mismo modo, Apple tiene el control absoluto sobre iPhone y su sistema iOS, por consiguiente, las actualizaciones que Apple realiza a este sistema, las reciben todos sus dispositivos. Dentro de estas actualizaciones, se incluyen parches y mejoras en la seguridad.

En cambio con Android, quien tiene el control del software y hardware de los dispositivos no es Google, sino los fabricantes. Por lo tanto, no todos los dispositivos pueden recibir las actualiza-

[4]https://en.wikipedia.org/wiki/Exploit_(computer_security) consultado el 14 de noviembre de 2016

ciones de Google, al menos no al mismo tiempo. Esto trae como consecuencia que las *explotaciones*[5] y vulnerabilidades que afectan a una o más versiones no sean corregidos en los teléfonos o tabletas que son vulnerables.

Además, la actitud de los usuarios que utilizan Android hacia la seguridad de sus dispositivos es abrumadoramente indiferente. Por ejemplo, el estudio realizado por Robinson et al. [6] toma en cuenta al mal uso de los permisos como uno de los principales riesgos de seguridad asociados a Android; de acuerdo a los resultados de este estudio, los usuarios tienen un entendimiento relativamente pobre de qué permisos se activan en una aplicación. Originalmente no se podía hacer nada al respecto, ya que hasta la versión 5, al instalar una aplicación, debían de aceptarse todos los permisos o no se instalaba la aplicación, posiblemente esto pudo haber contribuído a una pobre cultura de seguridad; sin embargo, se requieren estudios al respecto.

Cabe destacar que en marzo de 2017 WikiLeaks lanzó un comunicado mencionando que durante los últimos años, la CIA ha traspasado las barreras de seguridad de computadoras de todas las plataformas de escritorio como Windows, Mac, Linux y plataformas móviles como iOS y Android, además de televisiones Samsung y sistemas de control de vehículos inteligentes dándole la capacidad de espiar a los usuarios. En el caso de Android, la División de Dispositivos Móviles o *Mobile Devices Branch (MDB)* de la CIA logró saltar y romper los esquemas de cifrado del sistema de Google con explotaciones del día cero o *zero-day*.

Con esto, la MBD tiene la capacidad de *hackear* y controlar los teléfonos inteligentes con Android permitiéndole obtener datos importantes como la geolocalización de los dispositivos, activar la cámara, el micrófono, obtener las comunicaciones de texto y el tráfico de mensajes de aplicaciones como WhatsApp, Signal, Telegram y Wiebo [7].

Asimismo, en 2017 la agencia de seguridad Citizen Lab, confirmó que los teléfonos inteligentes de activistas mexicanos tenían rastros de Pegasus, un malware creado por NSO para espiar dispositivos con iOS y Android, utilizando como medio de infección el *spear phishing*. Si el ataque tiene éxito, el malware es capaz de monitorear gran parte de las actividades que el usuario realice en su teléfono y la información obtenida es enviada a un servidor. En el apéndice A, se aborda con más detalle este *malware*.

[5] del inglés *Exploit*

1.1. Planteamiento del problema

El entorno de los móviles con Android puede ser un arma de doble filo, porque si bien estos dispositivos ofrecen una amplia gama de funcionalidades, esto trae como efecto que los usuarios almacenen datos personales e información susceptible que, al menor descuido, pueden quedar comprometidos; esto representa una amenaza a su seguridad y a su privacidad.

Dentro de estas amenazas a la seguridad, se encuentran el *malware* que se encuentra en las aplicaciones, mantener encendidos los servicios de Wi-Fi y Bluetooth en cualquier lugar, conectarse a redes Wi-Fi públicas y las vulnerabilidades existentes, tanto en el sistema como en las aplicaciones; es decir, que la inseguridad se halla en la misma tecnología que hace posible su funcionamiento. Esto ha motivado a entidades no autorizadas (intrusos) a realizar ataques a estos dispositivos para obtener información de los usuarios que potencialmente sea útil para otros fines.

Es posible recuperar información relacionada con el usuario, aunque ésta se haya eliminado, utilizando herramientas de código abierto y técnicas de cómputo forense. Dado que la mayoría de los movimientos, que realiza el usuario, se convierten a datos y éstos se guardan en bases de datos SQLite [1] cuando se elimina información de alguna de estas bases de datos, el espacio en el disco donde está esa información no está perdido, solamente ya no es devuelto al Sistema Operativo.[6]

De igual forma también es posible obtener información de un usuario incluso después de haber restablecido al dispositivo con los datos de fábrica [8]. En resumen, con el hecho de tener un dispositivo móvil, ya se está exponiendo en menor o mayor grado la privacidad del usuario.

1.2. Objetivos

Objetivo General

Se desea analizar la información del usuario que se puede obtener de un dispositivo Android y evaluar su privacidad y seguridad.

Objetivos Particulares

- Conocer los trabajos relacionados a la obtención de datos del usuario y al análisis de seguridad y privacidad de Android.

[6] https://sqlite.org/faq.html consultado el 03 de noviembre de 2016

- Diagnosticar la eficacia del servicio de confidencialidad, haciendo un estudio de la criptografía que ofrece Android en sus medios de almacenamiento secundarios.
- Investigar las vulnerabilidades de la arquitectura de Android y los ataques en los que se han obtenido información personal.
- Conocer los datos que se pueden conseguir a través de la inspección de paquetes en los servicios WiFi y Bluetooth.
- Recuperar información del usuario eliminada usando técnicas de tallado de archivos (*data carving*).
- Evaluar la seguridad de los navegadores que traen preinstalados los dispositivos y de la API de navegación exponiéndolos a ataques de husmeo.
- Descubrir nuevas vulnerabilidades en el sistema Android.

1.3. Metodología

Primero se realizó un estudio del estado del arte, es decir, una investigación del trabajo relacionado al tema de tesis, a través de la consulta de artículos científicos, libros, sitios y foros en Internet afines a Android, Seguridad de la Información y Criptografía.

Se hizo un estudio de los dispositivos móviles, particularmente de los teléfonos inteligentes, que comprende la investigación de su software y hardware y un estudio general del sistema operativo Android, es decir, se estudió su arquitectura, sus aplicaciones, cómo funciona y las versiones que se utilizan en México a través de consulta en libros, revistas y en sitios de Internet relacionados al tema.

Posteriormente se estudió el modelo de Seguridad de Android consultando libros, artículos científicos y sitios de Internet. A su vez, se investigó la privacidad de Android hacia los usuarios, consultando las advertencias de privacidad de los fabricantes de dispositivos Android.

Enseguida se hizo un estudio de cómo es la secuencia del borrado seguro o restablecido de fábrica del dispositivo. Después se realizó un análisis de la generación y almacenamiento de las llaves de cifrado y de los certificados digitales del dispositivo para determinar el grado de fiabilidad de los servicios de seguridad, haciendo un estudio en la literatura al respecto e investigando cómo se realiza el proceso de cifrado y descifrado.

A continuación se hicieron pruebas experimentales, llenando los dispositivos con información personal ficticia. Se eliminó esa información de dos formas, una de forma manual o normal y la otra con el borrado seguro.

Inmediatamente estos datos se trataron de recuperar, utilizando métodos forenses a través del uso de técnicas de *data carving* (tallado de archivos), técnicas que consisten en explorar secuencialmente el soporte de datos en busca de cadenas de caracteres características de cada tipo de archivos [4], usando herramientas como `foremost` versión 1.5.7, `scalpel` versión 1.60 y `bulk_extractor` versión 1.6.0.

Después se analizó qué información se puede recuperar del servicio de Bluetooth, realizando transferencias de datos entre dos dispositivos, eliminando la información transferida y se determinó qué se pudo recuperar de esa información. Seguidamente se realizó una evaluación del uso de aplicaciones de algunos fabricantes que preinstalan en sus dispositivos midiendo la cantidad de CPU, de memoria, de batería y de red que consumen.

A su vez se configuraron puntos de acceso maliciosos para realizar ataques de husmeo ”intruso de en medio” en el servicio de WiFi y se analizó qué información en claro generan y reciben los dispositivos al utilizar este servicio, haciendo uso de las aplicaciones que vienen preinstaladas por los fabricantes.

También, se realizó un estudio de los navegadores que vienen instalados de algunos fabricantes, para evaluar algunos de los sitios web más visitados en México que usan el protocolo SSL/TLS. Se hicieron experimentos con estos sitios para ver cuáles pueden ser forzados a utilizar http haciendo ataques de husmeo y así recuperar información confidencial, particularmente credenciales de autenticación.

Similarmente, se hizo una aplicación utilizando la API de navegación de Android para probar la seguridad del motor de renderizado web (WebKit y Chromium) que trae el sistema. Con esta aplicación se visitaron los mismos sitios y durante esas visitas, se realizaron ataques de husmeo.

Finalmente, se realizó un análisis de los *front-end* de WordPress en los navegadores de Android y en la misma aplicación, instalando un sitio web de WordPress en un servidor y utilizando como mecanismo de seguridad el protocolo TLS. Con este sitio se realizaron ataques de husmeo para saber si se podía recuperar información en claro.

1.4. Estado del arte

El trabajo de Ciurana et al. [9] hace un estudio de algunos mecanismos de seguridad en Android, enfocándose en la aplicación WhatsApp, la conectividad USB, los peligros de conectar los teléfonos en máquinas externas y la seguridad del control de acceso. También menciona que los métodos de seguridad de los dispositivos es mejorable y que para detectar las vulnerabilidades de las aplicaciones es necesario establecer una metodología general.

Simon et al. [8] hace un análisis referente al reinicio de datos de fábrica de Android, buscando si es posible recuperar datos que estaban almacenados antes del citado reinicio. Para eso realizó estudios en 21 teléfonos inteligentes con las versiones 2.3.x hasta la 4.3. Los resultados mostrados por Simon et al. muestran que sí se pueden recuperar datos a pesar de este borrado, se recuperaron archivos `SQLite`, algunas conversaciones (SMS y correos electrónicos), *tokens* de autenticación de Google y contactos. Este trabajo resalta las fallas críticas como la falta de soporte por parte tanto del sistema operativo como de los vendedores de teléfonos inteligentes hacia el borrado seguro; esto es, que hay fallos en la implementación del borrado de la información en la memoria flash y supone que también se debe a la personalización que los fabricantes imponen en sus dispositivos.

La investigación realizada por Ntagonian et al. [10] hace la evaluación experimental de la privacidad de los dispositivos móviles Android, utilizando herramientas de análisis forense de código abierto para verificar si se pueden encontrar credenciales de autenticación en la memoria volátil. Se realizaron experimentos en 13 aplicaciones para ver si se pueden descubrir patrones y expresiones que apunten a la posición de las credenciales.

Este estudio reveló que la mayoría de las aplicaciones evaluadas son vulnerables a la recuperación de tales credenciales desde la memoria volátil. Dentro de estas aplicaciones se incluyen aquellas que toman la seguridad como prioridad, tales como aplicaciones bancarias. La memoria volátil contiene credenciales de autenticación, a menos que se reinicie el dispositivo o que se remueva la batería [10].

De manera similar Apostolopoulos et al. [11] demuestra que las credenciales de autenticación pueden ser descubiertas en la memoria volátil de Android, utilizando herramientas gratuitas como el `Dalvik Debug Monitor Server (DDMS)`. Se realizaron experimentos en dos escenarios con 30 aplicaciones, de las cuales en 29 de ellas se recuperaron las contraseñas.

El trabajo realizado por Skorobogatov [12] muestra un ataque *mirroring* basado en hardware a

un dispositivo iPhone 5c, utilizando un equipo de electrónica de bajo presupuesto. En este ataque se desolda el chip NAND Flash de la tarjeta principal de un iPhone para crear un clón de este chip y realizar ingeniería inversa, obteniendo como resultado el rompimiento del límite de intentos que permite iOS para introducir la contraseña en el dispositivo. Como este método de autenticación es un número de 4 dígitos, solamente existen 10^4 posibilidades, haciendo factible obtener la contraseña del usuario con un ataque de fuerza bruta en sólo unas horas.

Otro método para vulnerar la seguridad de los usuarios de dispositivos Android es implementado en el trabajo realizado por Eom et al. [5] que muestra el riesgo de conectar los teléfonos inteligentes a puntos de acceso inalámbricos públicos o desconocidos porque en estos entornos un ataque de intruso de en medio (*Man In The Middle Attack*) es posible. Los resultados de Eom et al. demuestran que un intruso puede interceptar la comunicación en el dispositivo y puede insertar código malicioso en la instalación de aplicaciones genuinas.

Choi et al. [13] afirman que existen cuatro modos de fuga de información en Android y estos son: *smishing (SMS+Phishing)*, ingeniería social, factores físicos (pérdida del móvil) e Internet inalámbrico. Es por eso que propone un modelo de control de acceso basado en la inferencia para la detección y protección de fuga de información personal causados por código malicioso via contexto de ontología de los permisos de acceso.

1.5. Criptografía

Debido a los riesgos y amenazas a la seguridad que enfrentan varios mecanismos del sistema Android, para administrar el uso de la información, es necesario tener en consideración los siguientes servicios de Seguridad.

- **Confidencialidad.** Significa que la información únicamente puede ser accedida y manipulada por las entidades autorizadas.
- **Autenticación.** Asegura que las entidades que establecen la comunicación son quien dicen ser. Esto se logra con diferentes mecanismos como: firmas y certificados digitales o características biométricas.
- **Integridad.** Avala que los datos no han sido modificados por entidades no autorizadas.
- **Disponibilidad.** Garantiza que las entidades autorizadas tengan acceso a la información y a su uso.

- **No repudio.** Impide que una entidad niegue que realizó una acción, como por ejemplo, haber transmitido un mensaje.

Para efectuar estos servicios, la Criptografía juega un papel crucial como una herramienta para defenderse de las amenazas a la seguridad. La Criptografía es una disciplina que, por medio de técnicas matemáticas, se encarga de la transmisición segura de información entre dos entidades o usuarios a través de un canal inseguro de comunicación en presencia de adversarios maliciosos, para evitar que ellos puedan ver dicha información o hacer mal uso de ella.

En Criptografía, el mensaje transmitido es llamado *texto en claro*. A la acción de codificar el mensaje de forma que no pueda ser leído por adversarios maliciosos se denomina *Cifrar*. El mensaje cifrado se le conoce como *texto cifrado*. *Descifrado* es el proceso de obtener el *texto en claro* a partir del *texto cifrado*. Para cifrar y descifrar, es necesario el uso de una *llave*. De acuerdo con [14], la Criptografía puede ser clasificada en tres ramas: Criptografía Simétrica o de Llave Privada, Criptografía Asimétrica o de llave Pública y en Protocolos Criptográficos.

1.5.1. Criptografía simétrica

Esta Criptografía es un modelo que consiste básicamente en la utilización de una misma llave para el cifrado y el descifrado de la información que debe mantenerse en secreto. Hoy en día es ampliamente utilizada para realizar cifrado de datos y la revisión de la integridad de los mensajes [14] Para cifrar los mensajes, se utilizan dos esquemas de cifrado; el *cifrado por flujo* y el *cifrado por bloques*.

- **Cifrador por flujo.** Realiza el cifrado de los datos bit a bit. Un ejemplo de este tipo de cifrados es el RC4, que es utilizado en comunicaciones HTTPS por algunos navegadores preinstalados de Android.
- **Cifrador por bloques.** Se encarga de procesar la información por grupos de bits de longitud fija denominados bloques. Para cifrar los datos, se utilizan los modos de operación, que permiten el uso repetido y seguro de un cifrador por bloques con una sola llave. Los modos de operación más utilizados son el *Cipher Block Chaining* (CBC), el *Electronic Codebook* (ECB), *Cipher Feedback* (CFB), *Counter* (CTR) y *Galois Counter Mode* (GCM).

Los algoritmos de cifrado simétrico mas conocidos son el DES (*Data Encryption Standard*), su variante 3DES y AES (*Advanced Encryption Standard*) [15]. En la actualidad, DES ya no es seguro porque es vulnerable a ataques de fuerza bruta, debido a su espacio de llaves pequeño, 3DES es más seguro pero en general es lento y se recomienda su desuso [16]. Por el contrario AES es más

seguro, ya que su espacio de llaves es mucho más grande al contar con una seguridad de 128 a 256 bits.

Función picadillo

También conocidas como función *hash* o de resúmen, es una primitiva criptográfica que no utiliza llaves y que produce una cadena de bits de longitud fija a partir de un mensaje. A esta cadena se le denomina *resúmen* de un mensaje, que puede ser considerado como su huella digital. Algunos autores clasifican a la función picadillo como una rama de la criptografía, mientras que otros las ubican dentro de la criptografía simétrica.

Para que una función picadillo sea segura, deben cumplir con las siguientes propiedades [14]:

- **Debe ser resistente a preimágen.** Deben ser de un sólo sentido, esto significa que dada una salida r, debe ser computacionalmente difícil encontrar un mensaje m, tal que $r = h(m)$.
- **Debe ser resistente a segunda preimágen.** Significa que debe ser computacionalmente difícil obtener dos mensajes distintos m_1 y m_2, tal que $m_1 \neq m_2$, generen el mismo resúmen $r_1 = h(m_1) = h(m_2) = r_2$.
- **Debe ser resistente a colisiones.** Esto es, debe ser computacionalmente difícil crear dos mensajes $m_1 \neq m_2$ tales que generen el mismo picadillo $h(m_1) = h(m_2)$.

Algunos ejemplos de funciones picadillo son MD5 y SHA-1, que son considerados inseguros porque ya no son resistentes a colisiones, pero también están las funciones SHA-256, SHA-384 y SHA-512, las cuáles aún cumplen con las tres propiedades.

1.5.2. Criptografía Asimétrica

Se basa en problemas matemáticos difíciles de resolver y se diferencia de la Criptografía Simétrica porque se utilizan dos llaves, una para cifrar (llave pública) y la otra es para descifrar (llave privada). La llave pública es del conocimiento de todos los usuarios mientras que la privada, como lo dice su nombre, es de carácter privado, ya que solamente es conocida por la entidad propietaria. Ambas llaves se relacionan porque la llave pública es derivada a partir de la llave privada. De esta forma, a cada entidad le corresponde un par de llaves, donde la llave pública es distribuída a todas las entidades, de modo que a un grupo de n entidades, sólo se necesitarán un par de llaves por entidad.

Dentro de los principales esquemas criprográficos de llave pública se encuentran RSA (Rivest, Shamir y Adleman), ElGamal y la Criptografía de Curvas Elípticas. La Criptografía de Llave Pública se utiliza sobre todo para cifrar y para firmas digitales.

Firma digital

La Firma digital es una primitiva criptográfica que emplea criptografía asimétrica que se utiliza para proporcionar los servicios de autenticación, integridad y no repudio. Su propósito es brindar los medios necesarios a una entidad para vincular su identidad con una pieza de información [17]. Además, cualquiera puede verificar la firma sin necesidad de conocer la llave privada del firmante. Para obtener una firma digital se ocupa un algoritmo de firma, como DSA, que da como salida un número que es generado a partir de la llave privada del firmante y del contenido del mensaje firmado.

1.5.3. Protocolos criptográficos

Son algoritmos distribuidos que determinan las acciones que deben de realizar las entidades para lograr un objetivo específico de seguridad. Para lograrlo, se ocupan esquemas de cifrado simétricos y asimétricos, firmas digitales, funciones picadillo, entre otras primitivas. Los protocolos son importantes porque ofrecen los servicios de autenticidad, confidencialidad, integridad y no repudio [17].

Un protocolo puede ser vulnerado si un atacante logra manipularlo, en consecuencia el protocolo fallará en alcanzar los objetivos para los que fue diseñado. Las principales causas que originan las vulnerabilidades en los protocolos son la debilidad en alguna primitiva criptográfica y haber establecido garantías de seguridad asumidas que no fueron completamente entendidas.

1.6. Organización del documento

El resto de la presente tesis está organizada en 5 capítulos. El Capítulo 2 aborda una visión general del sistema operativo Android, mencionando de manera breve sus origenes y las versiones existentes a la fecha. También, se da una explicación de su arquitectura, se habla de los servicios del sistema, haciendo énfasis en los servicios de comunicación como WiFi y Bluetooth. Además, se explican la estructura de las aplicaciones, los tipos de aplicaciones, las particiones en su medio de almacenamiento externo y los sistemas de archivos que soporta.

El Capítulo 3 describe la seguridad y privacidad en Android, abordando su modelo de seguridad, los usuarios y los permisos. Del mismo modo, se describen los mecanismos de cifrado en sus medios de almacenamiento, el firmado del código en las aplicaciones, los certificados en el dispositivo También, se da una explicación del borrado de información del dispositivo tomando en cuenta el borrado normal como el borrado seguro, es decir, el restablecimiento de fábrica.

Posteriormente, en el Capítulo 4 se presentan los ataques que se han hecho en Android, particularmente aquellos dirigidos a los servicios de comunicación y a la recuperación de información que se eliminó del dispositivo. De igual forma, se da una breve explicación de las escalas de privilegios (*rooting*).

A continuación, en el Capítulo 5 se explica el desarrollo y experimentación de los ataques que se realizaron en dispositivos Android. Se realizaron ataques de husmeo a las aplicaciones preinstaladas de algunos fabricantes, recuperación de información eliminada tanto del modo normal como del modo seguro. También se analizan los mecanismos de comunicación de algunos fabricantes, la seguridad del navegador Google Chrome, el navegador de AOSP y de la API de navegación (webview). De la misma manera, se describe la recuperación de las bitácoras del bluetooth y se muestra qué información del usuario se pudo recuperar. Finalmente, en el Capítulo 6 se enlistan las conclusiones de este trabajo de investigación.

Capítulo

2

Visión general de Android

"Dime y lo olvido, enséñame y lo recuerdo, involúcrame y lo aprendo."

Benjamin Franklin

El objetivo de este capítulo es explicar los fundamentos y los principales mecanismos del sistema operativo Android. La Sección 2.1 habla de lo que es Android, sus orígenes y las versiones que han salido hasta la escritura de esta tesis. La Sección 2.2 se refiere a cómo está estructurada la Arquitectura de Android. Las Secciones 2.3 y 2.4 abordan los servicios del sistema y los servicios de comunicación, respectivamente. En la Sección 2.5 explican los diferentes componentes de una aplicación. La Sección 2.6 se presentan los diferentes tipos de aplicaciones. Y finalmente, en la Sección 2.7 se presentan los sistemas de archivos y las particiones que soporta el medio de almacenamiento secundario de los dispositivos Android, la memoria flash.

2.1. ¿Qué es Android?

Android es un sistema operativo de código abierto basado en el kernel de Linux y está desarrollado por Google y la *Open Handset Alliance* (OHA), es una plataforma diseñada principalmente para dispositivos móviles tales como teléfonos inteligentes y tabletas. Aunque en los últimos años también se ha destacado por su implementación en otros dispositivos empotrados como relojes inteligentes, cajas registradoras, televisores inteligentes, entre otros.

NÚMERO DE VERSIÓN	NOMBRE CÓDIGO	NIVEL DE API	AÑO DE LANZAMIENTO
1.0	Alpha	1	2008
1.1	Beta	2	2009
1.5	Cupcake	3	2009
1.6	Donut	4	2009
2.0-2.1	Eclair	5-7	2009
2.2-2.2.3	Froyo	8	2010
2.3-2.3.7	Gingerbread	9-10	2010
3.0-3.2.6	Honeycomb	11-13	2011
4.0-4.0.4	Ice Cream Sandwich	14-15	2011
4.1-4.3.1	Jelly Bean	16-18	2012
4.4-4.4.4	KitKat	19	2013
5.0-5.1.1	Lollipop	21-22	2014
6.0-6.0.1	Marshmallow	23	2015
7.0-7.1.1	Nougat	24-25	2016
8.0	Oreo	26	2017

Tabla 2.1: Versiones de Android

Los orígenes de Android se remontan al año 2005, cuando la compañía llamada Android Inc., que se dedicaba al desarrollo de aplicaciones para móviles, fue adquirida por Google [18]. En el 2007 se forma la Handset Alliance, un consorcio de diferentes fabricantes de equipos y software de telecomunicaciones, liderada por Google, dentro de sus funciones destacan el desarrollar estándares para dispositivos móviles y promover la unificación de la plataforma Android.

En 2007 la OHA anunció la salida de Android. Pero no es sino hasta el año 2008 cuando salió la primera versión comercial llamada Android 1.0 con el nómbre código *Alpha*. A partir de entonces, han salido diferentes versiones de esta plataforma por lo general cada año y a partir de la versión 1.5, su nombre código tiene el nombre de una golosina en inglés en orden alfabético.

Además, cuando una versión es lanzada, también se publican de una a tres APIs acorde a la versión que salió, las APIs son utilerías que los desarrolladores pueden usar sin tener que interactuar con las capas más bajas. La versión de Android más reciente al momento de la escritura de esta tesis es la 8.0 con el nómbre código *Oreo*. A continuación en la Tabla 2.1 se muestran las diferentes versiones de Android hasta el momento de la escritura de esta tesis:

Usualmente Google reporta la distribución de versiones a nivel mundial en el sitio developer.android.com. De acuerdo al reporte de agosto de 2017, como se muestra en la figura 2.1, la versión más utilizada es *Marshmallow* con un porcentaje de distribución de 32.3 %, seguido por la versión 5.1 con un 29.2 %.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.7%
4.1.x	Jelly Bean	16	2.7%
4.2.x		17	3.8%
4.3		18	1.1%
4.4	KitKat	19	16.0%
5.0	Lollipop	21	7.4%
5.1		22	21.8%
6.0	Marshmallow	23	32.3%
7.0	Nougat	24	12.3%
7.1		25	1.2%

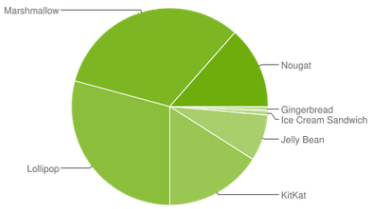


Figura 2.1: Distribución de versiones de Android en Agosto de 2017 [1].

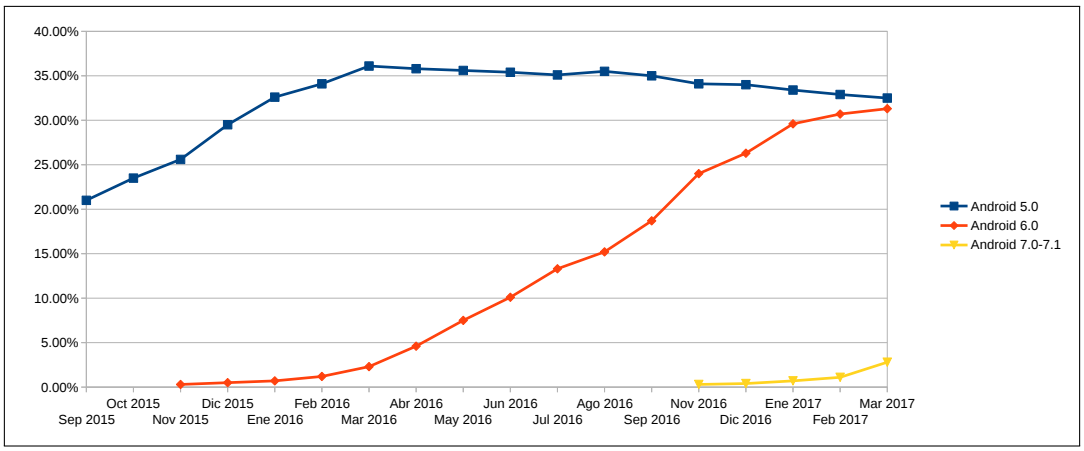


Figura 2.2: Proyección de distribución de las últimas tres versiones de Android.

Cabe notar que por lo regular casi siempre la penúltima y antepenúltima versión reciente son las más utilizadas. Además, la última versión por lo general alcanza el mayor porcentaje entre 3 a 8 meses después de haber salido su versión sucesora. Para ilustrar esto, en la gráfica de la figura 2.2 se muestra la distribución de las últimas 3 versiones (5, 6 y 7) de Septiembre de 2015 a Marzo de 2017. En base a las proyecciones mostradas de la figura 2.2, se puede inferir que para finales de 2017 o inicios de 2018, la versión 7 alcance el primer lugar de distribución.

La mayor parte de la plataforma Android es de código abierto y es desarrollada bajo el uso del *Android Source Open Project (AOSP)*. Este proyecto es apoyado por una comunidad de desarrolladores quienes se encargan de incluir nuevas características, mejoras y correcciones. El AOSP se vale de

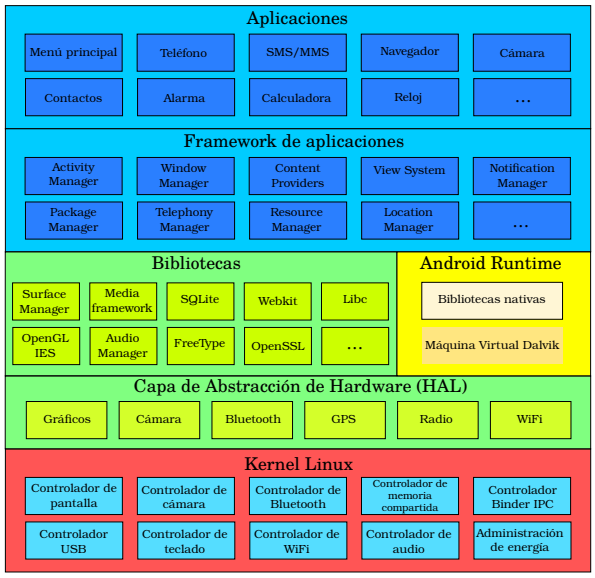


Figura 2.3: Arquitectura de Android

dos licencias de código libre, la Licencia de Software Apache 2.0 (ASL 2.0) y la Licencia General Pública GNU, versión 2 (GPLv2) [4].

Por otro lado, existen componentes de esta plataforma que son de código cerrado. Partes como el cargador de arranque, el firmware periférico, componentes de radio y algunas aplicaciones [3] son algunos ejemplos que contienen código cerrado. Esto es con el objetivo de proteger la propiedad intelectual de los fabricantes y los operadores de telefonía móvil, incluyendo Google.

No obstante, también esto representa un obstáculo a la comunidad del AOSP, ya que en muchas ocasiones esto ha traído como consecuencia que el código fuente de una versión sea publicado mucho tiempo después del lanzamiento de esa versión dada, siendo que esto va contra los principios del software de código abierto.

2.2. Arquitectura del sistema

Android se constituye de una serie de componentes que se ajustan a una pila o un modelo de capas. Este modelo se divide en 6 categorías principales en el orden del más bajo al más alto nivel: capa del kernel Linux, capa de abstracción del Hardware, capa de las Bibliotecas, capa del entorno de ejecución (donde está la Máquina Virtual Dalvik (DVM o MVD)), capa de los *Frameworks* y capa de aplicaciones, tal y como se ilustra en la figura 2.3.

2.2.1. Capa del kernel Linux

Como se había mencionado en la Sección 2.1, Android está basado en el kernel de Linux, y éste representa la capa más baja de este modelo. En parte está integrado por las partes que componen los núcleos Linux de la mayoría de las distribuciones de escritorio como Debian o Red Hat, por mencionar algunas; sin embargo, presenta algunas modificaciones realizadas por Google para que pueda adaptarse mejor a los entornos móviles [19].

Esta capa tiene como función gestionar las tareas del sistema como manejo de memoria, administración de los procesos, de la seguridad y de la red. Representa una abstracción del hardware y a través de sus controladores, permite la comunicación entre las capas superiores y el hardware como por ejemplo, la pantalla, la cámara, el WiFi, el Bluetooth, entre otros [20]. El kernel está escrito en código nativo, es decir, que está programado en C y C++.

Binder IPC

Dentro de la estructura de la capa del kernel destaca el Binder IPC (Comunicación Entre Procesos, *Inter Process Comunication*), que es un componente de código abierto que se encarga de gestionar la comunicación y el intercambio de datos entre los procesos del sistema. Así, las aplicaciones pueden interactuar directamente con los servicios del sistema [21]. Binder se fundamenta en el principio del IPC de los sistemas basados en Unix y también está basado en el proyecto OpenBinder de Palm Inc., pero tiene ciertas modificaciones para que pueda adaptarse al entorno Android.

Por razones de seguridad, Binder mantiene a los procesos aislados unos de otros, es decir, que un proceso no puede acceder directamente a la memoria de otro proceso. En lugar de eso, es el kernel quien se encarga de controlar los procesos y de las llamadas IPC [22], esto se realiza con llamadas a la función del lenguaje C `ioctl()`.

Por tal motivo, cuando un proceso A desea comunicarse con otro proceso B, el proceso A envía un mensaje que llega al kernel. El kernel copia el mensaje, se lo envía al proceso B, éste accede al mensaje y así se lleva a cabo la comunicación y la transferencia de datos. Con la seguridad aportada por este mecanismo, se previenen ataques como el escalado de privilegios.

2.2.2. Capa de abstracción de hardware

Un componente esencial dentro de Android es la HAL o Capa de Abstracción de Hardware (*Hardware Abstraction Layer*). Programada en código nativo, la HAL es una interfaz estándar que permite a Android entrar en contacto directo con el hardware. Los componentes que se encargan de comu-

nicarse con la HAL son los controladores de la capa del kernel. Esta interfaz facilita a los fabricantes la adaptabilidad de Android con su hardware.

La HAL está constituida y empaquetada en un conjunto de archivos de módulos de bibliotecas compartidas ".so" [23], tales módulos pueden ser de la AOSP o del fabricante, dependiendo del dispositivo. Cada elemento del hardware tiene su respectiva definición de HAL así como su servicio del sistema [24]; por ejemplo, hay una definición de HAL de WiFi y un servicio del sistema de WiFi, como se verá más adelante en este Capítulo.

HAL está estructurado por dos componentes: un módulo y un dispositivo. Un módulo contiene los metadatos como el nombre, la versión y el autor del módulo. Así, Android puede localizar la definición de HAL y cargarlo correctamente [23]. Un dispositivo se ocupa de abstraer un componente de hardware, por ejemplo un módulo de la cámara puede tener un dispositivo USB de cámara y un dispositivo Bluetooth de cámara.

2.2.3. Capa de las bibliotecas

Las bibliotecas o bibliotecas nativas son un conjunto de servicios escritos en código nativo, que se encargan de gestionar las diferentes tareas del software y son utilizadas por los diferentes elementos de Android de las otras capas. Lázaro [4] menciona que son archivos que contienen código reutilizable para realizar tareas específicas.

Las aplicaciones acceden a las bibliotecas a través de llamadas de Java Native Interface (JNI) [21] y se comunican con la capa del kernel por medio de sus controladores. Android cuenta con alrededor de 100 bibliotecas, como por ejemplo WebKit, SQLite, OpenGL, Bluedroid, Bionic, SGL, Surface Manager, OpenSSL, entre otros.

2.2.4. Capa del entorno de ejecución

Por lo general, las aplicaciones de Android están escritas en Java. Por lo tanto, estas aplicaciones se compilan en archivos "`.class`" que generan un código intermedio llamado *Java bytecode* y que se ejecuta en una máquina virtual. Normalmente, los programas en Java se ejecutan en la máquina virtual de Java (MVJ).

En el caso de Android, por razones de rendimiento, hasta la version 4.4 el *Java bytecode* generado se convierte en *Dalvik bytecode* que se ejecuta en la máquina virtual Dalvik (MVD), la cuál está optimizada para consumo mínimo de recursos, dado que está en un entorno empotrado. Los

archivos `class` se convierten en un archivo `.dex/.odex` (Dalvik Executable/Optimized Dalvik Executable), con la ayuda de la utilería `dx` [20] para poder ejecutarse en la MVD (ver figura 2.4).

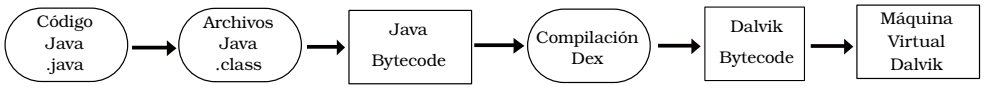


Figura 2.4: Procedimiento de la máquina virtual Dalvik

Dex es un formato de archivo ejecutable que está optimizado para almacenamiento eficiente [21], similar a los archivos ".jar" generados por la MVJ. Después de haberse generado el ".dex", se comprime en formato *Android package file* (.apk).

A partir de la versión 5, la MVD es sustituído por el *Android Run Time* (ART). El ART, es similar a la MVD, con algunas funcionalidades adicionales; por ejemplo, un *Garbage Collector* mejorado y compilado para las arquitecturas de 64 bits [25]. Otra diferencia yace en que Dalvik utiliza la compilación *Just In Time* (JIT, Justo a Tiempo), esto significa que la MVD compila el *bytecode* cada vez que la aplicación es lanzada. En cambio, ART usa la compilación *Ahead Of Time* (AOT, Antes de Tiempo), donde la compilación se realiza desde que la aplicación es instalada [20].

2.2.5. Capa de los frameworks

Es también llamada capa de los frameworks de las aplicaciones. En este nivel están las APIs, las cuáles están codificadas en Java y utilizan los recursos del sistema y del hardware que necesiten las aplicaciones. Desde aquí es donde se ejecutan y administran las aplicaciones. Está integrado por un conjunto de servicios o componentes, entre los que destacan los siguientes:

- ***Activity Manager.*** Se encarga de gestionar el ciclo de vida de la aplicación.
- ***Notification Manager.*** Permite que las aplicaciones muestren notificaciones o alertas cuando ocurre un evento.
- ***Resource Manager.*** Concede acceso a recursos de la aplicación que no están en código, como por ejemplo, gráficos, cadenas y mapas de bits.
- ***Content Provider.*** Facilita a las aplicaciones la capacidad de compartir datos con otras aplicaciones y también acceder a sus datos.
- ***Telephony Manager.*** Otorga el acceso a información de los servicios del teléfono, SMS y MMS.

- ***View System.*** Proporciona a las aplicaciones los componentes y eventos de la interfaz de usuario como botones, cuadros de texto, botones de radio, casillas de verificación, entre otros.
- ***Location Manager.*** Ofrece acceso y notifica a las aplicaciones de las actualizaciones de los servicios de localización como GPS.
- ***Package Manager.*** Es el encargado de instalar y remover aplicaciones, además de proporcionar información de cuáles aplicaciones están instaladas.

2.2.6. Capa de aplicaciones

Es la capa de nivel más alto. Aquí se encuentran las aplicaciones que interactúan con el usuario y todas ellas tienen la misma estructura, tanto las del sistema, como las del fabricante y las instaladas por el usuario, de las cuáles se hablarán más adelante en este capítulo. Las aplicaciones se ejecutan en la máquina virtual Dalvik. Ejemplos de aplicaciones son: el reloj, el correo electrónico, el calendario, el SMS, los contactos, la cámara, entre otras.

2.3. Servicios del sistema

Se encargan de poner en funcionamiento las diferentes propiedades de Android como telefonía, administración de la batería, apoyo para la pantalla táctil y la conectividad de red. Las aplicaciones y otros servicios pueden hacer uso de estos servicios a través de las interfaces remotas que éstos establecen [22]. Estos servicios ejecutan un Sistema Operativo orientado a objetos que está construído sobre Linux [24]. La mayoría de los servicios están programados en Java, pero también algunos están implementados en código nativo.

En los servicios del sistema están involucrados tres procesos. Por un lado está el proceso *System Server* que aloja servicios como la batería, la conectividad de red, la barra de estado, la ubicación, la búsqueda, el fondo de pantalla, entre otros. También está el proceso *Media Service* que se encarga de los servicios relacionados con multimedia como el audio y la cámara. Por otra parte, el proceso *Phone App* es el que se ocupa del servicio de telefonía.

De acuerdo con Yaghmour, estos servicios son interdependientes y los servicios del sistema que están hechos en la AOSP deben de estar disponibles todo el tiempo, dado que así lo suponen la mayor parte de los componentes de Android [24].

2.4. Servicios de comunicación

Android tiene a su disposición entre 60 a 80 distintos tipos de servicios del sistema; sin embargo, en esta tesis se hace énfasis a dos servicios que se orientan en la comunicación: el WiFi y el Bluetooth, ambos pertenecientes al proceso *system server*.

2.4.1. Servicio de WiFi

El WiFi es una tecnología que engloba hardware y software, y que tiene como funcionalidad la conexión inalámbrica de un anfitrión a Redes de Área Local (LANs) que pueden proveer acceso a Internet y están basados en el estándar IEEE 802.11. Los dispositivos Android tienen implementado el WiFi en un chip dedicado para este servicio, no obstante, este circuito integrado también puede estar destinado tanto para el WiFi como para el Bluetooth.

Elenkov [22] menciona que la arquitectura de WiFi en Android está integrada por una capa de adaptador de kernel, un demonio nativo llamado *wpa_supplicant*, la Capa de Abstracción de Hardware (HAL) y el servicio de sistema *WiFiService*. La capa del kernel incluye controladores del hardware y módulos de kernel. El demonio *wpa_supplicant* maneja la asociación del controlador WLAN 802.1X, además de administrar la negociación de llaves del protocolo de seguridad WPA. La HAL se encarga de restransmitir comandos desde la capa de los frameworks hasta el *wpa_supplicant* a través de su control de puertos.

El servicio de sistema *WiFiService* es el responsable de manejar los estados del WiFi, es decir, se encarga de deshabilitar y habilitar el WiFi, de las credenciales de autenticación, escanear las redes que se encuentran alrededor del dispositivo, calcular el nivel de la señal y de agregar y remover redes del dispositivo. También, obtiene las direcciones DHCP que han sido asignadas desde la última petición DHCP exitosa [26]. El servicio es gestionado por la API `WiFiManager` y se puede encontrar su documentación en https://developer.android.com/reference/android/net/wifi/WifiManager.html.

2.4.2. Servicio de Bluetooth

El bluetooth es una tecnología inalámbrica para el intercambio y transmisión de información entre dispostivos fijos y móviles en distancias cortas, creando un entorno de una Red de Área Personal (PAN).

Android tiene su implementación de la pila de protocolos Bluetooth con las bibliotecas Bluedroid y Bluez [27]. Esta pila se divide en dos capas: la primera es el Sistema Empotrado Bluetooth (BTE,

the Bluetooth Embedded System) que lleva a cabo la funcionalidad del protocolo Bluetooth a bajo nivel y la segunda es la Capa de Aplicación Bluetooth (BTA, *Bluetooth Application Layer*) que se ocupa de comunicarse con las aplicaciones del framework de Android.

El servicio del sistema Bluetooth se comunica con la pila de Bluetooth a través de la JNI y con las aplicaciones por medio del Binder IPC. Este servicio, es implementado como una aplicación y está ubicado en el directorio `packages/apps/bluetooth` y pone en funcionamiento el servicio de bluetooth. Las aplicaciones hacen uso de este servicio a través de la API de bluetooth y su documentación está disponible en [28].

2.5. Estructura de las aplicaciones

Las aplicaciones Android están integradas por un conjunto de componentes que hacen posible su funcionamiento. A continuación se dará una explicación de cada uno de ellos.

2.5.1. Actividades

Una actividad (*activity* en inglés) es esencialmente el equivalente a una ventana en un sistema operativo de escritorio. Es una pantalla, una interfaz gráfica con la que el usuario puede interactuar con la aplicación [19], tal y como lo muestra la figura 2.5. Cada aplicación tiene una o más actividades, dado que no requieren de muchos recursos.

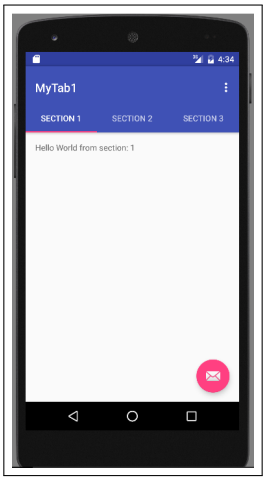


Figura 2.5: Actividad de una aplicación

Así, cuando una aplicación inicia, se ejecuta una actividad, que es la actividad principal. A partir de la actividad principal, el usuario puede explorar las otras actividades de la aplicación [19]; por ejemplo, en la aplicación de mensajería SMS/MMS está la actividad del listado de mensajes y otra actividad destinada a una conversación con un contacto o número telefónico específico.

2.5.2. Archivo manifiesto

Es un archivo XML que viene bajo el nombre de `AndroidManifest.xml` que está en texto plano durante el momento de la programación de la aplicación y cuando es compilado, se convierte a formato XML binario. Este archivo contiene información acerca de la aplicación, como el nombre del paquete de la misma, la versión SDK en la que fue programada, requerimientos de hardware y software [4], los componentes que la integran como actividades, servicios, Broadcast Receivers, definiciones de instrumentación [3] y metadatos adicionales.

Así también, incluyen los permisos que necesita la aplicación para su instalación y funcionamiento. Además, tiene directivas complementarias de apoyo, como la ubicación de la instalación e información de la interfaz de usuario [3]. En la figura 2.6 muestra un ejemplo del manifiesto.

```
11 11<?11xml11 11version11="1.0"11 11encoding11="11utf11-8"11?>
 <!--Etiqueta manifest, es el elemento raiz del archivo manifiesto, debe contener la etiqueta <aplication> -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.navegador.seguridad.navegacion6">
   <!--permiso de establecer conexion a Internet-->
   <uses-permission android:name="android.permission.INTERNET" />
   <!--permiso para gestionar descargas-->
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <!--etiqueta aplicacion, que se encarga de declarar los componentes de una aplicacion que tienen atributos-->
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <!--Declara una actividad que implementa la interfaz de usuario visual-->
      <activity android:name=".MainActivity"
         android:windowSoftInputMode="adjustPan"
         android:screenOrientation="portrait">
         <!--Especifica los tipos de intents que una actividad, servicio o broadcast receiver puede responder-->
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

Figura 2.6: Ejemplo de un archivo manifiesto.

2.5.3. Intentos

Los intentos (o *intents*, en inglés), son mensajes, que contienen paquetes de datos [4] y establecen comunicación entre las actividades, otros componentes de una aplicación y entre aplicaciones para realizar una acción solicitada por el intento. Los intentos son abstracciones de alto nivel de los IPC y están programados sobre Binder. Se ocupan para comenzar una actividad, para iniciar un servicio y para entregar un mensaje. Los intentos pueden ser de dos tipos: explícitos o implícitos [29].

Los intentos explícitos determinan qué componente debe activarse, es decir, ya se sabe a dónde irá dirigido el intento. Este tipo de intentos son útiles para los desarrolladores si es que diferentes partes de sus aplicaciones van a interactuar entre sí, ya que conocen los componentes de sus aplicaciones.

En cambio, los intentos implícitos no determinan el nombre del componente, no tienen un destino en específico, por lo tanto no se sabe hacia dónde van dirigidos. Cuando este intento es lanzado, Android busca qué aplicación debe recibir ese intento, tomando criterios en base a la acción definida por el mismo intento.

Si hay más de una aplicación que cumpla los criterios de la acción solicitada, Android le muestra al usuario ese conjunto de aplicaciones que pueden ejecutar esa acción y le ofrece dos opciones al escoger una aplicación; que la acción siempre se ejecute con la aplicación seleccionada o solamente una vez. Si elige la primera opción, tal aplicación se abrirá automáticamente la próxima vez que se ejecute la acción, convirtiéndose en una aplicación por defecto asociada a este intento. También sucederá lo mismo si sólo existe una aplicación que sea capaz de utilizar el intento.

2.5.4. Broadcast receivers

Este componente actúa como receptor de los mensajes del sistema o de las aplicaciones como los intentos. Los broadcast receivers permanecen inactivos hasta que reciben un mensaje porque ocurrió un evento y entonces realizan cierta acción [19]. A estos mensajes se les llama *broadcast*. Para que una aplicación pueda ejecutar un broadcast receiver, necesita ser declarada esta acción en el archivo manifiesto.

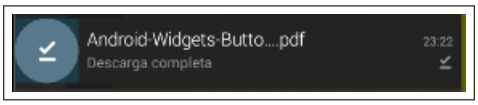


Figura 2.7: Broadcast receiver indicando que un archivo se ha descargado

Una aplicación puede enviar *broadcasts* a sí misma o a otras aplicaciones [19]. De la misma forma que los intentos, los broadcast receivers también son una abstracción de IPC. Ejemplos de broadcast receivers son la alerta de batería baja, un aviso de que un archivo ha sido descargado completamente de internet (ver figura 2.7) o que un archivo enviado por bluetooth ha sido recibido correctamente.

2.5.5. Proveedores de contenido

Originalmente llamado *content providers*, son mecanismos que autorizan a las aplicaciones el uso compartido de datos estructurados almacenados en el dispositivo, incluyendo aquellos que son privados. De acuerdo con la documentación de proveedores de contenido del sitio oficial de desarrolladores de Android, los proveedores de contenido son la interfaz estándar que conecta datos en un proceso con código que se ejecuta en otro proceso [30]. Con los proveedores de contenidos, otras aplicaciones pueden tener acceso a estos datos y también modificarlos.

Ya que esto podría implicar un riesgo en su seguridad, un proveedor de contenidos proveé mecanismos de seguridad para restringir el acceso a los datos, esto se hace en el archivo manifesto por medio de la etiqueta `<provider>` [19]. También, es posible leer y escribir datos privados de una aplicación sin necesidad de que estén compartidos [30]. Por lo regular, los proveedores de contenido están guardados en bases de datos SQLite o en un sistema de archivos.

2.5.6. Servicios

Los servicios como componente de una aplicación no son lo mismo que los servicios del sistema. Éstos son componentes que se ejecutan en segundo plano, es decir, que aunque el usuario no esté utilizando la aplicación, el o los servicios seguirán ejecutándose y de hecho, no poseen interfaz gráfica. Los servicios se ocupan para realizar operaciones durante un largo período de tiempo [19] y para comunicarse con otras aplicaciones.

Los demás componentes y otras aplicaciones pueden hacer uso de los servicios realizando llamadas al Binder IPC. Para que una aplicación pueda hacer uso de algún servicio, éste debe ser declarado en el archivo manifiesto, usando la etiqueta `<service>`. Ejemplos de servicios son la descarga de un archivo, un reproductor de música y la alarma.

2.6. Tipos de aplicaciones

Las aplicaciones son las entidades con las que hay una interacción directa entre el usuario y Android para realizar una función específica. Existen una amplia variedad de ellas y se pueden clasificar en aplicaciones del sistema, aplicaciones del fabricante y aplicaciones instaladas por el usuario.

2.6.1. Aplicaciones del sistema

Estas aplicaciones ya vienen instaladas en el sistema y son desarrolladas por la AOSP. Pueden ser parte del núcleo de Android o simplemente son aplicaciones preinstaladas [22] y están montadas en el directorio `/system` y también en `/system/priv-app`, ambas de sólo lectura.

Las aplicaciones en `/system/priv-app` tienen más privilegios que las otras aplicaciones, ya que poseen permisos con nivel de protección `SignatureOrSystem` lo que les permite realizar operaciones como manejo de eventos y administrar la política de red, los niveles de protección de permisos se verán en el Capítulo 3. Ejemplos de aplicaciones del sistema son el reloj, la configuración, llamadas/contactos telefónicos, correo electrónico y SMS, entre otros. Estas aplicaciones no pueden ser desinstaladas, pero el usuario puede actualizarlas o utilizar otra aplicación en su lugar [22].

2.6.2. Aplicaciones del fabricante

La mayoría de los fabricantes, operadores de telefonía móvil y Google, tienen sus propias aplicaciones que preinstalan en sus dispositivos antes de que lleguen a las manos del usuario, a este conjunto de aplicaciones se les conoce como *bloatware*. Por lo regular, cumplen las mismas funciones que las de AOSP y algunas de ellas poseen los mismos privilegios que las aplicaciones del sistema, algunos autores clasifican a algunas de estas aplicaciones dentro de las del sistema.

En varios dispositivos los fabricantes han sustituido varias aplicaciones de AOSP por las suyas. Habitualmente son de código cerrado y en muchas ocasiones no pueden ser desinstaladas, a menos que el usuario adquiera privilegios de *root*. Ejemplos de estas aplicaciones son el reproductor de música, album de fotos, radio FM, redes sociales, la tienda de aplicaciones del proveedor (ej. Samsung), entre otras.

2.6.3. Aplicaciones del usuario

Existe una gran variedad de estas aplicaciones, son instaladas por el usuario y se pueden instalar de tres formas. La primera es desde la tienda de aplicaciones Google Play, la segunda es a través

de terceras entidades como tiendas de aplicaciones no aprobadas por Google o instalándolas directamente desde el archivo apk. Tales aplicaciones son instaladas en la partición */data*, que es de lectura y escritura. Pueden ser desinstaladas, aunque tienen menos privilegios que las aplicaciones del sistema y que las del fabricante.

Hasta la versión 5, para que una aplicación pueda hacer uso de recursos del sistema operativo o del hardware, necesita solicitar permisos a través del archivo manifiesto, que son mostrados al usuario antes de instalar la aplicación. Esto cambia a partir de la versión 6, donde estos permisos se solicitan en tiempo de ejecución, justo antes de que la aplicación empiece a realizar una acción que requiera el uso de algún recurso. Ejemplos de estas aplicaciones son juegos, ofimática, redes sociales, entre otros.

2.7. Particiones y sistemas de archivos

Debido a la naturaleza empotrada de los dispositivos, en lugar de disco duro, el medio de almacenamiento externo de Android es la unidad Flash y usa la tabla de particiones GUID o también llamada GPT. El archivo `/proc/partitions` contiene una lista de las particiones que contiene un dispositivo.

La distribución de particiones en Android varía debido a que cada fabricante establece sus propias tablas de particiones; sin embargo, existen particiones que son comunes en todos los dispositivos. La mayoría se pueden montar, excepto la partición boot y recovery [25]. A continuación se explicarán las particiones más comunes.

- `boot`. Es la partición encargada de arrancar el sistema, dado que contiene elementos como el kernel e información sobre la RAM.
- `recovery`. Contiene información que permite entrar al modo de consola de recuperación al momento de arrancar el teléfono. En esta partición se encuentra almacenada una imagen de arranque de Android.
- `system (/system)`. Aquí están almacenados la mayoría de los componentes de Android como las bibliotecas, el framework de Android, archivos binarios del sistema, las aplicaciones del sistema, demonios, comandos de la *shell* y archivos específicos del vendedor.
- `userdata (/data)`. Se le llama partición de datos y es donde se almacenan los datos de las aplicaciones y la mayoría de los datos del usuario.

- `sdcard (/sdcard)`. Por lo regular en esta partición se guardan los archivos multimedia del usuario como fotos, imágenes, vídeos, archivos de ofimática. También, algunas aplicaciones generan directorios y archivos en esta partición, generalmente archivos multimedia. Además está la partición `/sdcard1`, que se asigna a la tarjeta de memoria externa y que cumple los mismos propósitos que la `/sdcard`.
- `cache (/cache)`. Almacena los datos y archivos que son accedidos con más frecuencia, como por ejemplo, las bitácoras de recuperación y paquetes de actualización.
- `radio`. Contiene información sobre las actividades y configuración de telefonía.
- `vendor (/vendor)`. Tiene archivos relacionados a las modificaciones del fabricante realizadas para Android.

Sistemas de archivos

Android soporta la mayoría de los sistemas de archivos de Linux y también de otras plataformas como MSDOS, unos más utilizados que otros. Los sistemas de archivos en Android se clasifican en tres grupos o categorías: sistemas de archivos basados en flash, basados en medios y pseudosistemas de archivos.

Los sistemas de archivos basados en flash en Android son los siguientes:

- `Extended File Allocation Table (exFAT)`. Sistema de archivos propietario de Microsoft optimizado para dispositivos flash.
- `Flash Friendly File System (F2FS)`. Este es un sistema de código libre de Samsung.
- `Journal Flash File System version 2 (JFFS2)`. Este sistema de archivos está estructurado como bitácora. Es el sistema por defecto de la AOSP desde la versión 4.3.
- `Yet Another Flash File System version 2 (YAFFS2)`. Está diseñado para adaptarse mejor a la memoria NAND flash. Del mismo modo que JFFS2, YAFFS2 también está estructurado como bitácoras, tiene un buen rendimiento para mantener la integridad de los archivos y era el más utilizado por los dispositivos Android; sin embargo, a partir de la versión 2.3 en el año 2010, Android comenzó a migrar de YAFFS2 a EXT4 y además, los kernels más recientes ya no tienen soporte para YAFFS2.
- `Robust File System (RFS)`. Tiene soporte para la memoria NAND flash en dispositivos Samsung y está basado en FAT16/FAT32. Sus desventajas subyacen en que tiene tiempos de

retraso en operaciones de lectura y escritura, lo que hace más lenta la realización de operaciones de Android y también presenta cuellos de botella en procesadores de doble núcleo.

Estos son sistemas de archivos basados en medios:

- `EXTended file system (EXT2/EXT3/EXT4).` Sistema de archivos virtual distintivo de Unix y Linux. Actualmente, muchos dispostivos Android tienen este sistema. Entre sus características destacan su estabilidad, su soporte para procesadores de doble núcleo y su eficacia al momento de un apagado inesperado del equipo, ya que si esto sucede, no es necesario volver a inspeccionar el sistema de archivos.
- `File Allocation Table (FAT).` Sistema de archivos que tienen por defecto los sistemas MSDOS.
- `Virtual File Allocation Table (VFAT).` Diseñado por Microsoft, es una extensión de los sistemas FAT16 y FAT32. Muchas tarjetas SD externas tienen FAT32 como sistema de archivos.

Cabe destacar que FAT y VFAT son patentes de Microsoft, quién ha declarado que Android infringe su patente al usar estos sistemas de archivos. Así que para solucionar este problema, algunos fabricantes le pagan a Microsoft entre 5 a 15 dólares de regalías por cada dispositivo Android vendido.

Por último dentro de los pseudosistemas de archivos se encuentran:

- `Control group (cgroup).` Previamente llamados contenedores, reúne grupos de control de procesos para limitar y aislar el uso de recursos (procesador, memoria, entre otros). Se encarga de añadir nuevos procesos y de seguir su rastro [20].
- `rootfs.` Está montado en el directorio raíz (/). Cuando Android realiza el proceso de arranque, tiene que montar el sistema de archivos raíz y la información necesaria para ejecutar este proceso está en este sistema de archivos. Los demás sistemas de archivos son montados a partir de rootfs.
- `procfs.` Tiene información del contenido del directorio `/proc` como procesos, estructuras de datos, kernel y otros aspectos del sistema.
- `sysfs.` Este sistema de archivos contiene información acerca de la configuración del dispositivo y está montado en el directorio `/sys`.
- `tmpfs.` Está montado en el directorio `/dev`, es un sistema de almacenamiento temporal y es responsable de contener los datos de la memoria RAM. Cuando el dispositivo se apaga o se reinicia, toda la información que se había guardado en `/dev` es borrada.

Capítulo

3

Seguridad en Android

"Nuestras virtudes y nuestros defectos son inseparables, como la fuerza y la materia. Cuando se separan, el hombre deja de existir"

Nikola Tesla

En este capítulo se presentan los distintos mecanismos de seguridad ofrecidos por Android. Primeramente, en la Sección 3.1 se explica cómo funciona el modelo de seguridad en Android. La Sección 3.2 habla acerca de cómo se lleva a cabo el restablecimiento de fábrica, mientras que en la Sección 3.3 se muestra la manera en que se realiza el borrado de la información en la memoria flash. En la Sección 3.4 se presentan los diferentes formas en que se puede bloquear la pantalla del dispositivo. Asimismo, en la Sección 3.5 se presenta el cifrado de la memoria flash con el Cifrado de Disco Completo (CDC); en la Sección 3.6 aborda la generación de la llave de cifrado en el CDC y finalmente, la Seccion 3.7 se refiere a cómo se generan y almacenan las llaves de cifrado de los fabricantes Sony y Samsung.

3.1. Modelo de seguridad en Android

La plataforma Android tiene mecanismos de seguridad que ofrecen a los usuarios y desarrolladores cierto grado de protección sin tener que sacrificar sus funcionalidades. De acuerdo con la documentación de la AOSP [31], esta plataforma ofrece un entorno donde se protegen los servicios de confidencialidad, integridad y disponibilidad del sistema operativo, de los usuarios y de las

aplicaciones.

Al estar basado en Linux, Android ha adoptado muchos mecanismos del kernel creado por Linus Torvalds, incluyendo la gestión de la seguridad. Linux se rige bajo el contexto de usuarios análogo a un usuario real. Cada usuario está dentro de un grupo y ambos poseen un ID de usuario (UID) y un ID de grupo (GID), respectivamente. También, cada usuario tiene sus propios procesos junto con su propio espacio de memoria. Con esto, un proceso de un usuario no puede ser accedido por otro usuario.

Cabe destacar que la arquitectura de Android está diseñada para que cada capa asuma que la capa anterior está protegida de manera apropiada; así, por ejemplo la capa de los frameworks de las aplicaciones considera que la capa de las bibliotecas nativas y del entorno de ejecución están aseguradas (ver figura 2.3, en la página 16).

3.1.1. Usuarios y modelo de permisos

Técnicamente, en Android también se aplican los conceptos de usuario y grupo de Linux, con la diferencia de que cada usuario en lugar de ser una persona es una aplicación. Por consiguiente, cada aplicación posée un identificador diferente ID. No obstante, el ID de una aplicación en un dispositivo no necesariamente tiene que ser el mismo para otros dispositivos [32].

Asimismo, a los datos o archivos generados por una aplicación le son otorgados el ID de usuario de la misma, con esto las otras aplicaciones no podrán hacer uso de estos datos. Si se necesita que los archivos sean accedidos o escritos por otra aplicación, se hace uso de las banderas `MODE_WORLD_READABLE` y `MODE_WORLD_WRITABLE` que permiten a otra entidad leer y escribir, respectivamente, los datos de esa aplicación.

Así como en los demás sistemas basados en UNIX, en Android existe un súperusuario o *root*, usuario que posee todos los permisos sobre todos los archivos y programas del sistema operativo, sin embargo, su activación es diferente en comparación con las distribuciones Linux de escritorio, ya que para ganar privilegios de superusuario, es necesario realizar un proceso conocido como *rooting*. Los archivos y directorios que posee el *root* están en la partición `/system`, la cuál, por seguridad, es de sólo lectura. También, el *root* es el único usuario que puede acceder y escribir en algunas particiones como `/data`.

Permisos de las aplicaciones

La política de seguridad en Android establece que ninguna aplicación puede ejecutar acciones que perjudiquen al sistema operativo o al usuario, es decir, cada aplicación solamente tiene acceso a los recursos que va a necesitar, aplicándose así el principio del menor privilegio (*principle of least privilege*). Si las aplicaciones necesitan compartir o utilizar datos de otras aplicaciones o hacer uso de los recursos del sistema, deberán hacerlo de forma explícita [32] y esto se logra a través de los permisos.

El modelo de permisos consiste en determinar lo que una aplicación puede realizar en el sistema a través de un permiso. Por ejemplo, una aplicación que requiera conexión a internet necesitará el permiso de INTERNET. El permiso es examinado por Android en el momento en que la aplicación hace un llamado a la API de un recurso específico [13]. La Tabla 3.1 muestra algunos de los permisos que existen en Android.

NOMBRE DEL PERMISO	**SIGNIFICADO**
ACCESS_COARSE_LOCATION	Permite a la aplicación tener una localización aproximada
ACCESS_FINE_LOCATION	Permite a la aplicación acceder a una localización precisa
ACCESS_NETWORK_STATE	Permite a la aplicación tener acceso sobre las redes
ACCESS_WIFI_STATE	Permite a la aplicación acceder a la información sobre las redes WiFi
BATTERY_STATS	Permite a una aplicación obtener las estadísticas de la batería
BLUETOOTH_ADMIN	Permite a las aplicaciones descubrir y emparejar dispositivos con bluetooth
CALL_PHONE	Permite que la aplicación inicie una llamada telefónica sin necesidad de que el usuario confirme la llamada
CAPTURE_AUDIO_OUTPUT	Permite a una aplicación capturar el audio de salida
CAPTURE_VIDEO_OUTPUT	Permite a una aplicación capturar la salida de video
CAMERA	Permite tener acceso a la cámara
INTERNET	Permite a las aplicaciones tener acceso a sockets de red abiertos
NFC	Permite a las aplicaciones realizar operaciones de Entrada/Salida sobre NFC
READ_CALENDAR	Permite a una aplicación leer el calendario de datos de un usuario
READ_CALL_LOG	Permite a una aplicación leer los registros de llamadas del usuario
READ_CONTACTS	Permite a la aplicación leer los contactos del usuario
READ_EXTERNAL_STORAGE	Permite que la aplicación pueda leer desde un medio de almacenamiento externo
READ_SMS	Permite a una aplicación leer mensaje SMS
REBOOT	Requerido para reiniciar el dispositivo
RECORD_AUDIO	Permite a una aplicación grabar audio
WRITE_CONTACTS	Permite a una aplicación escribir en los contactos del usuario

Tabla 3.1: Algunos permisos en Android.

Los permisos de una aplicación se ubican en el archivo manifiesto (ver figura 3.3) y se pueden clasificar en cuatro diferentes categorías o niveles de protección, dependiendo de qué operaciones en el sistema necesite realizar la aplicación:

- **Normal.** Este tipo de permisos son automáticamente concedidos a la aplicación sin tener la obligación de solicitárselos al usuario. Esto da lugar a que si el archivo manifiesto de una aplicación tiene declarados permisos normales, Android no le pedirá al usuario conceder tales permisos, porque no representan riesgos que puedan poner en peligro a la seguridad del usuario en caso de mal funcionamiento.

  Ejemplos de estos permisos son `ACCESS_NETWORK_STATE`, que permite a las aplicaciones obtener información sobre las redes y `BLUETOOTH_ADMIN`, que brinda a las aplicaciones, el acceso a descubrir y vincular dispositivos por el servicio de bluetooth.

- **Peligroso.** Con esta clase de permisos, la aplicación es capaz de acceder en mayor o menor grado a los datos del usuario o de tomar cierto control sobre el sistema. Hasta la versión 5.0 y la API 22, para que una aplicación pueda acceder a esta clase de permisos, primero debe mostrárselos al usuario a través de un cuadro de diálogo que debe contener el nombre de los permisos que necesite y una breve explicación de los mismos. Después, será el usuario quien decida si deberá otorgarle o no los permisos a la aplicación (ver Figura 3.1). Ejemplos de estos permisos son `CAMERA`, que permite el acceso a la cámara así como sus características y `ACCESS_FINE_LOCATION`, que accede a la ubicación del dispositivo.

  Esta forma de concesión de permisos tiene la desventaja de que antes de instalarse la aplicación se deben de aceptar todos los permisos o ninguno, es decir, que si el usuario no desea conceder algún permiso, en consecuencia debe de rechazar los otros permisos y la aplicación no se instalará. Además, una vez que se hayan concedido permisos a la aplicación, no se podrán revocar después.

  Sin embargo, a partir de la versión 6.0 y la API 23, cada permiso peligroso es solicitado al usuario durante la ejecución de la aplicación, en el momento de la acción que necesite el uso de un permiso, como el de Figura 3.2, donde la aplicación requiere hacer uso de la cámara y en ese instante solicita el permiso. También, incluso después de haber concedido un permiso, es posible denegarlo y volver a concederlo desde la aplicación de configuración.

- **Firma.** Esta clase de permisos solo son concedidos si la aplicación que solicita el permiso está firmada con el mismo certificado que la aplicación que declaró el permiso. Si ambos certificados son compatibles, Android concederá el permiso a la aplicación solicitante sin notificárselo al usuario.

- **Firma o Sistema (*SignatureOrSystem).*** Este nivel de protección solo se otorgan a aplica-

ciones que se encuentren en la imágen del sistema. Por lo tanto, las aplicaciones del sistema son las que poseen este tipo de permisos. Se emplean en situaciones especiales como por ejemplo las aplicaciones que los fabricantes colocan en la imágen del sistema y necesitan compartir características específicas [33].

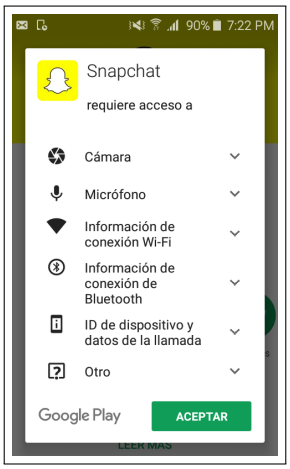


Figura 3.1: Cuadro de diálogo con permisos peligrosos en Android 5.0

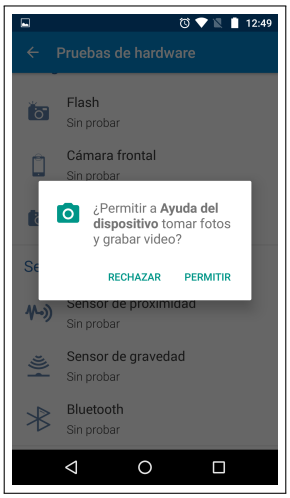


Figura 3.2: Cuadro de diálogo que muestra un permiso peligroso durante la ejecución de la aplicación en Android 6.0

Con este modelo las aplicaciones tienen restricciones hacia los recursos del sistema lo que brinda protección. Sin embargo se han asignado permisos inapropiados a aplicaciones o a sistemas de archivos y esto ha sido fuente de numerosas vulnerabilidades en Android [22].

```
 <!--permiso de usar INTERNET-->
<uses-permission android:name="android.permission.INTERNET" />
<!--permiso para descargas-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!--permiso para acceder al estado de la red-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figura 3.3: Permisos en el archivo manifiesto

3.1.2. *Sandbox*

Un proceso no puede acceder a la memoria de otro proceso, este concepto se materializa al nivel de la capa de las aplicaciones donde los procesos de un usuario se llevan a cabo en su propio espacio de direcciones de memoria [4] y no pueden ser accedidos por otro usuario, esto significa que los datos de una aplicación no pueden ser accedidos por otra aplicación, a este mecanismo se le llama *sandbox* o caja de arena.

De acuerdo con Lázaro [4], si existen procesos de diferentes aplicaciones que tengan la necesidad de compartir información, lo podrán hacer a través del método `sharedUserId`, y entonces Android les asignara a esas aplicaciones el mismo ID, como lo muestra la figura 3.4, donde el ID 10026 lo utilizan 4 aplicaciones. Para que el *Sandbox* pueda funcionar, es necesario firmar la aplicación.

```
com.sonyericsson.localcontacts 10026 0 /data/data/com.sonyericsson.localcontacts default
com.android.providers.contacts 10026 0 /data/data/com.android.providers.contacts default
com.android.providers.applications 10026 0 /data/data/com.android.providers.applications default
com.android.providers.userdictionary 10026 0 /data/data/com.android.providers.userdictionary default
```

Figura 3.4: Paquetes que poseen el mismo ID

3.1.3. Firmado de las aplicaciones

Una forma de identificar quién es el autor o autores de una aplicación en Android, es a través del firmado de código. Todas las aplicaciones deben de estar firmadas con la clave privada del autor de la aplicación ya sean aplicaciones del usuario, de los fabricantes o del sistema.

Con la firma de aplicaciones se asegura que las actualizaciones en las aplicaciones provienen del mismo autor que las desarrolló [22] (a esto se le llama política del mismo origen), proporcionando el servicio de autenticidad. Así, se puede comprobar si el código de la aplicación ha sido modificado por un tercero.

Una vez que la aplicación es firmada, se genera un certificado que puede ser autofirmado por el autor y junto a ese certificado va ligada la firma. Cuando la aplicación se instala en el dispositivo, el gestor de paquetes de Android se encarga de verificar la firma del apk.

Cuando dos o más aplicaciones son firmadas por el mismo autor, el gestor de paquetes de Android se encarga de verificar las llaves públicas de los certificados de esas aplicaciones, y si esas llaves resultan ser iguales, esto significa que la aplicación que fue instalada más recientemente, puede solicitar en el archivo manifiesto la compartición del mismo ID con la o las aplicaciones que tengan la misma firma, dando lugar a que pueden compartir datos entre ellas mismas.

Por lo general, las aplicaciones son firmadas utilizando la herramienta `signapk` que se encuentra ubicado en el directorio `/build` del código de la AOSP y los certificados pueden ser generados en el entorno de desarrollo que ocupa el programador como por ejemplo Eclipse o Android Studio.

3.1.4. CA instaladas en el dispositivo

Android tiene una lista de certificados de autoridades certificadoras (CA) raíces almacenada dentro del dispositivo en el directorio `/system/etc/security/cacerts`, muchos de estos certificados están almacenadas en hardware, aunque también hay casos donde el almacenamiento es en software, dependiendo del fabricante.

En algunos casos, estos certificados están codificados en PEM (*Privacy-Enhanced-Mail*, Correos con Privacidad Mejorada), una forma de codificación en Base64 para almacenar certificados, llaves, entre otros datos, y que sirve para proporcionar confidencialidad, autenticación e integridad [34]. Pero también hay dispositivos donde los certificados codificados en PEM, están precedidos o sucedidos por texto ASCII plano [25]. Estos certificados se pueden visualizar desde la aplicación de configuración (ver figura 3.5) y Android los clasifica como credenciales del sistema.

Además de los certificados instalados en el sistema, el usuario puede instalar certificados a través de un asistente de instalación, y cuando se instala, Android lo clasifica como CA del usuario. Si los certificados CA están habilitados, se almacenan en el directorio `/data/misc/keychain/cacerts-added`, pero si se desea marcar alguno como no de confianza, habría que apagar el control del interruptor (*switch*) del certificado correspondiente como los que aparecen en la figura 3.5 y cuando esto sucede, una copia del certificado se guarda en el directorio `/data/misc/keychain/cacerts-removed`, cuando se reactiva el certificado, se borra el archivo que había sido copiado a dicho directorio.

Figura 3.5: Lista de algunos certificados CA raíces que están instalados en el sistema

3.1.5. Validación de certificados

Android utiliza dos métodos para validar los certificados instalados: la lista negra (*blacklisting certificate*) y el *certificate pinning*.

Certificados en la lista negra (*blacklisting certificate*)

La revocación de certificados en Android no es *online*, lo que implica que no se pueden borrar certificados CA comprometidos automáticamente, a menos que se ejecute una actualización del sistema operativo [22]. Para dar solución a este conflicto, Android tiene a su disposición una lista negra (*blacklisting*), mecanismo que bloquea y revoca certificados que posiblemente hayan sido comprometidos.

Para gestionar la lista negra, Android almacena en el directorio `/data/misc/keychain/pubkey_blacklist.txt` los picadillos de las llaves públicas de los certificados revocados y en `/data/misc/keychain/serial_blacklist.txt` guarda los números de serie de los mismos. La modificación de estos archivos, requieren de permisos que solamente poseen las aplicaciones del sistema.

Certificate pinning

Este concepto es una medida de seguridad adicional al protocolo SSL para evitar la validación de certificados apócrifos o que hayan sido comprometidos. Consiste en asociar el certificado o la llave pública del anfitrión con el anfitrión, que sería la aplicación o CA, a esta asociación se le llama *pin* y los *pinnes* son el conjunto de esas asociaciones. Los *pinnes* se almacenan en una lista almacenada en un archivo llamado *pins*, que se ubica en el directorio `/data/misc/keychain`. Con los *pinnes* se

tienen registradas las CA que pueden ser aceptadas.

Cuando Android valida un certificado, se consulta en la lista de *pins* para ver si el *pin* de ese certificado está en la lista y si no lo está, ese certificado no se instalará. En Android existen numerosos *pinnes* preinstalados, como los *pinnes* por defecto de los certificados de Google, y los fabricantes también tienen la capacidad de adicionar *pinnes* [25].

3.2. Restablecimiento de fábrica

Esta función de Android se encarga de realizar un borrado de los datos y aplicaciones del usuario, sin eliminar las aplicaciones del sistema y de los fabricantes, dejando al dispositivo con la configuración que tenía de fábrica antes de ser adquirido por el usuario.[1] A este procedimiento también se le llama borrado seguro y existen dos maneras de realizarlo, la primera es desde el menú configuración y la segunda es desde el recovery. El restablecimiento de fábrica borra las particiones `/data`, `/sdcard` y `/cache`.

3.2.1. Restablecimiento desde el recovery

El recovery es una consola de recuperación que se encuentra almacenada en la partición `recovery` que tiene como función realizar tareas como actualizar el sistema operativo, montar la particion `system`, ver las bitácoras de la partición `recovery` y realizar restablecimiento de fábrica (ver figura 3.6). Esta consola se activa con una combinación de botones al momento de encender el dispositivo y esta combinación varía dependiendo del fabricante.

Figura 3.6: Menú recovery

[1] En el contexto de que compró o adquirió el dispositivo nuevo y no usado

Para realizar el borrado seguro, primero se debe escoger la opción `'wipe data/factory reset'` y una vez que es seleccionada y activada, se ejecuta el código /bootable/recovery/recovery.cpp, de la AOSP. Luego, el sistema principal se encarga de escribir `-wipe-data` al archivo `/cache/recovery/command`. Después, el sistema se inicia en recovery, la función `get_args()` ejecuta el *bootloader control block* (BCB), un mecanismo utilizado para que el recovery se comunique con el *bootloader* con los argumentos `boot-recovery` y `-wipe-data`, con esto el sistema se reinicia para comenzar el borrado. Posteriormente, las particiones `data` y `cache` son borradas con la función `erase_volume()` y la partición `/sdcard` se elimina con la función `WipeData()`. Luego, la función `finish_recovery()` se ocupa de borrar el BCB. Finalmente, el sistema principal es reiniciado.

La partición `/sdcard` es formateada con la función `WipeData()`. De acuerdo a la documentación de la AOSP, la codificación de esta función está a cargo de cada fabricante.

3.2.2. Restablecimiento desde el menú configuración

Este método de borrado seguro es activado desde la aplicación configuración o ajustes (*Settings*). Para llevar a cabo este método de borrado, primero, cuando el usuario selecciona el borrado seguro, se llama al Intent `android.intent.action.MASTER_CLEAR` de la clase `MasterClear.java`, que se encarga de llamar al intent `android.intent.action.MasterClearReceiver` perteneciente a la clase `MasterClearConfirm.java`. Después de eso, el intento `android.intent.action.MASTER_CLEAR_NOTIFICATION` de la clase `RecoverySystem.java` se encarga de reiniciar al equipo y llama al recovery enviándole el parámetro `'-wipe-data'`, el cual se ocupa de confirmar el restablecimiento de fábrica. Finalmente, comienza el procedimiento de borrado que realiza el recovery ya mencionado en la sección anterior, la figura 3.8 ilustra esta secuencia.

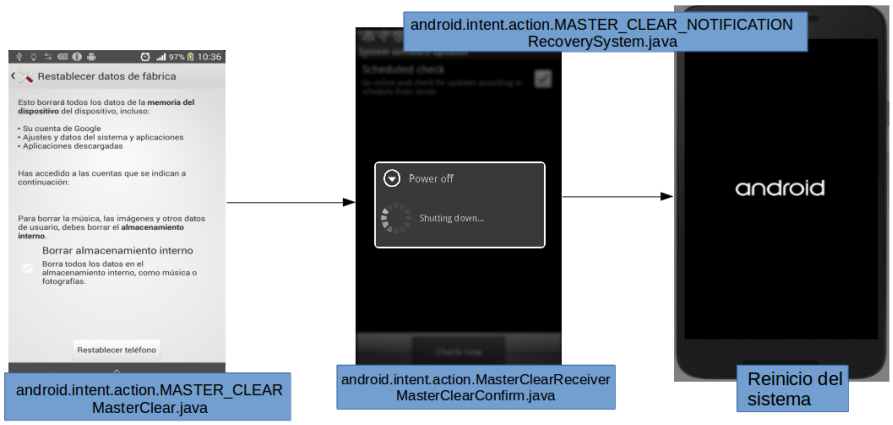


Figura 3.7: Secuencia de restablecimiento de fábrica desde el menú configuración

3.3. Borrado de la información

Cuando hay un restablecimiento de fábrica o el usuario elimina datos de Android, se realiza un proceso de borrado de información utilizando sanitización de datos.

3.3.1. Niveles de sanitización

La sanitización de datos es un método para borrar información de un medio de almacenamiento secundario. Las clases de sanitización que se ocupan por lo regular son la análoga, la digital y la lógica. La sanitización análoga es el nivel de borrado más eficaz, en vista de que la información eliminada no es posible de recuperar ni siquiera con los equipos de detección más sofisticados, la norma NIST 800-88 denomina a esta sanitización como "purgado"[35]. Por otra parte, en la sanitización digital los datos no pueden ser recuperados por medios digitales, incluyendo los comandos del controlador de la memoria que no están documentados. De manera similar, la sanitización lógica proporciona un borrado donde los datos no se pueden recuperar por medio de interfaces estándar de hardware [35].

3.3.2. Proceso de borrado en Android

Las memorias flash usualmente se organizan en páginas y bloques. De acuerdo con Simon y Anderson [8], dentro de cada bloque hay datos y datos/metadatos oob (*out of band*). Los oob se encargan de realizar trabajos como la administracion de bloques malos y establecer códigos de corrección de error.

El procesador solamente puede escribir o leer una página en la memoria flash (por lo regular de 512+16 bytes a 4096+128 bytes de datos+metadatos, respectivamente) y puede borrar bloques de 32 a 128 páginas de memoria. Esto es debido a que la tecnología flash solo puede soportar un número muy limitado de ciclos de borrado (aproximadamente 100 mil). Así que para optimizar su vida útil, se usan los algoritmos *wear-levelling*, los cuáles logran maximizar la vida util de los bloques, haciendo un uso eficiente de su espacio, además de que pueden detectar e indicar los sectores que son inutilizados para no perder información [36].

A partir de la versión 2.3, el almacenamiento flash de Android está soportado con la arquitectura eMMC (*Embedded MultiMedia Card*), el cuál, no da al sistema operativo acceso directo a la memoria flash, más bien le proporciona una vista lógica de la misma y está dividida en bloques lógicos. Cada bloque lógico es asociado a su correspondiente bloque físico a través de un controlador del eMMC [8].

Una vez que ocurre la eliminación de un bloque lógico N que estaba asociado a un bloque físico M+2, el controlador eMMC deshace dicha asociación, busca un bloque físico K sin contenido y reasocia al bloque N con el bloque K, dejando al bloque M+3 con la información que tenía almacenada proveniente del bloque N. Este proceso se ilustra en la figura 3.8.

En las primeras versiones de Android, cuando ocurría el restablecimiento de fábrica, el borrado de la información de la partición `/sdcard` se hacía con llamadas a la función de C `format()`, pero tenía el inconveniente de que solamente realizaba un borrado de unas cuántas docenas de MB. No obstante, para la partición `/data` se hacen llamadas al sistema `ioctl()` con el parámetro MEMERASE, un comando de eMMC que proporciona sanitización digital [8].

A partir de la versión 4.1, para borrar la partición `/data` en la llamada a `ioctl()`, se envía como parámetro `BLKSECDISCARD`, comando de eMMC que proporciona un borrado más seguro. Esto es porque se utilizan los comandos `SECURE TRIM` y `SECURE ERASE` que realizan un borrado de la memoria justo cuando se emiten estos comandos, al momento de ejecutarse tal operación, se necesita que el dispositivo espere hasta que esta operación esté completa antes de realizar el siguiente movimiento, con esto se realiza sanitización lógica. También, utiliza el comando `SANITIZE`, que se encarga de borrar los bloques físicos que han sido disasociados [37], proporcionando así sanitización digital [8].

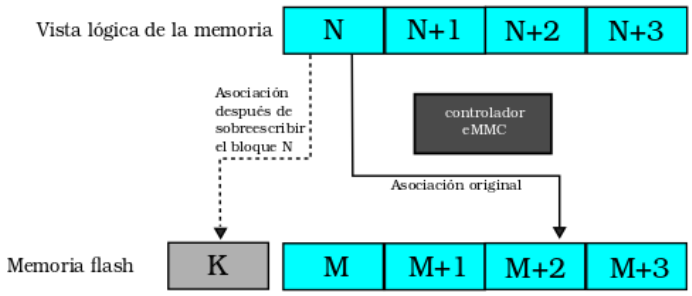


Figura 3.8: Asociación de los bloques físicos de la memoria flash con los bloques lógicos del controlador eMMC

3.4. Mecanismos de protección de la pantalla

Para evitar que cualquier persona tenga acceso físico a un celular Android, al igual que los sistemas operativos de escritorio, Android ofrece al usuario la capacidad de bloquear de la pantalla después de un tiempo determinado si es que no ha habido actividad por parte del usuario, brindando así el servicio de autenticación. Existen tres métodos de bloqueo: por contraseña, por PIN o por patrón, también algunos dispositivos tienen equipado el bloqueo por reconocimiento facial o de huellas dactilares.

3.4.1. Bloqueo por contraseña, PIN y patrón

El bloqueo por contraseña consiste en ingresar una contraseña de caracteres alfanuméricos de tamaño arbitrario, pero se necesita que sea de al menos cuatro caracteres y no más de 16 caracteres, la interfaz de este tipo de bloqueo se ilustra en la Figura 3.9. El bloqueo por PIN es similar al de la contraseña, con la diferencia de que solamente son una serie de caracteres numéricos y debe ser de entre 4 a 16 dígitos. De manera similar, el bloqueo por patrón consiste en establecer un patrón de movimiento [38] en una matriz de puntos de 3×3 y debe cubrir como mínimo 4 puntos de la matriz. Cuando el usuario escoge alguno de estos métodos, puede decidir si los caracteres, números o línea de movimiento entre los puntos de la matriz serán visibles o no al momento de desbloquear la pantalla.

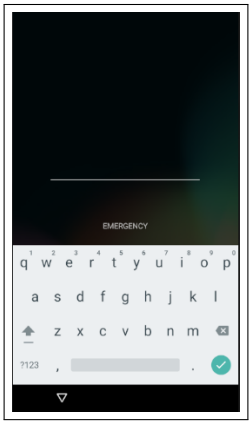


Figura 3.9: Pantalla de bloqueo con contraseña

3.4.2. Otros mecanismos de bloqueo: reconocimiento facial y huella dactilar

Algunos dipositivos, en especial los de gama media o alta ofrecen este tipo de autenticación. El bloqueo por reconocimiento facial consiste en que el usuario ubique su rostro frente a la parte

delantera del dipositivo, para desbloquear la pantalla. En el bloqueo con huella dactilar o digital, el usuario coloca la yema de su dedo (por lo regular el pulgar) en un sensor de huella digital, generalmente ubicado en la parte inferior frontal del dispositivo para desbloquear la pantalla, cabe destacar que aquí el usuario tiene que haber establecido con anterioridad un PIN o contraseña, este tipo de autenticación funciona solamente si el dispositivo tiene un sensor de huellas digitales. Cada fabricante tiene sus propias bibliotecas del manejo de las plantillas (*templates*) de las huellas digitales, mientras que Android se encarga de la interfaz entre el hardware (el sensor) y las bibliotecas.

3.5. Cifrado en el medio de almacenamiento secundario

Hasta la versión 2.3, en Android no existía un mecanismo que permitiera cifrar la memoria no volátil. Pero a partir de la versión 3.0, Android maneja el esquema de Cifrado de Disco Completo (CDC) o *Full Disk Encryption (FDE)*, que se encarga de cifrar los datos del usuario a nivel de particiones de disco[2]. Y a partir de la versión 7.0, soporta el Cifrado Basado en Archivos (CBA) o *File Based Encryption* (FBE) que cifra a nivel de archivo y cada uno de estos archivos pueden ser cifrados con llaves diferentes. Se puntualizará más al CDC, debido a que los dispositivos utilizados para experimentación en este trabajo de tesis tienen versiones inferiores a la 7.0. Cuando se activa el CDC o el CBA, el dispositivo permanece cifrado mientras esté apagado y descifrado cuando está iniciado el sistema operativo.

Cifrado de Disco Completo

El CDC se ocupa para cifrar la partición `/data` y lo hace antes de almacenar los datos en disco con una llave que había sido previamente cifrada llamada Llave de Cifrado de Datos (LCD) o *Data Encryption Key* (DEK), que a su vez también es cifrada con una Llave de Cifrado de la Llave (LCL) o *Key Encryption Key* (KEK). Para que este proceso se lleve a cabo, Android utiliza el subsistema `dm-crypt` de Linux [2]. El cifrado se hace con AES-128 modo CBC y ESSIV:SHA256, y es implementado a través del módulo `cryptfs` del demonio vold. Hasta la versión 5, para que se pueda activar el cifrado, es imprescindible tener establecido un PIN o contraseña, el mismo que se ocupa para bloquear la pantalla, pero a partir de la versión 6, esto ya es opcional.

Existen dispositivos con versiones 5 y posteriores que están cifrados por defecto, la documentación

[2]Se tomará como disco una palabra equivalente a flash y a partir de aquí se mencionarán ambos términos de manera indistinta

de la AOSP [39] menciona que en esta situación, durante el primer arranque (*first boot*) del dispositivo, Android crea una llave aleatoria de 128 bits y a esta partir de esta llave se obtiene su resumen (*hash*) con una contraseña por defecto y una salt, esta contraseña es "default_password", esto es porque así está establecido en el archivo `cryptfs.c` del código fuente de la AOSP, que está disponible en `https://android.googlesource.com/platform/system/vold/+/master/cryptfs.c`. El *hash* es firmado a través de la tecnología *Trusted Execution Environment* (TEE, de la que se hablará más adelante) y también es utilizado para cifrar la llave.

Asimismo, hay que tomar en cuenta que en el momento de cifrar, descifrar o borrar la partición `/data`, ésta no debe estar montada. Sin embargo, durante estos procesos se debe mostrar al usuario una interfaz que muestre el avance de cifrado o que solicite la contraseña para descifrar el dispositivo. Esta interfaz necesita de `/data` para funcionar, para lograr eso se monta un sistema de archivos temporal (tmpfs) de `/data`. Cuando `/data` es montada, Android detiene todos los procesos del tmpfs y los reinicia en la partición montada [39].

Existen cuatro situaciones en las que se puede cifrar/descifrar el dispositivo. La primera de ellas se aplica para el primer arranque a partir de Android 5.0 y es cuando se cifra el dispositivo con `forceencrypt`, una bandera que se activa cuando se detecta la partición que aún no está cifrada y cuando lo hace, comienza a cifrar la partición `/data` y a montar un sistema de archivos temporal también llamado `/data`. Después, durante el proceso de cifrado, se muestra una interfaz indicando el progreso de avance del cifrado, como el que se muestra en la Figura 3.10. Esta interfaz desaparece cuando la partición está completamente cifrada y finalmente continúa el proceso normal de arranque [39].

Otra situación de cifrado es cuando se ha establecido una contraseña y esto es para todos los dispositivos 4.4 o anteriores. En este caso, primero se revisa la contraseña, luego el dispositivo se apaga, vuelve a arrancar y durante el arranque, se crea un cripto footer (estructura donde se almacena la Llave de Cifrado de Datos cifrada). Posteriormente se crea un archivo *breadcrumb* (miga de pan en inglés, a manera de indicador), se reinicia el sistema, que detecta el archivo breadcrumb y comienza a cifrar la partición `/data`. Mientras, se monta el sistema de archivos temporal tmpfs y, de manera similar al método con `forceencrypt`, se muestra la interfaz de la Figura 3.10 indicando el avance de cifrado. Una vez que `/data` está cifrado, el *cripto footer* es actualizado.

También está la situación donde se arranca el dispositivo cifrado sin PIN/contraseña (versión 5 en adelante). Aquí, primero se detecta la partición `/data` que está cifrada, una vez detectada, se procede a descifrar dicha partición. Cuando `/data` ya está descifrada, se monta y empieza el arran-

que normal del sistema.

Por último, para los dispositivos cifrados con una contraseña o pin con versiones 5 en adelante, primero se verifica que `/data` está cifrado con contraseña, de manera similar a la primera situación, se monta un sistema de archivos temporal. Posteriormente, comienza a mostrarse una interfaz pidiendo la contraseña. Cuando se ingresa la contraseña correcta, `/data` es descifrada, se desmonta el sistema de archivos temporal y se monta `/data`. Finalmente, el sistema ya está listo para el arranque.

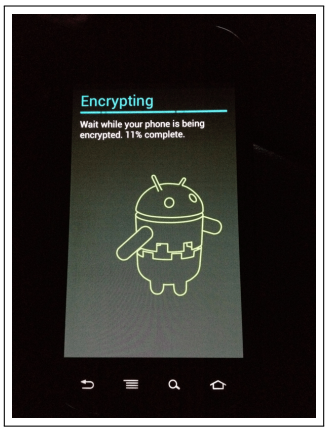


Figura 3.10: Interfaz que muestra el proceso de avance del cifrado de los datos.

3.6. Llave de cifrado

La Llave de Cifrado de Datos está guardada en el dispositivo y se encuentra en los metadatos, que están en el directorio `/efs/metadata`. Los parámetros de cifrado se almacenan en el cripto footer o `cripto_footer`, una estructura implementada en software que contiene datos relacionados con el cifrado como el tamaño de la llave, el algoritmo de cifrado, la sal y la Llave de Cifrado de Datos cifrada (ver Figura 3.11). Comúnmente el cripto footer está almacenado en los últimos 16 KB de la partición cifrada.

El cifrado se realiza con la LCD que había sido cifrada por la LCL. De acuerdo con [39] para cifrar y guardar esta LCD se tienen que realizar los siguientes pasos que siguen la secuencia ilustrada en la Figura 3.12, que está basada en [2].

1 Generar la LCD con una función de derivación de llaves como PBKDF2 o scrypt y con una salt de 16 bytes.

```
Android FDE crypto footer
-------------------------
Magic          : 0xD0B5B1C4
Major Version  : 1
Minor Version  : 3
Footer Size    : 2288 bytes
Flags          : 0x00000000
Key Size       : 128 bits
Failed Decrypts: 0
Crypto Type    : aes-cbc-essiv:sha256
Encrypted Key  : 0x825F3F10675C6F8B7A6F425599D9ECD7
Salt           : 0x0B9C7E8EA34417ED7425C3A3CFD2E928
KDF            : unknown (3)
N_factor       : 15     (N=32768)
r_factor       : 3      (r=8)
p_factor       : 1      (p=2)
-------------------------
...
```

Figura 3.11: Cripto Footer en Android CDC

2 A partir de la contraseña y de la salt, generar una llave intermedia IK1.

3 Rellenar IK1 con n ceros, siendo n el tamaño de la llave privada establecida en hardware por la TEE.

4 Generar una segunda llave intermedia, firmando el IK1 rellenado con la misma llave privada de la TEE del paso anterior usando RSA, a esta firma generada de 256 bytes se le llama IK2.

5 Generar una tercera firma intermedia IK3 de 32 bytes con IK2 y con la salt.

6 Utilizar los primeros 16 bytes de IK3 como la LCL y los últimos 16 bytes de IK3 como el vector de inicialización IV.

7 Cifrar la LCD con AES-CBC con LCL y IV.

8 Almacenar la LCD cifrada, la salt y la llave de la TEE en el cripto footer.

Trusted Execution Environment

Por lo regular Android protege las llaves de cifrado en hardware. Y el hardware que se ocupa de llevar a cabo la gestión y almacenamiento de las llaves está basado en el *Trusted Execution Environmet* (TEE) o Entorno de Ejecución de Confianza. El TEE es un tipo de tecnología ARM, un área segura del procesador principal [40] que a nivel hardware protege la LCL y se ocupa de la ejecución segura de las aplicaciones [41], es decir, que las mismas se ejecuten de manera aislada, que su almacenamiento y comunicación sean seguros. Al conjunto de aplicaciones que se llevan a cabo en el TEE se les llama *trustlets* y este entorno se ejecuta en paralelo con el sistema operativo.

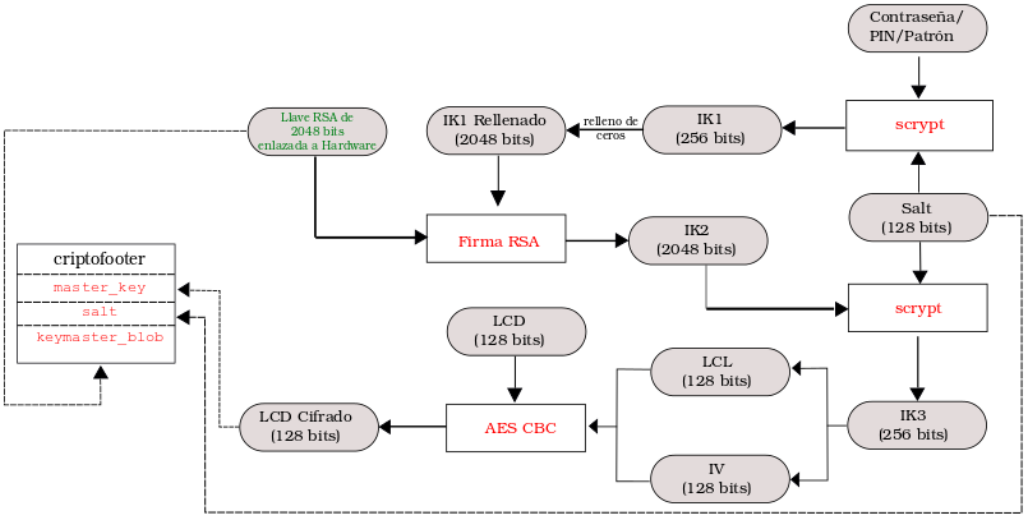


Figura 3.12: Cifrado de la Llave de Cifrado de Datos [2].

Una diferencia fundamental en el TEE con el sistema operativo es que las *trustlets* tienen acceso a todo el hardware del dispositivo, mientras que todo lo que sea procedente del SO debe pedir permiso al TEE para utilizar un recurso.

3.7. Llaves de cifrado en Android de algunos fabricantes

Cada fabricante tiene su propia manera de generar llaves y cifrar otras particiones como por ejemplo `/sdcard`. A continuación, se explicarán cómo es la gestión de llaves de los fabricantes Sony y Samsung en sus dispositivos móviles con Android.

3.7.1. Generación de las llaves en los dispositivos Sony Xperia con Android

De acuerdo a la documentación de Sony [42], en los dispositivos Sony Xperia, se utiliza el Cifrado de Disco Completo y tienen capacidad para realizar Funciones de Derivaciones de Llaves basadas en contraseñas acorde a una primitiva criptográfica específica como HMAC-SHA-1, SHA-256, SHA-384 y SHA-512 con 32768 iteraciones.

De manera similar a Android, en Sony se manejan dos tipos de llaves; las Llaves de Cifrado de Datos (DEK, *Data Encryption Key*) y la Llave de Cifrado de la Llave (KEK, *Key Encryption Key*). También, existe otra llave denominada Llave de Cifrado Raíz (REK, *Root Encryption Key*), que tam-

bién es una KEK. Las DEK se ocupan para proteger los datos, mientras que las KEK se emplean para proteger las DEK. La forma en cómo se administran estas llaves se muestra en la Figura 3.13. Algunas de estas llaves requieren de autenticación para acceder a ellas, típicamente una contraseña que puede tener más de 16 caracteres.

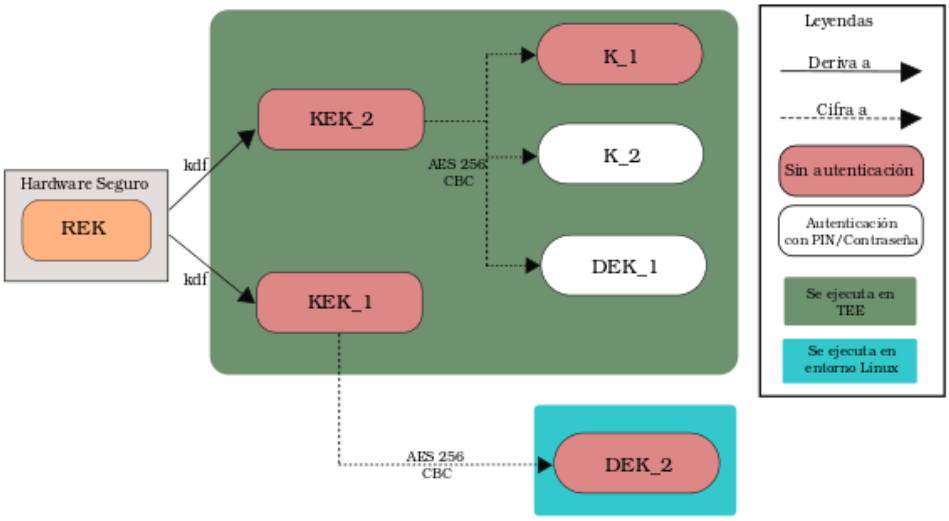


Figura 3.13: Administración de las llaves en los dispositivos de Sony con Cifrado de Disco Completo

La REK está implementada en hardware, es originada a partir de un generador de números aleatorios de fábrica, no necesita de autenticación y jamás es destruida. Como se puede observar en la Figura 3.13, a partir de la REK se derivan el resto de las llaves.

Dentro de las KEK están la KEK_1 y la KEK_2. La KEK_1 se encarga de cifrar a nivel de archivos la SDCard y cifrar la DEK_2 con AES-256 modo CBC, mientras que la KEK_2 cifra las llaves DEK_1, K_1 y K_2 también con AES-256 modo CBC (Figura 3.13); ambas llaves son derivadas de la REK, están implementadas en TEE y tampoco requieren de autenticación.

La DEK_2 es una llave de 256 bits que se ocupa de cifrar el almacenamiento de la SDCard externa, es creada solamente cuando el usuario habilita esta opción y está originada en software a partir del generador de números aleatorios de Linux. La DEK_1 está implementada en TEE de 256 bits con AES modo XTS y es originada a partir del generador de números aleatorios de la TEE. A diferencia de las llaves anteriores, la DEK_1 sí necesita de la contraseña del usuario durante el arranque del dispositivo, para poder descifrar el sistema de archivos.

La K_1 es en realidad un conjunto de llaves implementadas en TEE y que están almacenadas en hardware, que se utilizan para las aplicaciones. La K_2 es similar a la K_1, con la diferencia de que la aplicación etiquetada con la llave sólo es accesible si el usuario establece una contraseña.

Con excepción de la REK, las llaves pueden ser destruidas de acuerdo a la ubicación en la memoria. Si la llave está en la memoria volátil, se hace una sobreescritura de ceros y una verificación de lectura. Por otro lado, para las llaves que estén en la memoria EEPROM no volátil, se genera un patrón pseudoaleatorio y una verificación de lectura. Para las llaves que estén en la memoria flash, se realiza una sobreescritura directa de ceros y una verificación de lectura. Y finalmente para las llaves que no están en la memoria no volátil que no sea flash o EEPROM, la llave se sobreescribe tres veces o más con un patrón aleatorio que es cambiado antes de cada escritura.

3.7.2. Generación de las llaves en los dispositivos Samsung con Android

Samsung tiene un módulo de gestión de llaves criptográficas implementado en Software, de acuerdo a la documentación de [43]. Este módulo se encarga de crear las siguientes llaves y parámetros de seguridad:

- **Una Llave Semilla *(Seed Key).*** Es generada a partir de la utilería de Linux `/dev/random`.
- **Llave de Cifrado del Dispositivo, *Device Encryption Key* (DEK).** Es una llave de 32 bytes derivada de un generador de números aleatorios.
- **Llave Maestra *Master Key* (MK).** Es originada a partir de una función de derivación de llaves basada en contraseñas. Se utiliza como una llave para cifrar la Llave de Cifrado de Datos con AES de 256 bits y se deriva a partir de una contraseña de 4 a 32 caracteres.
- **Llave de Cifrado del Dispositivo Cifrado, *Encrypted device encryption Key* (EDK) y *Payload* EDK:** La EDK es la DEK cifrada con AES a partir de la Llave Maestra. El Payload EDK es la EDK con la salt y la HMAC de la EDK.

Algunas llaves necesitan de un generador de números aleatorios (RNG) [43], para eso, Samsung emplea un RNG que utiliza las semillas de `/dev/random` y tiene una entropía de 128 bits. Para destruir la DEK, Samsung realiza un llenado de ceros con la API de puesta a ceros (*Zeroization*), que se encarga de llenar la memoria de ceros. Y para borrar los otros parámetros de seguridad como la Salt y la HMAK, estas variables son sustituídas con ceros a través de la función `memset()`.

Capítulo

4

Problemas de seguridad en Android

"Los sistemas totalmente seguros no existen hoy en día y no existirán en el futuro."

Adi Shamir

En el capítulo anterior se dió una visión general de los mecanismos y servicios de seguridad que tiene Android. Sin embargo, existen vulnerabilidades y problemas con la implementación de éstos que han puesto de manifiesto la inseguridad en esta plataforma.

4.1. Problemas en el borrado de información

En la sección 3.3 en la página 41 se hizo mención de como es el proceso del borrado de información durante la eliminación de datos ya sea en el restablecimiento de fábrica o el borrado normal. Sin embargo, cuando un archivo es eliminado, lo que sucede realmente es que el sistema operativo solamente borra el nombre del archivo de una tabla y no borra su información contenida.

Por lo tanto, los datos aún permanecen en los bloques físicos de la memoria, debido al procedimiento que realiza el controlador eMMC cuando reasocia el bloque lógico donde el sistema operativo realizó la sobrescritura. Para solucionar esto, se usan los comandos estándar de eMMC que realizan el borrado físico de la información, `SECURE TRIM`, `SECURE ERASE` y `SANITIZE`.

De acuerdo con Bommisetty [44], la información eliminada de un dispositivo realmente se bo-

rra cuando llega el momento de almacenar nuevos datos y para eso sea necesario cubrir el espacio que está ocupado por los datos que se supone están eliminados, es decir, los bloques físicos que fueron desasociados. Para que esto se pueda hacer, antes de escribir los nuevos datos, primero se debe de borrar el contenido del bloque a ocupar. Esta situación sucede por ejemplo cuando el usuario está realizando actividades en el dispositivo que requiere el almacenamiento secundario de información o si ocurren nuevos eventos en el teléfono como una llamada telefónica entrante o la descarga de un archivo de internet que quedó pendiente.

Si bien la opción `BLKSECDISCARD` proporciona un borrado seguro, un problema que existe en el borrado de algunas particiones de la memoria está en que se ocupa la opción `BLKDISCARD`, que borra la partición `/sdcard` pero no garantiza seguridad ya que no proporciona sanitización. Esto es debido a que `BLKDISCARD` activa los comandos de eMMC `DISCARD` y `TRIM` [8], que no solicitan al eMMC que realice un purgado de los bloques, mas bien colocan los bloques a ser borrados en una cola, para que puedan ser borrados en un momento posterior. Por esta razón, `BLKDISCARD` resulta menos seguro [37].

Tallado de archivos

Para aprovechar esta situación ineficaz del borrado en la memoria, existe una técnica llamada tallado de archivos o *Data/File Carving*, que consiste en la busqueda secuencial en una imágen en bruto o crudo (*raw image*), de un dispositivo de almacenamiento secundario como un disco duro o una memoria flash, de caracteres que sean propios de un tipo de archivo en específico, de esta manera va "excarvando" a lo largo de la unidad.

La imágen en bruto no cuenta con un sistema de archivos, solamente son cadenas binarias y tiene toda la información generada del sistema operativo, incluyendo los datos eliminados. Por lo tanto, con el uso de esta técnica es posible recuperar archivos borrados.

Generalmente, cada tipo de archivo es identificado por un conjunto de bytes llamado cabecera y algunos tipos también poseen lo que es un pie. Entre la cabecera y el pie se encuentra el contenido del archivo, mientras que en este espacio algunos archivos también tienen unos bytes que representan otras características como el tamaño del archivo como es el caso de los JPEG, o el descriptor de archivo para los ZIP.

Asimismo, existen tipos de archivos que pueden tener más de un encabezado como es el caso de los GIF, aunque la diferencia muchas veces es mínima. Por ejemplo, la figura 4.1 presenta una

muestra de la configuración de la herramienta Scalpel, donde el renglón que dice "jpg", tiene la cabecera `\xff\xd8\xff\xe0\x00\x10` y el pie `\xff\xd9`, mientras los archivos gif tienen las cabeceras `\x47\x49\x46\x38\x37\x61`, `\x47\x49\x46\x38\x39\x61` y el pie `\x00\x3b`.

GIF and JPG files (very common)				
			cabecera	pie
gif	y	5000000	\x47\x49\x46\x38\x37\x61	\x00\x3b
gif	y	5000000	\x47\x49\x46\x38\x39\x61	\x00\x3b
jpg	y	200000000	\xff \xd8\xff \xe0\x00\x10	\xff \xd9

Figura 4.1: Cabeceras y pies de los archivos tipo JPEG y GIF en los archivos de configuración de Scalpel.

Los algoritmos de tallado de archivos comúnmente realizan una búsqueda secuencial de todos los bytes que integran el archivo. Sin embargo, hay algoritmos que solamente con encontrar unos cuántos bytes, ya identifican al archivo.

El inconveniente que tiene el tallado de archivos yace en que no todos los archivos tienen un pie, lo que hace más difícil la búsqueda y su ejecución regularmente tiene un costo computacional alto. Además, como la información eliminada del sistema de archivos tarde o temprano va a ser sobreescrita, muchas veces no se recuperan todos los archivos. Existen herramientas de código abierto que realizan tallado de archivos como Autopsy, Foremost, Scalpel y Bulk-Extractor.

4.2. Información del bluetooth

El bluetooth se ocupa para la comunicación entre dos dispositivos en distancias cortas, entre sus usos destacan la transferencia de archivos, auriculares con el perfil de manos libres, acceso telefónico a redes, así como control remoto de audio y video. Sin embargo, este servicio puede ser una gran superficie de ataques, porque se ha demostrado que a través de esta tecnología se puede robar y manipular información del usuario [45].

Dentro de la información que se puede obtener de un dispositivo bluetooth están los metadatos de los archivos que se envían o se reciben. Además, los nombres de archivos que se envían y reciben por bluetooth, así como la dirección MAC los dispositivos vinculados se almacenan en una base de datos llamada `btopp.db` que está guardada en la partición `/data`, en el directorio `/data/data/com.android.bluetooth/` donde también se ubican los datos relacionados a la confi-

guración del bluetooth.

De manera similar, las actividades que realiza este servicio, como los archivos que son enviados y recibidos, los tipos de archivos, así como los datos de los dispositivos que se vinculan, se registran en un archivo de bitácora llamada `btsnoop_hci.log`, ubicado en el directorio `/sdcard`. Para generar ésta bitácora, es necesario activar la opción "Registro de Bluetooth HCI" desde la sección de "Opciones de desarrollador/programador" en la aplicación de configuración, porque no está activada por defecto, esta opción es válida para la versión 4.4 y posteriores.

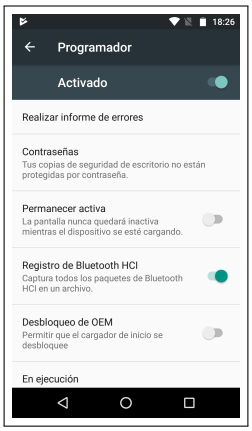


Figura 4.2: Registro de bluetooth hci en la aplicación Configuración

Para versiones anteriores a la 4.4, el procedimiento para activar esta bitácora puede resultar un tanto engorroso, en vista de que la aplicación configuración no soporta esta opción. Por lo tanto, para generar este archivo es necesario modificar el archivo `/etc/bluetooth/bt_stack.log`. En este archivo, si la línea `BtSnoopLogOutput` está asignada en `false`, se debe de cambiar por `true` y se debe reiniciar el servicio de bluetooth. De la misma manera, se hace lo mismo con las opciones `TraceConf` y `SSPDebug`. No obstante, en los dispositivos 4.4, el `btnsnoop_hci.log` se almacena en el directorio `/data/misc/bluedroid/btsnoop_hci.log`, que solo puede ser accedido por el usuario *root*.

Blueborne

El bluetooth no es un protocolo seguro, es por eso que se recomienda tenerlo apagado y encenderlo únicamente cuando se haga uso de este mecanismo. En Android, este servicio está apagado por defecto. Recientemente, esto se ha hecho más evidente porque se ha descubierto que un atacante puede interceptar este mecanismo sin mucha dificultad con Blueborne.

Blueborne es un vector de ataque descubierto por Armis [46] [47], que explota varias vulnerabilidades del protocolo bluetooth para ganar control del dispositivo, y afecta a los sistemas Android, Windows, iOS y Linux. Este vector se esparce por el aire y ataca al dispositivo vía bluetooth, el usuario no necesita realizar ninguna acción para establecer el ataque, mas que tener activado el servicio del bluetooth.

Si el atacante logra tener éxito, será capaz de controlar el dispositivo, podrá acceder a las redes, a los datos corporativos, esparcir malware como ransomware a los dispositivos con el bluetooth activado que estén cerca y crear *botnets* en dispositivos del internet de las cosas (IOT). Esto es posible porque que el bluetooth tiene altos privilegios en el sistema operativo. Además, un factor que pudo haber contribuído al éxito de este ataque podría deberse a que existe menos investigación en la seguridad del bluetooth, a comparación de otros mecanismos de comunicación como el WiFi [47].

4.3. Inseguridad en el servicio de WiFi

La mayoría de los dispositivos Android soportan los estándares de conexión inalámbrica 802.11g, 802.11n y 802.11ng [3]. Por defecto, el servicio de WiFi está encendido desde el momento que se configura por primera vez el dispositivo, a menos que el usuario lo desactive de manera manual (ver Figura 4.3), debido a que Android continuamente necesita de conexión a internet para actualizar el sistema o las aplicaciones, siempre está en busca de redes inalámbricas disponibles. Habría que destacar que Android prioriza el uso del WiFi sobre las redes celulares.

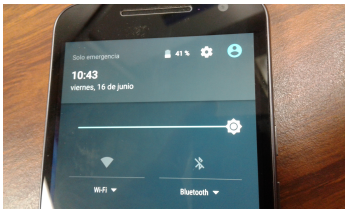


Figura 4.3: El servicio de WiFi está encendido por defecto

En Android existen dos maneras para conectarse a redes inalámbricas WiFi, con o sin autenticación. Cuando un dispositivo se conecta a una red sin autenticación, está expuesto a ser blanco de ataques de husmeo (*sniffing*) por atacantes que usen medios pasivos para interceptar el tráfico de red.

Caso contrario cuando está conectado a una red con autenticación, donde se ocupan esquemas de cifrado que impiden a un atacante que pueda hacer mal uso e interceptar la red sin conectarse,

a menos que tenga la clave. De entre los protocolos más utilizados están la Privacidad Equivalente al Cableado (WEP, *Wired Equivalent Privacy*), que ha sido roto y es considerado como no seguro, también están el Acceso Protegido WiFi (WPA, *WiFi Protected Access*) y su versión 2, WPA2 [3], que corrigen las debilidades de WEP ofreciendo un esquema más difícil de quebrantar.

De manera similar al bluetooth, algunas actividades que se realizan en el servicio de WiFi también se almacenan en archivo de bitácora. Esta bitácora se llama `wpa_supplicant.conf`, que está ubicada en el directorio `/data/misc/wifi`. En este archivo, se almacena la información relacionada a las redes inalámbricas en las que el dispositivo se ha conectado, almacenando datos como el nombre de la red, su protocolo de autenticación (si lo tiene), el número de prioridad y la contraseña de autenticación en claro para las redes que lo requieran, como se puede observar en la figura 4.4, donde las contraseñas de las redes "44e164" y "estmaestria" están en el campo de "psk". Para tener acceso a esta bitácora, es necesario contar con privilegios de *root*.

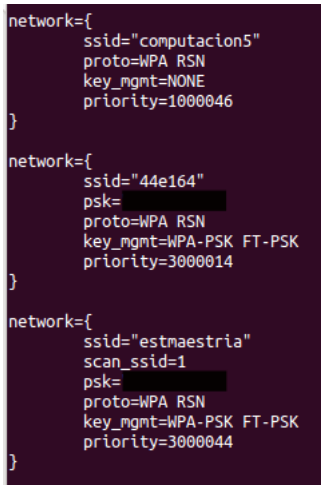


Figura 4.4: Archivo wpa_supplicant.conf

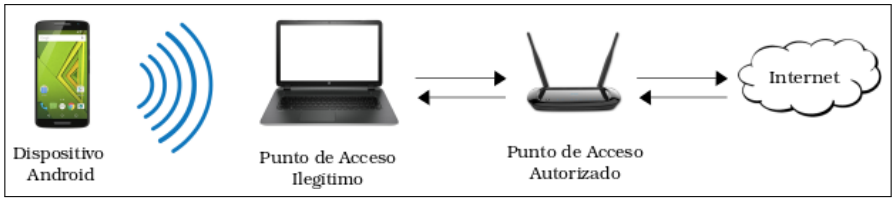
4.3.1. Ataques de husmeo con puntos de acceso

Choi et al. [13] menciona que uno de los casos donde existe fuga de información en Android es en el entorno de las redes inalámbricas, porque cuando los dispositivos Android están conectados a estas redes, sus aplicaciones, así como sus servicios del sistema, pueden transmitir datos confidenciales o sensibles como *cookies*, el ID del dispositivo o el IMEI.

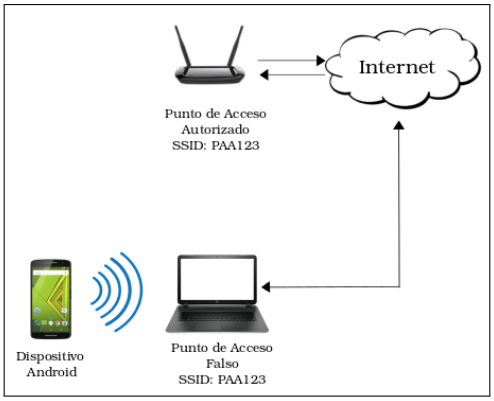
Tales datos pueden ser interceptados por un atacante, si éste está husmeando en la red, realizando Ataques de Intruso de en Medio (MITM, *Man In The Middle Attack*), principalmente en las redes

públicas o sin autenticación. Esto es posible porque las redes inalámbricas son más inseguras que las alámbricas ya que los datos viajan en el aire y esto trae como consecuencia que los atacantes puedan entrometerse con facilidad entre el usuario y el proveedor de servicios de Internet (ISP, *Internet service provider*) [5].

Existen dos formas de interceptar el tráfico de red, la primera es con un Punto de Acceso (PA) Malicioso o Ilegítimo independiente que el atacante instala en sitios públicos como aeropuertos, plazas públicas y centros comerciales. Como se puede ver en la Figura 4.5a, un modelo general de este ataque consiste en que el PA Ilegítimo se conecta al PA Autorizado de la LAN para poder brindar conexión a internet al terminal Android que se conecte al PA Ilegítimo. La segunda forma es con un Punto de Acceso Deshonesto o Falso (*Rogue Access Point*) que hace creer al usuario que es un punto de acceso genuino ya que poseé su mismo SSID (Figura 4.5b).



(a)



(b)

Figura 4.5: Modelo general de un ataque de husmeo: (a) Punto de Acceso Malicioso o Ilegítimo; (b) Punto de Acceso Deshonesto o Falso con el mismo SSID de la red legítima, en este caso PAA123

Tipos de análisis de tráfico de red

De acuerdo con Gupta [48], existen dos maneras de analizar el tráfico de red en el entorno Android.

- **Análisis pasivo.** El atacante únicamente se encarga de obtiener los paquetes de la red sin

interferir en la comunicación, guardando en archivos la información capturada.

- **Análisis activo.** El atacante interfiere en la comunicación y es capaz de capturar y manipular los datos que viajan a través de la red.

Comunicación HTTP y HTTPS

La mayor parte del tráfico de red en las aplicaciones y servicios en Android son del protocolo HTTP/HTTPS. Sin embargo, existe riesgo en utilizar HTTP porque toda la información que viaja a través de este protocolo se ve en claro. Cabe destacar que hay aplicaciones que aunque utilicen HTTPS, no lo implementan en toda la aplicación, porque hay eventos que hacen uso de HTTP.

Por ejemplo una aplicación que tiene el servicio de autenticación en HTTPS pero el manejo de la sesión lo sostiene en HTTP. Esto da lugar a que un atacante obtenga información del dispositivo si utiliza un Punto de Acceso Falso o Ilegítimo. De la misma manera, es posible que el atacante inyecte código malicioso Javascript en la transmisión HTTP.

4.3.2. Degradación de HTTPS a HTTP

Si bien mucha información que los dispositivos Android generan y reciben a través de las redes inalámbricas usa SSL/TLS, si no se implementa bien el manejo de este protocolo, es posible la degradación de HTTPS a HTTP, porque el protocolo SSL es inherentemente vulnerable a Ataques de Hombre en Medio [49]. Esto permitirá a un atacante ver toda la información que transita en la red, dentro de la que puede haber datos sensibles como credenciales de autenticación, e incluso, podrá insertar código malicioso en lenguajes como Javascript.

Este ataque consiste básicamente en evitar que el cliente, como un móvil Android, establezca comunicacón en el protocolo SSL/TLS con un servidor, por ejemplo una banca en línea o correo electrónico, y el atacante o intruso lo obliga a comunicarse vía HTTP reescribiendo el tráfico de red con apoyo de un Punto de Acceso Falso o Malicioso en las transmisiones de HTTP a HTTPS logrando así degradar esta transmisión a HTTP. La forma en cómo se realiza este ataque lo convierte en un análisis activo de tráfico de red.

De acuerdo con Frichot [50], existen dos maneras de ejecutar esta degradación; la primera es interceptando los datos que viajan en la red durante la petición del cliente al servidor para establecer conexión HTTPS. En este lapso, muchas aplicaciones web redirigen un estado de respuesta HTTP 302, lo que significa que habrá una redirección URL. Es en esta parte donde el atacante entra en acción y en lugar de entablar la conexión con la versión SSL/TLS del sitio o aplicación web, lo que

hace es redirigir a una versión HTTP del mismo. Para esto, reemplaza todos los enlaces que tengan escrito `https` por `http`.

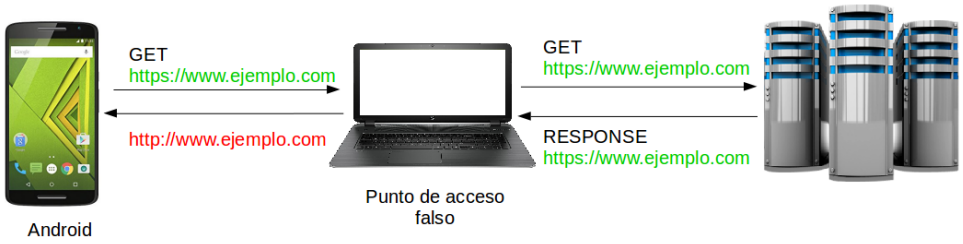


Figura 4.6: Esquema general de la degradación HTTPS a HTTP durante un Ataque de husmeo

El atacante podrá administrar la transición HTTP/HTTPS como si fuera el cliente y podrá husmear todo el tráfico que debería cifrarse, pero que no lo está. El resultado está en que el atacante se comunica con el servidor sobre HTTPS mientras que la víctima se comunica con HTTP tal y como lo representa el modelo de la Figura 4.6. Para llevar a cabo la degradación, se pueden ocupar herramientas como SSLStrip de Marlinspike [51].

Este ataque se coayuda del envenenamiento ARP (*ARP Poisoning/Spoofing*), que consiste en alterar el protocolo ARP de la red de forma que todo el tráfico de la LAN proveniente de los nodos de la red en lugar de dirigirse directamente al Punto de Acceso Autorizado, pase por el PA del atacante, envenenando así este protocolo. De esta manera, todo el tráfico de red pasará a través del PA Malicioso/Falso. Esto es posible porque ARP no tiene un método de validación. La herramienta ettercap es de gran utilidad para realizar el envenenamiento ARP.

La otra manera de realizar este ataque es reescribiendo la petición desde el navegador, es decir, modificar el código Javascript de la aplicación de tal forma que los enlaces a HTTPS sean modificadas a HTTP [50], esto es más fácil para los sitios que son vulnerables a ataques XSS.

4.4. Navegadores preinstalados en los dispositivos

Generalmente, cada dispositivo Android tiene una aplicación de navegador preinstalado como lo es Google Chrome. También, existe otro navegador llamado simplemente “Navegador” o “*Browser*”, que es de la AOSP (figura 4.7). Sin embargo, el Navegador de la AOSP ya no está presente en la mayoría de los dispositivos más recientes, porque muchos fabricantes lo han reemplazado por Chrome, sin embargo aún existen móviles y tabletas con ambas aplicaciones preinstaladas. Adicionalmente, algunos fabricantes tienen sus propios navegadores, como es el caso de Samsung.

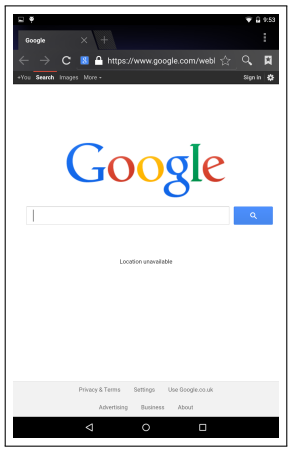


Figura 4.7: Navegador nativo de Android de la AOSP

Estos navegadores utilizan el motor de renderizado WebKit, biblioteca que se encuentra en la capa de las bibliotecas (ver figura 2.3). El módulo que maneja WebKit es libwebcore.so, que regularmente se ubica en el directorio `/system/lib`. A partir de la version 4.4 Android utiliza el motor de renderizado Chromium [52] ocupando los módulos libwebviewchromium.so y libchromium_net.so.

De acuerdo con Lanier et al. [3], WebKit y Chromium están incrustados en el *firmware*, en consecuencia, solamente pueden ser actualizados vía *firmware*.

4.4.1. Webview

Webview es la API de navegación que utilizan los desarolladores Android para desplegar contenido web en las aplicaciones, como por ejemplo la que se muestra en la figura 4.8, usando los motores del sistema, WebKit y Chromium. El contenido web se carga y se visualiza a través de una vista, con esto se puede incrustar una página web en una aplicación. Para que la aplicación pueda cargar contenido web desde internet, es necesario establecer el permiso `INTERNET` en el archivo manifiesto de esta forma: `<uses-permission android:name="android.permission.INTERNET" />` (ver figura 3.3). Con esta API, se puede personalizar el contenido y además las actualizaciones en la aplicación son más fáciles.

Webview tiene ciertas configuraciones por defecto que el desarrollador puede ir cambiando durante la codificación de su aplicación. Por ejemplo, Javascript estaba habilitado por defecto hasta la versión 4.3, pero desde la versión 4.4 ya no está activado, de manera similar, el *zoom* tampoco

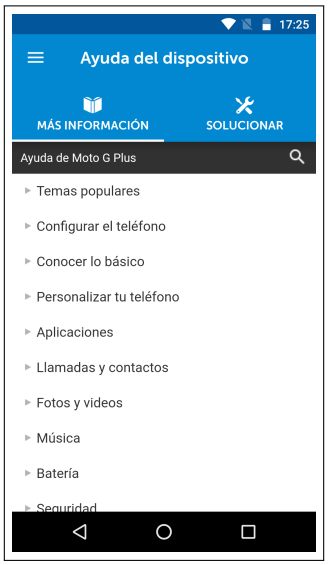


Figura 4.8: Aplicación "Ayuda" de Motorola que hace uso de webview para desplegar contenido web

lo está. Además, cuando el usuario activa un enlace a una url en particular, se abre el navegador preinstalado para visualizar el contenido de esa url. También, tienen acceso al sistema de archivos [53], a través de `file://` en la barra de dirección, pero solamente se pueden acceder a archivos de `/sdcard`. Otra característica de Webview es que permite la inserción de anuncios publicitarios *ads*, esto es con fines lucrativos para los programadores de las aplicaciones.

4.4.2. Problemas con WebKit y WebView

Webkit tiene muchas vulnerabilidades, y como Webview utiliza este motor, también está expuesto a tales susceptibilidades como lo es por ejemplo a inyección de código remoto. Además, Levin [25] menciona que WebKit contiene defectos como la malformación de HTML, CSS y Javascript. De la misma manera, el contenido web HTML de Webview está expuesto a los mismos ataques que los navegadores de escritorio como son Inyección SQL, *Cross-Site Scripting*, *Cross-Site Request Forgery*, Inyección HTML y *Phishing* [19].

Como el navegador tiene acceso a los archivos con `file://`, puede ser blanco de un ataque, porque tiene acceso a los archivos de la particion `/sdcard`, que es de lectura y escritura (figura 4.9), a esta vulnerabilidad se le conoce como *Scripting* de Zona-Cruzada basado en archivos (*File Based Cross-Zone Scripting* [54]).

Figura 4.9: Con file:// se puede ver los archivos de la partición `sdcard` desde el navegador

El agregado de interfaces de Javascript en WebView permite que este lenguaje pueada utilizar código Java [52]. Esto puede ser usado para explotar una vulnerabilidad conocida como Exceso de Autorización [54], que consiste en que si un atacante carga código Javascript en el sitio o aplicación, entonces puede llamar código Java, y en consecuencia pueda hacer uso de los recursos del dispositivo para realizar acciones tales como hacer llamadas, usar la cámara, envío de mensajes SMS, entre otros.

Asimismo, una superficie de ataque son las *ads* porque son susceptibles de recibir inyección de código a través de un ataque MITM, como por ejemplo el trabajo de Eom et al. [5], que muestra que es posible inyectar código malicioso en las *ads* durante la descarga de aplicaciones que tengan incrustadas *ads* desde Google Play.

De igual manera, webview permite que las advertencias de error producidas en la comunicación SSL/TLS sean ignoradas, si esto sucede, aumenta la probabilidad de ataques de Hombre en Medio.

4.5. *Root* habilitado en el dispositivo

En general, los dispositivos no tienen habilitado el usuario *root* y sin los privilegios que tiene el superusuario, no se puede acceder a particiones como `/data`. Solamente unos cuantos servicios

del sistema pueden tener tales privilegios [55]. Sin embargo, con el *rooting* se puede habilitar este superusuario, con esto el dispositivo estará *rooteado*. Con el *root* habilitado, el usuario será capaz de hacer casi cualquier cosa en su dispositivo, incluyendo el acceso y modificación a cualquier parte del sistema operativo.

El *rooting* consiste básicamente en insertar un archivo binario llamado `su` en el directorio `/system/xbin`, este archivo permite obtnener los privilegios de superusuario. De acuerdo con Cuadros [55], para realizar esto, es necesario que la partición `system` esté montada en modo lectura y escritura, ya que en un principio solamente tiene permiso de sólo lectura, además, cuando se instala este binario, la aplicación o proceso responsable de ello debe de tener asignado el bit `SUID`.

4.5.1. Riesgos de seguridad con el *root*.

Además de que muchos fabricantes estipulan en su documentación la anulación de la garantía, *rootear* el dispositivo trae consigo algunos riesgos de seguridad para el usuario, ya que las restricciones de permisos mencionadas en el capítulo 3 son traspasadas. Por ejemplo, con el *root*, el dispositivo es más vulnerable ante malware como virus o troyanos, porque podrán obtener información sensible, ya que con el *root*, se tiene acceso a cualquier parte del sistema. Inclusive, el malware puede instalarse durante el proceso de *rooting* [56].

También, cabe notar que si el *rooting* no se realiza de manera apropiada, es posible que el dispositivo llegue al estado de *bricked*, lo que significa que se volverá inutilizable.

4.5.2. Escala de privilegios, *rooting* con el uso de *exploits*

Aunque el dispositivo no este *rooteado*, un atacante puede utilizar distintas formas de activar el súperusuario sin el conocimiento del usuario, usando Escalación o Escala de Privilegios. Este termino se refiere a cuando una aplicación o un proceso, puede obtener privilegios de *root* temporal o permanentemente con el uso de un *exploit*.

De acuerdo con Levin [25], generalmente la forma en que los *exploits* logran el escalado de privilegios, es insertando una entrada en un proceso que está siendo ejecutado con permisos de súperusuario para corromper su memoria usando técnicas como *buffer overflow* y sobrescribiendo un puntero a alguna función, así el atacante puede manipular la ejecución de ese proceso.

Algunos de los *exploits* más famosos y con mayor impacto son *Framaroot, Exploid, RageAgainTheCage, Gingerbreak, Zimperlich, Towelroot, KillingInTheNameOf, memporoid, levitator* y reciente-

mente, *Drammer* y *Dirty Cow*.

Dirty Cow

Esta vulnerabilidad publicada en octubre de 2016 con el número *Common Vulnerabilities and Exposures* CVE-2016-5195, es más propio del kernel de Linux que uno exclusivo de Android, ya que todas las distribuiciones Linux con el kernel 2.0 a 4.26.28 son vulnerables. Al explotar esta vulnerabilidad un usuario puede realizar escalado de privilegios de forma local, escribiendo archivos de sólo lectura que son propiedad del *root*, obteniendo así privilegios de súperusuario, y una vez realizado el ataque, no hay rastro alguno del mismo en las bitácoras del sistema [57].

Para lograr esto, se aprovecha de un *bug* existente en el subsistema de gestión de memoria *copy-on-write* de Linux, que consiste en una condición de carrera (*race condition*), es decir, si las operaciones de lectura y escritura ocurren demasiado rápido, eventualmente el kernel sobreescribirá erróneamente el archivo que no debía ser escrito y así el atacante puede mover el espacio de memoria de sólo lectura con permisos de escritura.

4.6. Información en la memoria RAM

Tamma et al. [20] menciona que la información que se guarda en la memoria flash, en algún momento tiene que pasar por la memoria RAM, dentro de esta información se incluyen archivos abiertos, procesos en ejecución, eventos en la red, entre otros, por ejemplo la Figura 4.10 muestra una parte del volcado de memoria, que contiene los nombres de archivos de imágenes de capturas de pantalla que fueron creadas. Además, cualquier cosa que el usuario ingrese por teclado, también se almacena en la memoria principal, incluyendo credenciales de autenticación en texto plano, a las credenciales que están en memoria se les conoce como *data in motion*. Esto se debe en parte a que los procesos de las aplicaciones se ejecutan en segundo plano [11].

Toda esta información quedará almacenada hasta que sea sobreescrita o el dispositivo sea apagado o reiniciado. De manera similar, si se eliminan archivos como imágenes (siempre y cuando no sea a través de restablecimiento de fábrica), éstos quedan almacenados en la memoria principal, por lo que se pueden utilizar técnicas de tallado de archivos para obtener estos datos que se hayan eliminado.

Los trabajos realizados por Apostopoulos et al. [11] y Ntagonian et al. [10] demuestran que es posible recuperar credenciales de autenticación como usuario y contraseña de distintas aplicaciones en la memoria RAM utilizando herramientas de código abierto. Un inconveniente que tiene la

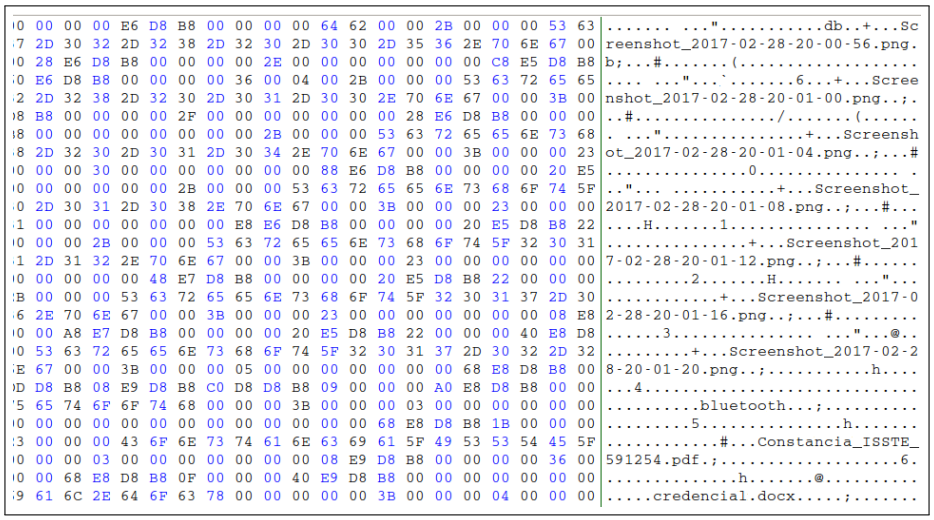


Figura 4.10: Información obtenida del volcado de memoria, nombres de archivos recién almacenados

obtención de la memoria es que se necesita acceso a *root* [20].

Existen distintas herramientas de software libres y gratuitas que permiten obtener el volcado de la memoria principal como son el Dalvik Debug Monitor System (DDMS), mem, y el Linux Memory Extractor (LiME). Sin embargo estas herramientas tienen sus inconvenientes, por ejemplo, DDMS no puede volcar la memoria completa [10]. LiME y mem pueden obtener la memoria completa, pero mem tiene que colocarse en la RAM, lo que implica que se tendrá que sobrescribir un pequeño cúmulo de la memoria, y a LiME le falta portabilidad, porque al ser un módulo de núcleo debe ser construido y cargado para cada kernel y su compilación varía de acuerdo al dispositivo, de igual forma, no todos los dispositivos pueden ser compatibles con LiME. También, es posible realizar técnicas basadas en hardware como el ataque de *cold boot* junto con el *framework* FROST [1] de Müller y Spreitzenbarth, su desventaja yace en que depende de la permanencia de los datos en la memoria.

4.7. Problemas con el modelo de permisos

Con el modelo de permisos, Android evita que las aplicaciones tengan acceso inmediato a los recursos del sistema y del dispositivo. Sin embargo, este modelo tiene fallos porque muchos permisos son de granularidad gruesa, es decir, que un sólo permiso puede tener acceso a muchos recursos

[1] https://www1.informatik.uni-erlangen.de/frost, consultado el 17-07-2017

como por ejemplo el permiso `INTERNET`, que permite a la aplicación enviar enviar peticiones HTTP/HTTPS a cualquier dominio.

Existen aplicaciones que hacen uso excesivo de permisos que no deben de usar y como hasta la versión 5, para instalar la aplicación el usuario debe aceptarlos todos, si los acepta, la aplicación tendrá todos esos permisos mientras esté instalada, aunque esto ya se mitigó en cierto grado a partir de la versión 6, porque el usuario ya puede decidir qué permisos puede tener la aplicación y en cualquier momento los puede desactivar, como los de la aplicación cámara que se muestran en la figura 4.11.

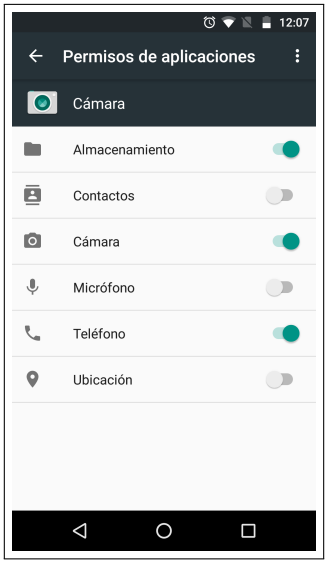


Figura 4.11: Desde la versión 6, los permisos pueden revocarse después de que la aplicación se haya instalado. Aquí, la aplicación cámara tiene revocados los permisos de acceder a los Contactos, al Micrófono y a la Ubicación

También, este modelo de permisos tiene el defecto de que una aplicación con pocos permisos o ninguno pueda hacer escala de privilegios debido a la falta de protección de aplicaciones que tienen componentes que pueden hacer uso de varios recursos, porque tales componentes pueden ser accedidos por cualquier otra aplicación.

4.8. Problemas en el Cifrado CDC

El Cifrado de Disco Completo protege los datos del usuario de la forma *Data on Rest*, es decir, cuando el dispositivo está apagado. Asimismo, el CDC está implementado completamente en soft-

ware hasta la versión 4.3, por lo tanto, su seguridad depende de la fortaleza de la contraseña [22]. Para que un atacante pueda descifrar el disco, necesita primero descifrar la LCD. Sin embargo, este mecanismo de cifrado presenta varios problemas que han permitido que se pueda obtener la LCD. Asimismo, cuando el usuario configura por primera vez el dispositivo, Android nunca le sugiere que active el cifrado, solamente le pregunta si desea establecer algún mecanismo de protección de la pantalla.

4.8.1. Fuerza bruta en las contraseñas

Para descifrar la LCD se necesita saber el PIN o contraseña, y para conseguirla, un atacante requeriría hacer ataque de fuerza bruta, que resultaría impráctico ya que tomaría años. No obstante, Elenkov [22] menciona que en versiones 4.3 y anteriores si el usuario estableció un PIN corto, realizar un ataque por fuerza bruta no toma mucho tiempo. Esto es por la función de derivación de llaves PBKDF, en la cuál es posible hacer un ataque de fuerza bruta con herramientas como bruteforce_stdcrypto y hashcat, más aún con una GPU porque esta función de derivación es paralelizable, para realizar este ataque, es necesario que el atacante tenga el cripto footer y la partición de la memoria flash cifrada.

A partir de la versión 4.4 se usa la función de derivación scrypt y usar la GPU ya no es viable porque esta función utiliza mucha memoria, algo que no suelen administrar bien las GPU. Sin embargo, hashcat puede obtener llaves de scrypt, de manera similar a PBKDF también se necesitan el cripto footer y generar todos los PIN o contraseñas posibles.

4.8.2. Problemas en TEE

Para la versión 5 y 6, se implementa la aceleración en hardware en el CDC y la LCL está protegida en la TEE. Sin embargo, este entorno tiene problemas de seguridad; porque Rosemberg [58] demostró que las implementaciónes de TEE en los circuitos de Qualcomm (QTEE) son vulnerables debido a que si el código del kernel manda una solicitud *Secure Monitor Call* (SMC) al QTEE para utilizar un servicio, QTEE puede provocar desbordamiento de enteros (*Integer Overflow*), así un atacante con permisos a nivel de kernel puede explotar esta vulnerabillidad, logrando ejecutar código arbitrario en el TEE y con esto podría quebrantar los mecanismos de protección del Sistema Operativo y comprometer los elementos que intervienen en la generación de las llaves de cifrado.

4.9. Problemas con los métodos de bloqueo de la pantalla

En el Capítulo 3 se habló acerca de los métodos de bloqueo de pantalla. Si bien ofrecen cierta seguridad al usuario en el caso de que por ejemplo un ladrón robe el dispositivo, no podrá ver el contenido si la pantalla está bloqueada y requiera de algún método de autenticación. Sin embargo, estos bloqueos pueden ser traspasados por un atacante, si tiene el dispositivo a la mano. Si el atacante intenta realizar fuerza bruta, algunos fabricantes establecen un límite de intentos para desbloquear la pantalla y en caso de exceder dicho límite, los datos son borrados y se realiza un restablecimiento de fábrica (aunque aún así se pueden recuperar datos, véase Sección 4.1), no obstante existen otros métodos que permiten obtener el patrón/PIN/contraseña.

Para llevar a cabo las técnicas mencionadas en las Secciones 4.9.1 y 4.9.2 es necesario tener *rooteado* el dispositivo y también tener habilitada la opción de depuración por USB o tener desbloqueado el *bootloader*, pero esto no es necesario para las técnicas mostradas en la sección 4.9.3.

4.9.1. Ataques al PIN/Contraseña

El PIN/contraseña no se guarda en claro en el dispositivo, mas bien es su *hash* al que previamente se le aplicó una salt el que se almacena en el archivo `password.key` ubicado en el directorio `/data/system/` (Figura 4.12). Para versiones 4.4 en adelante, la salt se guarda en la base de datos `locksettings.db` en la tabla `locksettings`, también en el mismo directorio y para las versión 4.3 y anteriores, se guarda en la base `settings.db` que está en `/data/data/com.android.providers.settings/databases/` en la tabla `secure`. Se puede obtener el PIN/Contraseña con el *hash* y el valor de la salt. Con ambos datos se pueden utilizar herramientas como hashcat, CCL Forensics o BruteForceAndroidPin para la recuperación del PIN o de la contraseña.

```
root@C5306:/data/system # cat password.key
1B529B8E1D2563EB55338F62B931B902FA3B0D0AED6380114E97E82F3A289583653C7001
```

Figura 4.12: Archivo password.key, que contiene el *hash* de la contraseña

4.9.2. Ataques al bloqueo por patrón

De manera similar al PIN/contraseña, se genera un resúmen del patrón que es almacenado en el archivo `gestures.key` ubicado también en el directorio `/data/system/`. Para traspasar el patrón existen dos opciones; eliminar `gestures.key` u obtener este archivo y tratar de conseguir el patrón que generó el *hash* obtenido, para eso pueden utilizarse herramientas como CCL Forensics o hashcat.

Asimismo, también es posible encontrar el patrón con el ataque *smudge*, que consiste básicamente en detectar las manchas de los dedos que deja el usuario en la pantalla, esto es más fácil si se utiliza la iluminación adecuada.

4.9.3. Otras técnicas de desbloqueo

A través de adb

Cuando se opta por utilizar *adb* para interactuar con el dispositivo, la computadora personal o estación de trabajo que hará uso de éste, debe de estar autorizada para hacerlo. Esto es una medida de seguridad para evitar que cualquiera que no sea el propietario del dispositivo haga mal uso del mismo. Sin embargo, existe un fallo en este mecanismo en las versiones 4.2.2 a 4.4.2, de acuerdo con la información de Tamma et al. [59], el cuadro de diálogo de autorización de *adb* aparece en el dispositivo antes de que la pantalla sea desbloqueada, por lo tanto, cualquiera que conecte el dispositivo a una estación de trabajo, como una PC, podrá acceder al dispositivo.

Quebrantar la pantalla de desbloqueo (versiones 5.0-5.1)

Esta técnica funciona sólo para bloqueo por contraseña y para algunos dispositivos que tengan las versiones 5.0-5.1. Se emplea en la parte de "Llamada de emergencia", en el marcador, donde se inserta una entrada aleatoria de teclas, típicamente diez asteriscos (*). Esa cadena de caracteres (las teclas presionadas) se copia y pega al final de la cadena, este proceso se hace varias veces, hasta que la cadena sea lo suficientemente larga para que cuando se intente seleccionarla, el texto ya no se resalte.

Esta larga cadena se copia y se pega varias veces en una interfaz que solicita al usuario la contraseña que aparece desde la pantalla de notificaciones, hasta que la pantalla se desbloquea. Para realizar esto último es necesario tener abierta la aplicación de Cámara.

Uso del modo seguro

Este método funciona en dispositivos que tengan bloqueos de pantalla de terceras fuentes, es decir, que no sean de la AOSP. Esta modalidad se activa cuando se presiona el botón de apagado y en el mensaje que aparece en la pantalla (Figura 4.13), se mantiene presionado la opción de "Apagar", es entonces cuando Android le pregunta al usuario si desea iniciar en el modo seguro y en caso de que se presione la opción "OK/Aceptar" el dispositivo se reiniciará desbloqueado porque todas las aplicaciones de terceras fuentes estarán deshabilitadas.

Figura 4.13: Mensaje que muestra la opción de reiniciar en modo seguro

Capítulo

5

Desarrollo y experimentación

"Vale más saber alguna cosa de todo que saberlo todo de una sola cosa"

Blaise Pascal

En este capítulo se explican las diferentes pruebas que se realizaron para comprobar la seguridad de algunos de los mecanismos que ofrece Android, cuáles son sus fallas y qué información es posible de recuperar.

La sección 5.1 aborda la metodología utilizada para obtener información eliminada de la memoria flash y qué es lo que se pudo rescatar. La sección 5.2 explica la información que es posible recuperar de la comunicación bluetooth. La sección 5.3 se refiere a la medición del consumo de recursos de las aplicaciones de algunos fabricantes. En la sección 5.4 se presentan las pruebas de seguridad del protocolo TLS/SSL que se realizaron a los navegadores preinstalados.

Y finalmente, la seccion 5.5 habla sobre el ataques de husmeo con punto de acceso ilegítimo a los navegadores preinstalados de dispositivos con diferentes versiones y a las aplicaciones preinstaladas de algunos fabricantes, así como el desarrollo de una aplicación que usa la API de navegación para comprobar que tan susceptible es ante este tipo de ataques.

Los dispositivos utlizados para las pruebas fueron un teléfono inteligente Sony Xperia SP C5306 con la versión 4.3 (Jelly Bean), una tableta Dell Venue 7 con la versión 4.4 (Kitkat), un teléfono

inteligente Samsung Galaxy Grand Prime con la 5.1 (Lollipop) y un Motorola Moto G cuarta generación con la versión 6.0 (Marhsmallow), a partir de aquí, a los cuatro dispositivos mencionados serán referidos como dispositivos.

5.1. Recuperación de información eliminada

En el capítulo 3 se vió la forma en cómo se elimina la información de la memoria flash y en el capítulo 4 los problemas que tienen los mecanismos de borrado y cómo es posible recuperar información eliminada con el uso del tallado de archivos. Así que para comprobar qué datos se pueden obtener, se hicieron pruebas con el restablecimiento de fábrica y con el borrado normal para verificar la información que se puede recuperar. Para realizar estas pruebas se ocuparon los cuatro dispositivos, cabe destacar que los que tienen la versión 4.3 (Sony) y la versión 4.4 (Dell) fueron *rooteados* y a los dispositivos con la versión 5.1 (Samsung) y 6.0 (Motorola) se les desbloqueó el cargador de arranque (*bootloader*).

5.1.1. Modelo de amenaza

Antes de realizar al borrado, se ingresó información al dispositivo como fotos, videos, contactos, llamadas telefónicas, envío de mensajes y archivos de ofimática, tal y como lo haría un usuario normal. Asimismo, se realizaron actividades en el navegador Chrome como navegación y búsqueda en algunos sitios web. Posteriormente, se llevó a cabo el proceso de eliminación, siguiendo dos escenarios:

- **Escenario 1.** Se realizó el borrado normal, es decir, los datos se eliminaron de forma manual, desde las aplicaciones. Por ejemplo, para eliminar fotos, se borraron desde la aplicación de fotos/album y se borraron los historiales de los navegadores. Inmediatamente, se obtuvo la imágen en bruto completa de la memoria flash y con esa imagen se aplicó tallado de archivos. También, se eliminó la cuenta de Google.
- **Escenario 2.** Se eliminó la información con el restablecimiento de fábrica desde el menú configuración. En seguida, de manera similar al Escenario 1, se obtuvo la imágen de la memoria para aplicar tallado de archivos.

En ambos escenarios, durante la extracción de la imágen de la memoria flash, se conectó al dispositivo a una computadora personal con Kali Linux 2016.1 con un cable usb Tipo C, y con el uso

de `adb` a través de los comandos `dd`, `if`, se pudo extraer la imágen de dicha unidad. Después, se aplicaron técnicas de tallado de archivos para recuperar la información eliminada.

Cabe destacar que un requisito indispensable para la extracción, es tener habilitada la opción de depuración por USB y de alguna forma tener habilitado el acceso al súperusuario. Esa es la razón por la que se *rootearon* dos de los dispositivos, y a los que tienen el cagador de arranque desbloqueado, se les cargó una imágen de recuperación llamada TWRP, que permite el uso de un *root* temporal. Las herramientas utilizadas para el tallado de archivos fueron Foremost, Scalpel y Bulk_Extractor.

Resultados obtenidos

Resultados obtenidos por el borrado normal

Cuando se aplicaron las técnicas de tallado de archivos, se pudo recuperar parte de la información que se había eliminado, principalmente imágenes, capturas de algunos videos, documentos PDF, nombres de contactos, números telefónicos e información relacionada al historial de navegación de los navegadores como nombres de sitios web.

Con Foremost y Scalpel se consiguieron imágenes, capturas de videos y recuperación de archivos de documentos. Mientras, con Bulk_Extractor se recuperaron unos cuántos números telefónicos, nombres de contactos e información relacionada al historial de navegación de los navegadores, también se obtuvieron imágenes, pero fueron muy pocas comparados con las que consiguieron Foremost y Scalpel. Cabe destacar que los nombres que las tres herramientas asignan a los datos recuperados no son los originales.

Muchas imágenes se recuperaron, las cuáles incluyen fotos tomadas con la cámara, imágenes descargadas de internet, capturas de pantalla, de videos y también algunas que forman parte de sitios visitados en los navegadores. Sin embargo, fueron pocas las imágenes que se encontraban en buen estado, en otras palabras, que se reconozca la imágen a simple vista y que esté completa o casi completa. Además de eso, hubo imágenes que estaban en un estado medianamente bueno o distinguible, lo que significa que se obtuvo nada más una parte de la imágen original. Esto es porque muchas de ellas se encontraban en mal estado, únicamente una muy escasa parte de cada imágen se podía distinguir o eran completamente indistinguibles.

El teléfono inteligente Sony (versión 4.3), contó con una considerable cantidad de recuperación porque cerca del 39 % de imágenes estaba en buen estado, pero el 25 % eran imágenes que eran más pequeñas que las originales y se obtuvieron 23 imágenes que eran parte del contenido de si-

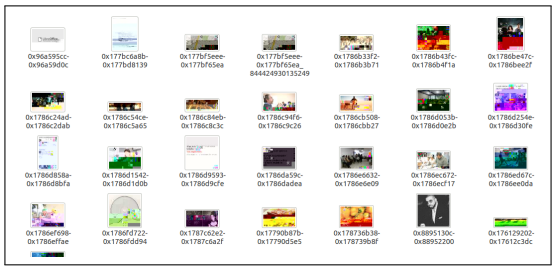


Figura 5.1: Algunas imágenes recuperadas después de realizar el borrado normal.

tios web, la figura 5.1 muestra algunas imágenes recuperadas de este dispositivo. En el dispositivo Dell, solo el 19 % de las imágenes se encontraba en buen estado, más nueve provenientes de sitios web estaban también en buen estado.

En el dispositivo con Lollipop (Samsung), 40.5 % de las imágenes estaban en buen estado, pero de este 40.5 %, el 32.4 % eran más pequeñas que su tamaño original, también se encontraron más de 50 imágenes obtenidas de sitios web. Finalmente, el dispositivo Motorola fue el que tuvo menos éxito, ya que solo el 14 % se encontraban en buen estado, y de ese conjunto, solo un 2 % tenía su tamaño original.

De los documentos recuperados, a la gran mayoría no se les podía ver su contenido. En el teléfono con Jelly-Bean se recuperaron más del 90 % de los PDF, y en la Dell se recuperaron 35.2 % de los documentos PDF, en la Samsung se obtuvo el 27 % de los PDF, y en el Motorola, un 50 %.
Un aspecto a destacar en la información de un archivo son sus metadatos. Si antes bien muchos archivos recuperados estaban en mal estado, en algunas ocasiones se lograron rescatar sus metadatos. Esto fue exitoso principalmente en los archivos PDF, porque en los metadatos que tenían, se encontraban el nombre original del archivo, el autor, las palabras clave y la fecha de creación. La figura 5.2 muestra un ejemplo de metadatos de un archivo PDF.

En el dispositivo Sony aproximadamente el 50 % de los documentos PDF recuperados tenían los metadatos intactos. En la tableta Dell, no se consiguieron los metadatos en ningún archivo. Pero en el teléfono Samsung esto se logró en un 40 % de los archivos PDF rescatados. Y finalmente, en el dispositivo Motorola, tampoco se logró obtener metadatos en ningún documento recuperado.
De la misma forma, se obtuvieron números telefónicos de contactos, así como sus nombres e información de la actividad en los navegadores, como los nombres del los sitios visitados. Esto sucedió más en el teléfono Sony, donde se obtuvieron la mayor parte de los números telefónicos, una página web y sitios web visitados en la aplicación del navegador preinstalado.

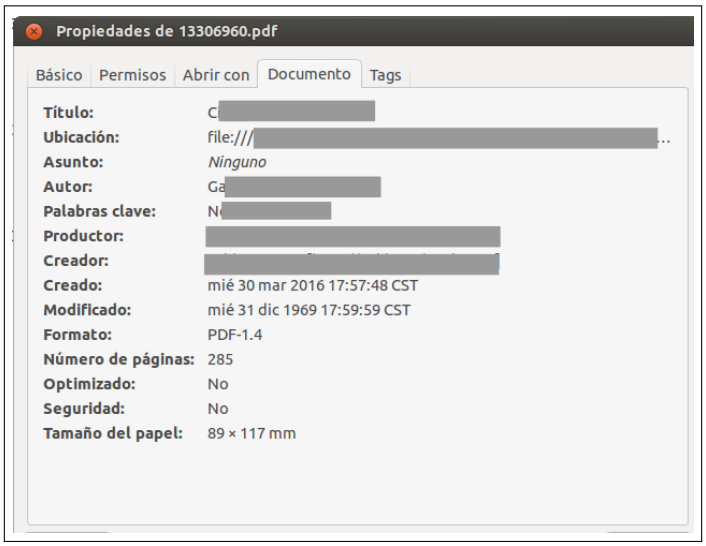


Figura 5.2: Metadatos de archivo PDF recuperado.

En la tableta Dell, se obtuvieron nombres de algunos contactos y todos los números telefónicos y una página web, así como todos los correos electrónicos empleados en las aplicaciones del dispositivo, y nombres de sitios web. En el telefono Samsung, se obtuvieron los correos electrónicos de los sitios visitados, también se recuperaron números telefónicos y nombres de sitios web. En el Motorola se consiguieron los nombres de los sitios, también los correos de las cuentas, números telefónicos y de manera parcial los nombres de todos los contactos, es decir, solamente se recuperaron una parte de sus nombres.

La figura 5.3 muestra algunos números eliminados, los cuáles son reales, por esa razón están cubiertos, y la figura 5.4 presenta la información de algunos sitios web visitados.

Resultados obtenidos por restablecimiento de fábrica

Los resultados del borrado del restablecimiento de fábrica fueron similares a los del borrado normal, sin embargo, la cantidad de información recuperada fue menor en proporción a la del borrado normal. Tal y como se puede distinguir en la figura 5.5, que muestra algunas imágenes recuperadas del dispositivo Sony, donde fueron un 17.5 % las imágenes que tenían un estado medianamente visible, un 8.7 % estaba en buen estado, del cuál el 8.7 % la mayoría era de un tamaño más pequeño al original.

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0-dev ($Rev: 10844 $)
# Feature-Recorder: telephone
# Filename: imagenXperia5.dd
# Histogram-File-Version: 1.1
n=50    +524
n=40    +525
n=38    +524
n=33    +524
n=32    +524
n=32    +524
n=32    +524
n=32    +524
n=32    +527
n=29    +525
n=29    +525
n=29    +529
n=29    8772
n=28    8778
```

Figura 5.3: Números telefónicos eliminados recuperados después de realizar el borrado normal.

{"load_images_requests_per_session":0,"load_images_snackbars_shown_per_session":0,"was_used_this_session":false},"dns_prefetching":{"host_referral_list":[2,["http://netflix.com/",["https://www.netflix.com/",2.34]],["http://www.cs.cinvestav.mx/",["https://www.cs.cinvestav.mx/",2.34]],["https://www.cs.cinvestav.mx/",["https://www.cs.cinvestav.mx/",28.51999999999999]],["https://mail.google.com/",["https://accounts.google.com/",1.01274064,"https://mail.google.com/",0.7300639251840001]],["https://accounts.google.com/",["https://accounts.google.com/",2.34,"https://accounts.youtube.com/",1.8843999999999999,"https://ssl.gstatic.com/",3.0131999999999994,"https://www.google.com/",1.5444]],["http://mx.yahoo.com/",["https://espanol.yahoo.com/",2.34]],["http://www.quien.com/",["http://b.scorecardresearch.com/",2.6799999999999997,"http://cdn.expansion.mx/",2.34,"http://cdn.taboola.com/",2.34,"http://config.seedtag.com/",2.34,"http://metrics.brightcove.com/",3.3599999999999994,"http://www.google-analytics.com/",2.34,"http://www.googleadservices.com/",2.6799999999999997,"http://www.quien.com/",3.3599999999999994,"https://cdn.expansion.mx/",9.819999999999997,"https://connect.facebook.net/",2.34]],["https://espanol.yahoo.com/",["https://beap.gemini.yahoo.com/",2.34,"https://espanol.yahoo.com/",2.34,"https://geo.yahoo.com/",1.8843999999999999,"https://s.yimg.com/",14.627599999999996,"https://sb.scorecardresearch.com/",1.8843999999999999,"https://udc.yahoo.com/",3.2443999999999993]],["https://s.yimg.com/",["https://

Figura 5.4: Nombres de sitios que se visitaron en los navegadores.

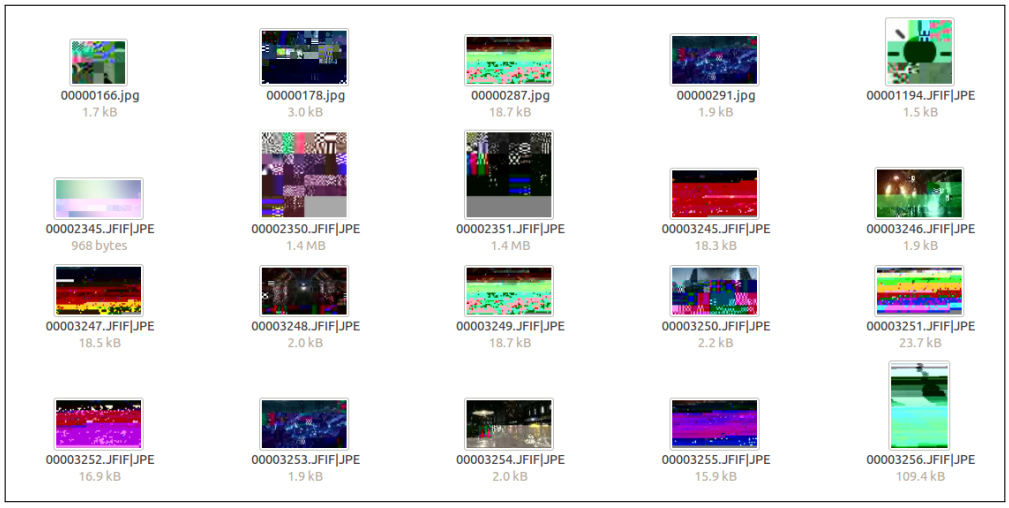


Figura 5.5: Imágenes recuperadas después de realizar el restablecimiento de fábrica.

De la misma manera, en la tableta Dell, solamente un 34 % de las imágenes se recuperaron en un estado medianamente distinguible. También, un aproximado de 10 % de las imágenes obtenidas del teléfono Samsung estaban en un estado medianamente distinguible y solamente el 6.29 % se

encontraban en buen estado. El dispositivo con la versión 6.0, resultó con la menor cantidad de imágenes recuperadas, porque solamente el 6.4 % de las imágenes recuperadas se encontraba en un estado medianamente distinguible, pero sólo el 0.9 %se hallaba en un buen estado.

Respecto a la recuperación de los documentos, una tercera parte de los mismos fue recuperada en el Sony, de los cuales solo un archivo PDF conservó sus metadatos originales. La tableta Dell fue posible obtener un 29 % de este tipo de archivos, pero en ninguno se les podía ver su contenido, ni siquiera sus metadatos. Mientras tanto, en el teléfono con Samsung más del 60 % de los documentos fueron recuperados, pero también sin el contenido legible, aunque el 29 % sí tenía sus metadatos. Mientras tanto, en el Motorola, hubo una recuperación de más del 80 % de documentos, pero ninguno tenía contenido legible y tampoco contaban con metadatos.

En consideración a la información del historial de los navegadores, en el dispositivo Sony no se recuperó nada al respecto, mientras que el la tableta Dell sí se obtuvieron la mitad de nombres de los sitios visitiados desde los navegadores. Entre tanto, en el teléfono Samsung, también se obtuvieron todos los nombres de los sitios visitados. En el Motorola, solo se recuperó una página web.

Para la parte de información relacionada a los contactos o números telefónicos, hay que destacar que en los teléfonos con las versiones 4.3 y 5.1 se consiguieron números telefónicos y en el Sony se obtuvieron nombres de contactos. No obstante, eso no sucedió en los otros dos dispositivos.

5.2. Recuperación de la información generada del bluetooth

Para recuperar la información del bluetooth mencionada en la sección 4.2 en la página 53, se ideó el siguiente escenario.

- **Escenario 3** Enviar archivos de un dispostivo D_1 a un dispositivo D_2 y hacer lo mismo de D_2 a D_1. Después, eliminar esos archivos, obtener la base de datos `btopp.db` y la bitácora `btsnoop_hci.log`.

La forma en cómo obtener `btsnoop_hci.log` se menciona en la sección 4.2 en la página 53. Para poder conseguir `btopp.db` es necesario tener privilegios de `root`, o tener desbloqueado el *bootloader*, no así con el archivo `btsnoop_hci.log`.

Resultados obtenidos

Dentro de la base de datos `btsnoop_hci.log` hay una tabla llamada btopp que tiene los siguientes campos que son de interés:

- `id`. Identificador del registro, es una clave primaria
- `uri`. *Uniform resource identifier*, del archivo proveedor de contenido
- `hint`. Nombre del archivo enviado/recibido
- `data`. Directorio donde se almacena el archivo, solo tiene valores diferentes de `null` para los archivos que son recibidos.
- `mimetype`. Tipo de archivo que se envió o se recibió.
- `destination`. Dirección MAC del dispositivo del que se recibe/envía el archivo

Esto significa que los archivos que se envíen o se reciban aunque estén eliminados, el nombre del archivo, así como la dirección MAC del dispostivo que lo envió/recibió, aún permanecen en la base de datos.

Aunque la bitácora `btsnoop_hci.log` no tiene el contenido del archivo enviado/recibido, sí contiene metadatos como lo es el nombre original del archivo o el autor del mismo. Esto significa que si el usuario u_1 que recibe el archivo a_1, modifica alguno de estos metadatos como el nombre del archivo, entonces u_1 se llamará $modif(u_1)$. Sin embargo, a pesar de que u_1 sea capaz de alterar esta información, es posible recuperar los metadatos originales de a_1, a pesar de haber sido cambiados. Por ejemplo, la figura 5.6, muestra el nombre original del archivo y su tipo.

Algo notable en esta prueba es que los archivos que fueron enviados y recibidos por bluetooth se muestran en una lista de una actividad desde la aplicación de configuración, aunque hayan sido eliminados.

Cada elemento de esta lista incluye el nombre del archivo, el usuario al que se envió o recibió el archivo, la fecha de envío/recibo y el tamaño del archivo. Sin embargo, si un archivo a_1 está eliminado, cuando el usuario toque el elemento e_1 de la lista que hace referencia a a_1 en la pantalla, Android mostrará un mensaje indicando que el archivo no existe y revocará a e_1 de esa lista.

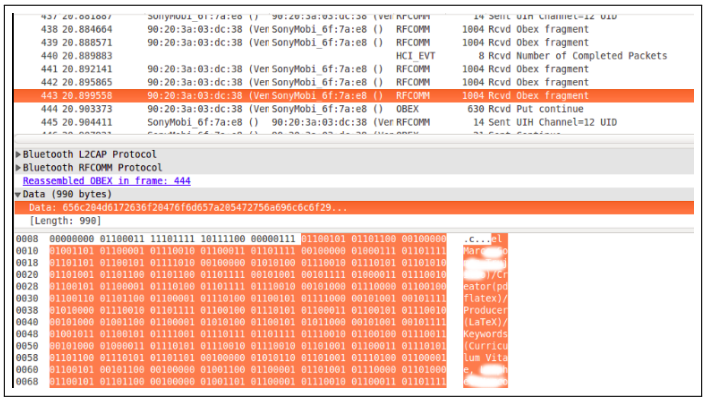


Figura 5.6: Archivo `btsnoop_hci.log`

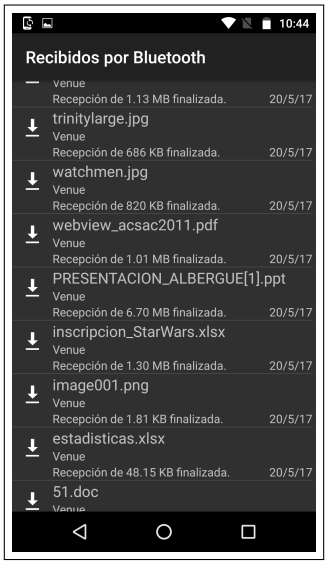


Figura 5.7: Lista que muestra los archivos enviados y recibidos por bluetooth.

	_id	uri	hint	_data	mimetype	direction	destination	visibility
1	1	content://media/external/file/	Chapter17.pdf	/storage/emulated/0/bluet	application/pdf	1	90	1
2	2	content://media/external/file/	51.doc	/storage/emulated/0/bluet	application/msword	1		1
3	3	content://media/external/file/	estadisticas.xlsx	/storage/emulated/0/bluet	application/vnd.openxmlformats-off	1		1
4	4	content://media/external/imag	image001.png	/storage/emulated/0/bluet	image/png	1		1
5	5	content://media/external/file/	inscripcion_StarWars.xlsx	/storage/emulated/0/bluet	application/vnd.openxmlformats-off	1		1
6	6	content://media/external/file/	PRESENTACION_ALBERGUE[1].ppt	/storage/emulated/0/bluet	application/vnd.ms-powerpoint	1		1
7	7	content://media/external/file/	webview_acsac2011.pdf	/storage/emulated/0/bluet	application/pdf	1		1
8	8	content://media/external/imag	watchmen.jpg	/storage/emulated/0/bluet	image/jpeg	1		1
9	9	content://media/external/imag	trinitylarge.jpg	/storage/emulated/0/bluet	image/jpeg	1		1
10	10	content://media/external/imag	IMG_20170420_095750.jpg	/storage/emulated/0/bluet	image/jpeg	1		1
11	11		VID_20170317_103337.mp4	/storage/emulated/0/bluet	video/mp4	1		1
12	12	content://com.google.android.	images.jpg		image/jpeg	0		1
13	13	content://com.google.android.	IMG_20170520_152704214_HDR.jpg		image/jpeg	0		1
14	14	content://com.google.android.	IMG_20170520_152701963.jpg		image/jpeg	0		1
15	15	content://com.google.android.	Screenshot_20170402-143622.png		image/png	0		1
16	16	content://com.google.android.	Screenshot_20170402-143932.png		image/png	0		1

Figura 5.8: Tabla btopp de la Base de datos btopp.

5.3. Consumo de recursos de aplicaciones

Se realizó una medición del consumo de memoria RAM y CPU que consumen las aplicaciones preinstaladas de los fabricantes de los dispositivos. Esto fue para verificar si las aplicaciones medidas realizaban un consumo excesivo que fuera sospechoso para realizar procesos que pusieran en riesgo la seguridad del usuario. Para lo cuál se realizó la siguiente metodología.

- **Paso 1** Sea $recurso_i = \{$ Memoria RAM, CPU $\}$
- **Paso 2** Usar la aplicación y medir un elemento de $recurso_i$.
- **Paso 3** Cambiar al fondo la aplicación y medir el mismo elemento de $recurso_i$.
- **Paso 4** Cerrar la aplicación y medir el mismo elemento de $recurso_i$.

El paso 3 se refiere a dejar de usar la aplicación e irse al menú principal o abrir otra aplicación, ya sea apretando los botones *Home* (ir al menú principal), Volver, o abrir otra aplicación desde el botón buscar pero sin cerrar la aplicación.

Para realizar estas mediciones se ocupó el comando `dumpsys` de `adb`, en el Apéndice B, las tablas B.1, B.3, B.2 y B.4 muestran los resultados de las mediciones realizadas a los cuatro dispositivos. Para ilustrar mejor estas mediciones, en las gráficas de las figuras B.1, B.3, B.2 y B.4, también del Apéndice B, se muestran los resultados de los promedios del consumo de estos recursos de estas aplicaciones.

Observaciones de las mediciones de los recursos de las aplicaciones

Cuando se realizaron estas mediciones, se registraron los eventos que generaban las aplicaciones durante estas pruebas con el uso del comando `logcat`, de `adb`.

Observaciones en las mediciones del dispositivo Sony

En el dispositivo con la versión Jelly Bean, se encontraron algunas anomalías durante esta prueba. Por ejemplo, Facebook, que tiene un consumo de CPU de casi un 9 %, convirtiéndolo en la tercera aplicación con mayor procesamiento y un consumo de memoria de 60 MB.

En las bitácoras generadas de esta aplicación, se detectó que no solamente tiene acceso al correo del usuario que está registrado en esta aplicación (ejemplo123@gmail.com), sino también el correo que está en la aplicación de correo electrónico (otroejemplo123@gmail.com), que no es el

mismo, como lo muestra la figura 5.9[1], esto lo puede hacer porque tiene el permiso peligroso de buscar cuentas en el dispositivo (GET_ACCOUNTS), y esto no es notificado al usuario. Asimismo,

```
I/ActivityManager( 869): START u0 {act=android.intent.action.VIEW dat=http://m.fa
cebook.com/r.php?app_lox_device_id=ba95f543-aae5-49c6-acc7-87b716de7b7a&cid=350685
531728&locale=es_LA&contactpoints=ejemplo123@gmail.com, otroejemplo123@hotmail.com
cmp=com.android.chrome/com.google.android.apps.chrome.Main} from pid 11191
```

Figura 5.9: Bitácora de la aplicación Facebook que muestra la obtención de los correos presentes en el dispositivo, incluso si éstos no están asociados a esta aplicación.

en la aplicación Twitter, el nombre del usuario siempre está presente durante la generación de las bitácoras. Del mismo modo, la aplicación Xperia Lounge tiene un promedio de alrededor de 8 % de consumo de CPU y casi 80 MB de memoria, siendo la cuarta aplicación con mayor consumo de estos recursos, envía a la dirección http://ad-x.co.uk los datos del teléfono, como el modelo, la versión de android y el idioma en el que está configurado.

Observaciones en las mediciones del dispositivo Dell

Para las aplicaciones de Kitkat (Dell), las gráficas de la figura B.3 muestran que la aplicación Skitch, que tuvo un consumo de procesamiento en promedio de casi 10 % (es la cuarta aplicación con mayor procesamiento), y casi 50 MB de memoria RAM (es la quinta aplicación con mayor consumo de memoria). Durante la medición del consumo de recursos, se detectó que ésta aplicación hace varios llamados a la aplicación de cámara enviándo el intento `ACTION_IMAGE_CAPTURE`, que se ocupa de capturar una imágen con la aplicación Cámara.

Observaciones en las mediciones del dispositivo Motorola

De manera similar, en el dispositivo con Marshmallow (Motorola), la aplicación "Ayuda" tiene un consumo de CPU alrededor del 4 % y casi 50 MB de memoria, convirtiéndola en la quinta aplicación con mayor consumo de procesamiento y la segunda con mayor consumo de memoria, se observó que esta aplicación hace llamados a la aplicación de mensajes.

Asimismo, se notó en la medición de aplicaciones como Cámara, Calendario, Fotos y Mensajes, se hacen llamadas al paquete com.amazon.phoenix porque hay un paquete de amazon en el dispositivo; com.amazon.mShop.android, es decir, que hay una aplicación presente en este dispositivo, pero no está instalada. Además, la aplicación cámara, obtiene el correo del usuario y trata de acceder a la base de datos de la aplicación de gmail. De la misma manera, la aplicación de moto y

[1]Los correos mostrados en la figura 5.9 no son los mismos que se ocuparon en las pruebas

música inicia la aplicación cámara con el intento `android.intent.action.MAIN`.

Observaciones en las mediciones del dispositivo Samsung

En las aplicaciones de este dispositivo no se detectaron actividades sospechosas. Sin embargo, cabe resaltar que una característica de este dispositivo es la contínua exploración del NFC, mecanismo que no tiene este teléfono inteligente.

5.4. Pruebas de seguridad del protocolo TLS/SSL en los navegadores preinstalados

Se realizó una evaluación de la seguridad del protocolo SSL/TLS en los navegadores preinstalados de los cuatro dispositivos, a través de la versión de SSL/TLS y los conjuntos de cifrados que soportan. Para realizar esta tarea, se utilizó la herramienta Qualys SSL Labs [16], la cuál, realiza una evaluación de SSL/TLS del navegador y también se encarga de verificar su vulnerabilidad ante los ataques Logjam, Freak y Poodle. En la tabla 5.1, se puede observar que los únicos dispositivos con navegadores que son vulnerables ante los tres ataques mencionados son el de la versión con Jelly-Bean (Sony) con el navegador Google Chrome de fábrica y el Kitkat (Dell) con el navegador Android AOSP y también con el Google Chrome que trae de fábrica.

VERSIÓN DEL DISPOSITIVO	NAVEGADOR	VERSIÓN MÁXIMA DE TLS QUE SOPORTA	VULNERABLE A LOGJAM	VULNERABLE A FREAK	VULNERABLE A POODLE
Marshmallow	Chrome Fábrica	1.2	No	No	No
Marshmallow	Chrome Actualizado	1.2	No	No	No
Kit-kat	Android AOSP	1.2 y SSL 3	Sí	Sí	Sí
Kit-kat	Chrome Fábrica	1.2	Sí	Sí	Sí
Kit-kat	Chrome Actualizado	1.2	No	No	No
Jelly-Bean	Chrome Fábrica	1.1 y SSL 3	Sí	Sí	Sí
Jelly-Bean	Chrome Actualizado	1.2	No	No	No
Lollipop	Samsung	1.2	No	No	No
Lollipop	Chrome Fábrica	1.2	No	No	No
Lollipop	Chrome Actualizado	1.2	No	No	No

Tabla 5.1: Resultados de la evaluación de los navegadores con Qualys SSL Labs, que muestra los navegadores que son vulnerables a los ataques Logjam, Freak y Poodle, así como la máxima versión de TLS/SSL que soportan.

Asimismo, la versión TLS que soportan la mayoría de estos navegadores es la 1.2, que, de acuerdo con Ristic [16], es la que hasta el momento ofrece algoritmos criptográficos modernos. Los únicos navegadores que tienen una versión inferior a ésta versión es el Google Chrome de Fábrica del dispositivo con Jelly-Bean y el de Android AOSP del dispositivo con Kitkat, el primero porque

tiene soporte hasta la versión 1.1, y ambos navegadores tienen soporte para SSL 3, que, de acuerdo con Ristic [16], es obsoleto e inseguro.

Conjuntos de Cifrado soportados por los navegadores

En lo que se refiere a los conjuntos de cifrado (*Cipher suites*), los resultados que arrojó Qualyss SSL muestra que la mayoría de los navegadores contienen conjuntos inseguros y débiles, unos más que otros. Los conjuntos débiles utilizan el algoritmo 3DES, que como se mencionó en el Capítulo 1, es necesario su desuso [16] y los conjuntos inseguros utilizan el cifrado RC4, que se considera inseguro o emplean ambos.

Los dispositivos que presentan un mayor número de estos conjuntos inseguros y débiles son el teléfono de la versión 4.3 con el Google Chrome de Fábrica, los navegadores Chrome de fábrica y Android AOSP de la tableta con Kitkat junto con el navegador Samsung. Los navegadores Google Chrome de fábrica y actualizados de los teléfonos con Lolipop y Marshmallow, así como los Google Chrome actualizados de los otros dispositivos, son los que presentaron el menor número de conjuntos débiles.

Además, algunos navegadores no soportan el grapado OCSP (*OSCP stapling*), mecanismo que se encarga de la gestión de los estados de la revocación de los certficados digitales X.509 [60]. Esto sucede con los navegadores de los dispositivos con Kitkat, Jelly-Bean y el navegador Samsung.

5.5. Ataques de Husmeo con Punto de Acceso Malicioso

Se implementó el Ataque de Husmeo con el Punto de Acceso (PA) Malicioso/Ilegítimo basado en la figura 4.5a para probar la seguridad del motor de renderizado de Android (WebKit y Chromium) de diferentes versiones ante este ataque, utilizando los navegadores preinstalados en los dispositivos. También se realizó lo mismo con las aplicaciones preinstaladas de los dispositivos y la API de navegación (Webview).

5.5.1. Modelo del ataque

Como Punto de Acceso se ocupó una estación de trabajo (Computadora Personal) con Kali Linux 2016.1 como sistema operativo. Para que los dispositivos que se conecten a este PA puedan tener una dirección IP, se configuró un servidor DHCP, utilizando la herramienta `Internet Systems Consortium DHCP Server` y para brindar el servicio DNS se empleó `dnsmasq`.

De igual forma, para llevar a cabo la degradación de HTTPS a HTTP, siguiendo el modelo de la figura 4.6, se ocupó la herramienta `SSLStrip`, de Marlinspike [51]. Así también, para poder ejecutar el envenenamiento ARP se utilizó la herramienta `ettercap`. De manera similar, para monitorear los paquetes que transitan en la red inalámbrica, se utilizó `wireshark` y `driftnet`, éste último es una herramienta que captura y visualiza las imágenes que son transmitidas por TCP, pero únicamente si están en HTTP. La figura 5.10 ilustra la forma en cómo se implementó este PA. El SSID de la red fue llamada “computacion5”.

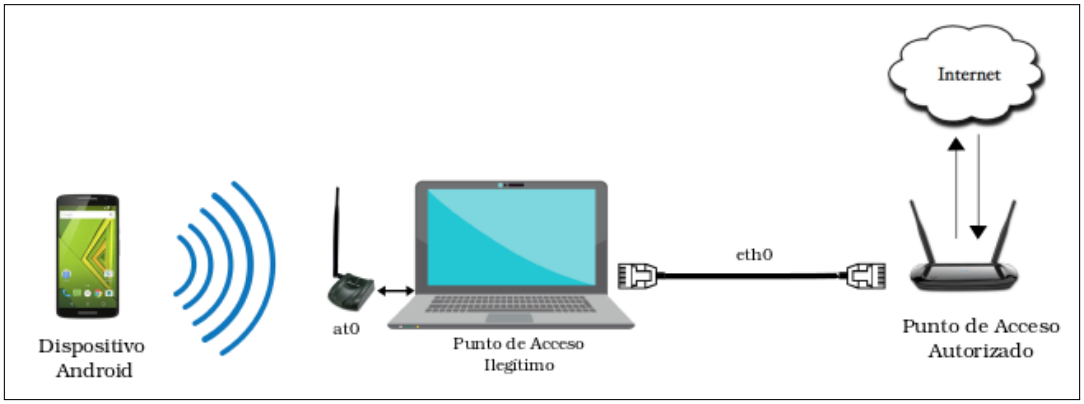


Figura 5.10: Modelo del Ataque de Husmeo con PA Ilegítimo para obtener información de los dispositivos Android.

Como se puede notar en la figura 5.10 la estación de trabajo para el PA tiene dos interfaces, una para brindar conexión a internet `eth0` y otra para la red inalámbrica, donde se establecerá la conexión con los dispositivos Android, `at0`. Para manejar la interfaz `eth0`, se empleó el adaptador de red alámbrica que está integrado en la estación de trabajo y está conectada a una Red de Area Local con conexión a internet suministrada por el Punto de Acceso Autorizado.

Mientras que para la interfaz `at0`, se utilizó el adaptador de red con un controlador ATHEROS, este adaptador opera en modo monitor y en modo maestro (*master mode*). El modo monitor, otorga a la computadora la capacidad de monitorear los paquetes que están circulando en la red y el modo maestro le permite comportarse como un Punto de Acceso.

5.5.2. Ataques a los navegadores preinstalados

Con el PA Falso se efectuaron ataques a los navegadores preinstalados de la Tabla 5.2, que utilizan los motores de renderizado del sistema Webkit y Chromium. Como se puede observar en la Tabla 5.2, las versiones de Google Chrome no son las mismas, y no corresponden a la versión más

actual de este navegador. Debido a esto, Google Chrome se actualizó en todos los dispositivos a la versión más reciente al momento de realizar estas pruebas, que fue la 56.0.2924.87.

DISPOSITIVO	VERSIÓN DE ANDROID	NAVEGADOR	VERSIÓN DEL NAVEGADOR	MOTOR DE RENDERIZADO
Sony Xperia SP C5306	4.3	Google Chrome	28.0.1.500.94	WebKit
Motorola Moto G	6.0	Google Chrome	49.0.2623.91	Chromium
Dell Venue 7	4.4 4.4	Google Chrome Android AOSP	36.0.1985.135 4.4.4-65	Chromium WebKit
Samsung Galaxy Grand Prime	5.1 5.1	Google Chrome Samsung	50.0.2661.89 3.3.38.141-0	Chromium Chromium

Tabla 5.2: Navegadores preinstalados de los dispostivos utilizados

Durante el ataque, con los navegadores de la tabla 5.2, se visitaron los sitios web más buscados de México. Adicionalmente, también se empleó el sitio del Departamento de Computación del Cinvestav, con más hincapié en el servicio de correo electrónico. De acuerdo con Alexa [61], estos son los sitios más visitados en México y que hacen uso del protocolo SSL/TLS para el momento en que se realizó el ataque.

- Google
- Facebook
- Hotmail/Outlook
- Yahoo
- Wikipedia
- Caliente
- Adf
- Netflix
- Uptodown
- Twitter
- Disqus

Para entrar en estos sitios en cada navegador, se eligieron cuatro escenarios que exponen las formas en la que se pueden acceder a ellos.

- **Escenario 1**. Ingreso de la dirección completa del sitio desde la barra de dirección, por ejemplo `www.nombre_sitio.com`.
- **Escenario 2**. Desde la barra de navegación escribir solamente el nombre del sitio, como por ejemplo, en lugar de colocar `www.nombre_sitio.com`, solo escribir `nombre_sitio`.
- **Escenario 3**. Búsqueda desde Google en su versión HTTP.
- **Escenario 4**. Ingreso al sitio en su versión HTTPS y después, borrar la 's' de `https` en la barra de dirección del navegador.

Cada uno de estos sitios se accedieron desde estos escenarios, y manejan la autenticación de usuarios para iniciar sesión en HTTPS. Por tal motivo, en este ataque se hace énfasis en la autenticación de usuarios.

5.5.3. Degradación de Google a HTTP

Hay que destacar que cuando el usuario visita un sitio desde algún navegador, en muchas ocasiones lo hace desde el buscador de Google. En el caso de Google Chrome, si escribe lo que va a buscar desde la barra de navegación, lo llevará al sitio de Google, con los resultados de su búsqueda, y casi siempre en primer lugar aparece el sitio al que desea acceder. También, frecuentemente primero accede a la página de Google y después realiza la busqueda del sitio. Es por esa razón que se establecieron los escenarios 2 y 3.

Por lo tanto, se determinó en cuáles circunstancias el sitio de Google puede degradarse a HTTP en los dispositivos, y esto se puede visualizar en la tabla 5.3, donde la primera y segunda columna indican la versión del dispositivo y el navegador, respectivamente.

La tercera columna muestra cuando se ingresa a Google desde la URL, es decir, ingresar "www.google.com" desde la barra de dirección del navegador. La cuarta columna indica el escenario desde la "BÚSQUEDA AUTOMÁTICA", esto significa que con ingresar cualquier búsqueda desde la barra de navegación inmediatamente se muestra la página de Google. Y finalmente, en la última columna está el "BORRADO DE LA 'S' DE HTTPS", que señala cuando se borra la 's' de https://www.google.

com, es decir, cuando ya está cargada en su version HTTPS.

VERSIÓN DEL DISPOSITIVO	NAVEGADOR	INGRESO DESDE URL	BÚSQUEDA AUTOMÁTICA	BORRADO DE LA 'S' DE HTTPS
Marshmallow	Chrome Fábrica	https	https	http
Marshmallow	Chrome Act.	https	https	http
Kitkat	Android AOSP	http	https	http
Kitkat	Chrome Fab.	http	https	http
Kitkat	Chrome Act.	https	https	http
Jelly-Bean	Chrome Fab.	http	https	http
Jelly-Bean	Chrome Act.	http	https	http
Lollipop	Samsung	http	https	http
Lollipop	Chrome Fab.	https	https	http
Lollipop	Chrome Act.	https	https	http

Tabla 5.3: Circunstancias donde Google se degrada a HTTP

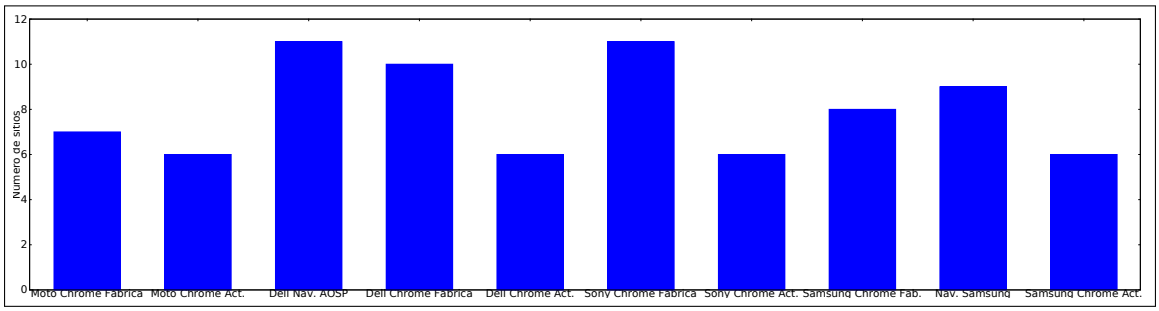
5.5.4. Resultados obtenidos de los ataques a los navegadores

Como se puede observar en la figura 5.11, el número de sitios degradados varía de acuerdo al dispositivo y al navegador. Con excepción del Escenario 2, donde los sitios siempre entraban en HTTPS, en los demás escenarios sí hubo degradaciones a HTTP. La figura 5.11a muestra los resultados del Escenario 1, la figura 5.11b los resultados del Escenario 3, la Figura 5.11c los del Escenario 4, y finalmente, la figura 5.11d presenta la suma del total de sitios de los tres escenarios anteriores.
En la figura 5.11c, se puede observar que para el Escenario 4, todos los dispositivos tuvieron el mismo número de sitios que se degradaron a http con borrar la "s"de https en la barra de navegación. Estos sitios fueron CS Cinvestav, Disqus, adf y uptodown.

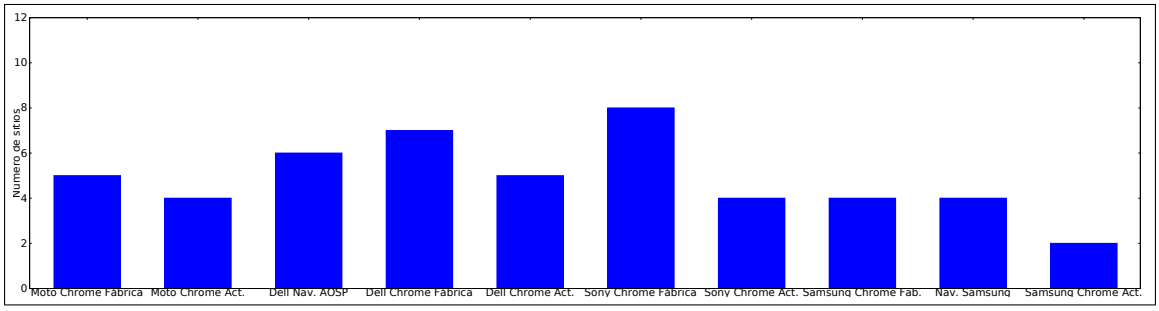
Navegadores del dispositivo con la versión Jelly-Bean

De acuerdo con las gráficas de las figuras 5.11a, 5.11b y 5.11d, el navegador Google Chrome que venía de fábrica en el dispostivo con la versión Jelly Bean, es el que tuvo el mayor número de sitios con degradaciones a HTTP.

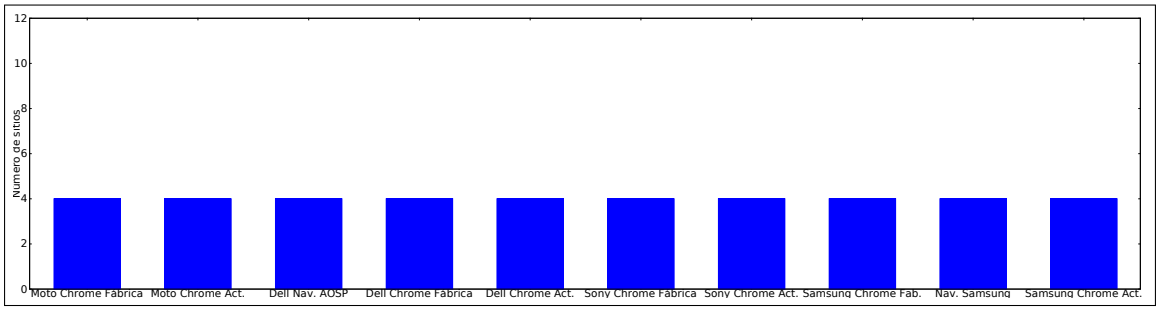
- **Google Chrome de fábrica.** Junto con el navegador AOSP de la tableta Dell, este navegador es el que tiene más sitios con degradación HTTPS a HTTP en el escenario 1 (figura 5.11a). De igual manera, en el escenario 3 mantiene el primer lugar (figura 5.11c). También, es el que tiene la mayor suma del número de sitios degradados de acuerdo a la figura 5.11d. En casi todos los sitios evaluados se obtuvieron datos sensibles como credenciales de autenticación con excepción de Caliente y Netflix.



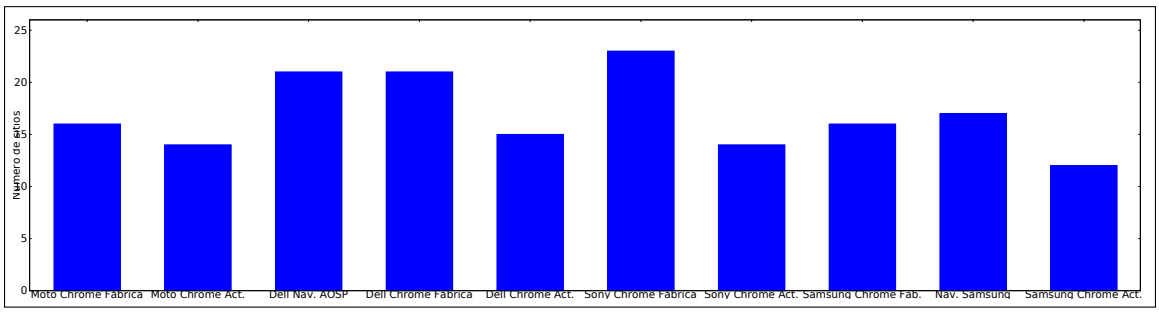
(a)



(b)



(c)



(d)

Figura 5.11: Sitios en los navegadores con degradación a http de los diferentes dispositivos; Sony (v. 4.3), Dell (v. 4.4), Samsung (v. 5.1) y Motorola (v. 6.0): (a) Número de sitios degradados a HTTP en el Escenario 1; (b) Número de sitios degradados a HTTP en el Escenario 3; (c) Número de sitios degradados a HTTP en el Escenario 4; (d) Suma total de los sitios degradados a http de todos los escenarios.

- **Google Chrome actualizado.** El número de sitios con degradación a HTTP fué menor en comparación a la versión de fábrica de esta misma versión de Android. En el escenario 1, fue de los navegadores que resultó menos afectado, junto con Google Chrome Actualizado de los otros dispositivos. De manera similar, sucedió lo mismo para el escenario 3.

  Por lo tanto, del total de número de degradaciones de todos los escenarios, este fue uno de los navegadores con una de las menores cantidades de sitios en la suma total de la figura 5.11d, ya que tuvo el mismo impacto que los navegadores Chrome Actualizado de Motorola, pero tuvo un mayor número que el Chrome Actualizado de Samsung. Hubo captura de información sensible, principalmente de credenciales de autenticación de sitios como Facebook, Hotmail, Adf, Uptodown, Disqus y CS Cinvestav. Sin embargo, en Gmail, Twitter, Yahoo, Wikipedia, Caliente y Netflix no hubo captura de información sensible.

Navegadores del dispositivo con la versión Kitkat

El navegador de la AOSP, junto con Google Chrome de fábrica en el dispositivo con Kitkat también resultaron afectados por este ataque. En el escenario 2 y la suma total de sitios fueron los que más tuvieron degradación, después del Google Chrome de fábrica de la versión 4.3.

- **Android AOSP.** En el escenario 1 tiene el primer lugar con degradación a HTTP junto con el Chrome de fábrica del dispositivo con Jelly Bean. En el escenario 3, solamente fué superado por Google Chrome de fábrica del mismo dispositivo. En todos los sitios examinados con excepción de Caliente y Netflix pudieron obtenerse datos sensibles.

- **Google Chrome de fábrica.** Este navegador también resultó afectado ante este ataque, ya que en el Escenario 1, es el segundo con más sitios afectados, al igual que en el Escenario 3. En cuanto al total de degradaciones, junto con el navegador de la AOSP, es el segundo lugar, después del Chrome de Fábrica del dispositivo Sony. De manera similar, se obtuvieron datos de la mayoría de los sitios, pero a diferencia del navegador de la AOSP, no se capturaron datos de Disqus ni de Twitter. Asimismo, tampoco se obtuvieron datos de Netflix o de Caliente.

- **Google Chrome actualizado.** En este caso, este navegador fue el que tuvo el menor número de sitios con datos sensibles en claro en este dispositivo. En el escenario 1, junto con los otros Chrome Actualizados, fue de los navegadores menos afectados. En el Escenario 3 aumentó un poco el número de sitios en HTTP. Si bien se recuperó información sensible en algunos sitios, esto no sucedió con la cuenta de Google, Twitter, Yahoo, Wikipedia, Caliente ni Netflix.

Navegadores del dispositivo con la versión Lollipop

Para el caso de los navegadores correspondientes a Lollipop, en general tuvieron menor impacto por este ataque que los navegadores de Kitkat y Jelly Bean.

- **Google Chrome de fábrica.** Para este navegador, en el caso del Escenario 1 tuvo más casos de degradación que el Chrome de Fábrica de Marshmallow y los Chrome actualizados de los todos los dispositivos probados. Hubo captura de información de muchos sitios menos de la cuenta de Google, Twitter, Caliente ni Netflix.

- **Google Chrome actualizado.** Este navegador fue uno de los que tuvo menor número de sitios en degradarse, sobre todo en el Escenario 3. Se capturó información de pocos sitios sitios, porque no se obtuvieron datos de la cuenta de Google, Twitter, Facebook, Yahoo, Wikipedia, Caliente ni Netflix.

- **Navegador Samsung.** En el caso del navegador propio de este fabricante, el grado de susceptibilidad que tiene ante este ataque es medianamente alto, dado que en el Escenario 3 únicamente tuvo más casos de degradación que el Google Chrome actualizado también de Samsung. Sin embargo, tomando en cuenta la suma de todas las degradaciones (figura 5.11d), solamente es superado por los Chrome de Fábrica de Sony, Dell y el Android AOSP de Dell. Los únicos sitios donde no se capturaron datos fueron Gmail, Twitter, Wikipedia, Caliente ni Netflix.

Navegadores del dispositivo con la versión Marshmallow

Aunque la resistencia a estos ataques resultó ser mejor que los navegadores con Jelly Bean y KitKat, en general, los navegadores de Marshmallow resultaron ser ligeramente más vulnerables que los de Lollipop.

- **Google Chrome de fábrica.** En el Escenario 1, este navegador resultó ser más expuesto que su versión actualizada en el dispositivo con las versiónes 5.1, 4.3 y 4.4 (figura 5.11a), mientras que para el Escenario 3 solamente está por debajo de Google Chrome de Fábrica del dispositivo con Jelly Bean y los navegadores AOSP y Chrome de Fábrica de Dell, dado que solamente no hubo captura de datos en la cuenta de Google, Twitter, Wikipedia, Caliente ni Netflix. Tomando en cuenta la suma total de sitios afectados (figura 5.11d), este navegador tiene más sitios con degradación que los Chrome Actualizados de todos los dispositivos.

- **Google Chrome actualizado.** Este navegador si bien en el escenario 1 fue de los que menos sitios fueron afectados con el ataque, para los escenarios 3 y 4 tuvo más ataques que el

Chrome Actualizado de la versión Lollipop. Los sitios que no resultaron afectados fueron Google, Twitter, Facebook, Yahoo, Wikipedia, Caliente ni Netflix.

5.5.5. Información recuperada de los navegadores

En los sitios donde la degradación de HTTPS a HTTP se logró con éxito la captura de información. Cabe destacar que de esta información obtenida se recuperaron credenciales de autenticación como nombres de usuario y contraseñas en claro. Por ejemplo, la figura 5.12 muestra el nombre del correo y contraseña de un usuario de Google[2] en claro. De la misma manera, en la figura 5.13 presenta credenciales capturadas de Facebook. Y también la figura 5.14 muestra las credenciales capturadas del correo electrónico de CS Cinvestav.

En casi todas las capturas de datos las credenciales de autenticación recuperadas estaban en claro, con excepción del sitio uptodown, porque si bien se pudo obtener el nombre de usuario, la contraseña obtenida no estaba en claro, era el resumen (*hash*) de la misma.


```
2017-03-21 18:22:34,681 SECURE POST Data (accounts.google.com):
Page=PasswordSeparationSignIn&GALX=Z1YyU7-
GRrk&gxf=AFoagUX_kSDgyDG0RDzc169FOZKCWj1VqQ%3A1490141641043&continue=http%3A%2F
%2Fmail.google.com%2Fmail
%2F&service=mail&rip=1&sacu=1&ProfileInformation=&SessionState=&_utf8=
%E2%98%83&bgresponse=
%21lpWllbRCG_EsiCctDElEEKiamemqYwYCAAAAxlIAAAATmQEjJCwIhq9IwZ9x2kAi9fErnKgJxq7scKiF
5LJnuODzEikmvJ9iXfO2fQZhP4cftp56G4eeWn4ZGAcleB5Be7J-
Sz402QN4HPyyXnL9I54hzQaHbyTxGACVkUHsYUTOffLEv5zA1u5udpibhzUHOFP7uN1FyOGaJdk29l45rer
r_TbYpEb74msfLX0TfpkKp3jDYDkuF6iFUU1-
j9iUUTXPqYt06dLdzv6LIDtIP2JJQc7NmvYw86DhDIJlkUN6HpOjjwGlQVqHAu7LRZtoFxtOkMN5P18KJrz
36V8oDwQ0dzHB1Dt0OW2WOcZA6VARfD5iVViMqOT889iDFyH1YEX3CINhqkDr4GooJH97QYvEQRcY7E8S7u
WXEh5eg7QV_HyLnZo2&pstMsg=1&checkConnection=&checkedDomains=youtube&identifiertoken
=&identifiertoken_audio=&identifier-captcha-
input=&Email=[redacted]&Passwd=[redacted]&PersistentCookie=yes
```


Figura 5.12: Credenciales de autenticación de una cuenta de Google. El nombre de usuario es una dirección de correo electrónico y está ubicado delante de “Email=” y la contraseña está localizada delante del texto “Passwd=”


```
2017-03-19 15:01:52,729 SECURE POST Data (www.facebook.com):
lsd=AVreOyLi&email=[redacted]&pass=[redacted]&timezone=360&lgndim=
eyJ3IjozNjAsImgiOjY0MCwiYXciOjM2MCwiYWgiOjY0MCwiYyI6MzJ9&lgnrnd=140041_W_ys&lgnjs=1
489957251&ab_test_data=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAT&local
e=es_LA&next=http%3A%2F%2Fwww.facebook.com%2F&login_source=login_bluebar
```


Figura 5.13: Credenciales de autenticación de una cuenta de Facebook. El nombre de usuario es una dirección de correo electrónico y está ubicado delante de “email=” y la contraseña está localizada delante del texto “pass=”

[2] Todos los datos realizados durante los ataques fueron ficticios

```
2017-03-18 12:06:52,778 POST Data (computacion.cs.cinvestav.mx):
login_username=[redacted]&secretkey=[redacted]&js_autodetect_results=1&just_logged_in=
1
2017-03-18 12:07:58,727 SECURE POST Data (computacion.cs.cinvestav.mx):
login_username=[redacted]&secretkey=[redacted]s_autodetect_results=1&just_logged_in=1
2017-03-18 12:10:30,288 POST Data (delta.cs.cinvestav.mx):
login_username=jimenez&secretkey=[redacted]p&js_autodetect_results=1&just_logged_in=1
```

Figura 5.14: Credenciales de autenticación de una cuenta de CS Cinvestav. El nombre de usuario está ubicado delante del texto “login_username=” y la contraseña está localizada delante del texto “secretkey=”

En la intercepción de datos, con `wireshark` se capturó información como paquetes de petición y respuesta HTTP, que incluyen cookies y *user-agents*, así como el código fuente de las páginas que estaban siendo usadas por el usuario en ese momento. De igual forma, con `driftnet` se obtuvieron las imágenes que forman parte de los sitios atacados.

Recuperación de información de la pantalla de inicio de Google Chrome actualizado

Existe una funcionalidad que viene incorporada en la de la versión actualizada de Google Chrome llamada “Artículos sugeridos”, que consiste en el despliegue de noticias que pueden ser de interés para el usuario y en caso contrario, puede cambiar la configuración para que se muestren los artículos que puedan ser de su interés, como lo muestra la figura 5.15. Pero esta característica siempre está en HTTP, por lo tanto, durante el ataque con el PA Malicioso se capturaron paquetes que contenían el código HTML en esta página de inicio, y con driftnet se interceptaron las imágenes que aparecieron en esta página.

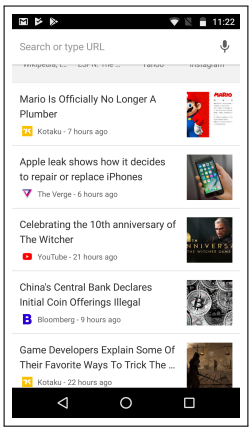


Figura 5.15: Pantalla de inicio de Google Chrome Actualizado, que está en HTTP y por lo tanto, la información visualizada puede ser interceptada por el PA Malicioso

Sitios con HTTP que tienen autenticación y manejo de sesión con HTTPS: IPN, Liverpool y Sears

Existen sitios que tienen su página principal en HTTP pero que gestionan el inicio de sesión de algunos servicios en HTTPS, por ejemplo, en el sitio del IPN, su servicio de correo electrónico está en SSL/TLS, lo mismo sucede en la compra en línea de tiendas como Liverpool y Sears. En estos tres sitios, todos los navegadores de todos los dispositivos fueron degradados a HTTP, en los cuatro escenarios durante el ataque, incluso en las secciones con HTTPS, siempre se mostraron al usuario en su versión HTTP.

La página principal del IPN es enteramente HTTP, mientras que su correo electrónico institucional es HTTPS. Tomando el papel de usuario, se visitó el sitio principal y desde ahí se accedió al correo. Durante el ataque con el PA ilegítimo, cuando se realizó la autenticación en este servicio, se obtuvieron los nombres de los correos y las contraseñas en claro, como se puede ver en la figura 5.17.

El sitio de Liverpool tiene un proceso de compra de artículos en línea; hasta que un usuario selecciona un producto de su tienda en línea, la comunicación es vía HTTP. Justo después de solicitar pagar el producto seleccionado, empieza el proceso de pago y a partir de aquí la comunicación es HTTPS. En el transcurso de este proceso, se llena un formulario con datos personales como nombre, dirección, teléfono y correo electrónico, entre otros. Enseguida, hay que escoger un método de pago, como el de tarjeta de crédito.

En la ejecución del ataque, se logró capturar casi toda esa información todos los dispositivos. No obstante, la información relacionada a la tarjeta de crédito solamente se pudo recuperar en el teléfono Motorola con el navegador Google Chrome actualizado y la tableta Dell también con Chrome actualizado (figura 5.16). En los demás dispositivos esto no se logró porque después de realizar el llenado de la información personal, se desplegaba un mensaje de error “ERR_TO_MANY _DIRECTIONS” que ya no permitía continuar con el proceso de compra.

Asimismo, con Sears el procedimiento es similar. Los resultados de esta tienda fueron similares a los de Liverpool. En todos los dispositivos se recuperaron los datos personales del usuario. Se llegó a la fase de captura de la información relacionada a la tarjeta de crédito en los dispositivos Motorola y Sony, ambos con el navegador Google Chrome Actualizado.

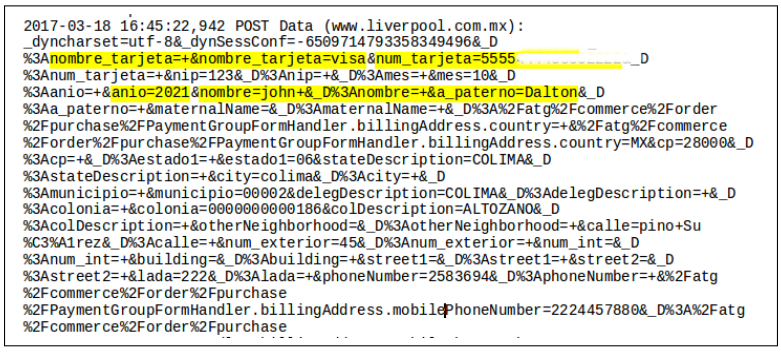


Figura 5.16: Datos de un usuario que realizó una compra en Liverpool

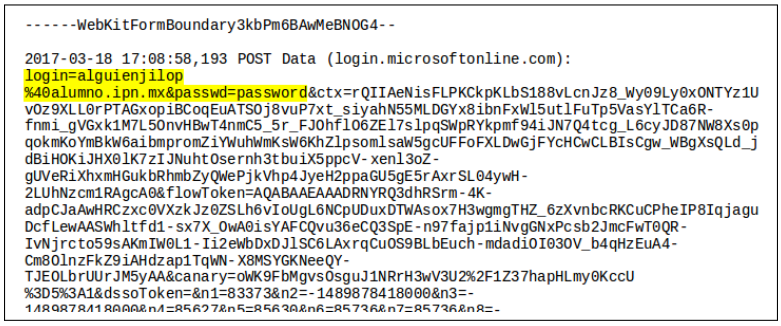


Figura 5.17: Credenciales de autenticación del correo institucional del IPN obtenidas en el ataque con el PA Ilegítimo. El nombre de usuario está ubicado delante del texto “login=” y la contraseña está localizada delante del texto “passwd=”

5.5.6. Pruebas del PA Malicioso con el SAT y el CONACyT

Se llevó a cabo el ataque con el PA ilegítimo a los sitios de las dependencias del gobierno mexicano; SAT y CONACyT (Servicio de Administración Tributaria y Consejo Nacional de Ciencia y Tecnología, respectivamente).

Pruebas del PA Malicioso con el CONACyT

El sitio de CONACyT está compuesto por varias secciones; entre las que destacan la página principal y una llamada “Trámites y servicios en línea", que se emplea para la realización de los trámites para obtener los apoyos y beneficios que proporciona esta dependencia. Existen diferentes servicios que ofrece esta sección, y los que se ocuparon para realizar el ataque fueron el Currículum Vitae Único (CVU) y el servicio de PNPC.

La mayor parte de estos sitios están en línea bajo el protocolo HTTPS. Sin embargo, existen sitios

que estan implementados en HTTP[3]. Para utilizar ambos servicios es necesario tener una cuenta de usuario y una contraseña, como el que se muestra en la figura 5.19.

La metodología que se empleó para realizar el ataque se llevó a cabo en el contexto de los cuatro escenarios descritos en la subsección 5.5.2. Para cada escenario primero se visitó la página principal. Después, acceder al sitio del CVU y si la autenticación fue exitosa, hacer uso del servicio y desconectarse. Y de manera similar, realizar lo mismo con el sitio de PNPC. Cabe aclarar que para usar el CVU, sí se empleó una cuenta de usuario real, pero para el de PNPC no.

Figura 5.18: Ventana de autenticación del servicio CVU del CONACyT

Resultados del ataque

Página principal

La página principal está en HTTPS, sin embargo con SSLStrip se logró degradar la comunicación a HTTP. Esto sucedió en todos los dispositivos y en todos los escenarios, con excepción del escenario 2. Por lo tanto, fue posible capturar la información de esa página, con driftnet se capturaron imágenes de la página principal.

CVU

Para acceder al servicio de CVU, no fue necesario realizar una degradación a HTTP, porque esta plataforma está implementada en este protocolo. Por lo tanto, fue más fácil realizar la captura de credenciales. En este servicio, están almacenados datos personales del usuario. Si el usuario acude

[3]Este ataque se llevó a cabo cuatro meses antes de la escritura de este capítulo. Para el momento de la escritura de esta tesis, hubo cambios en la estructura de los sitios del CONACyT, algunas páginas que estaban en HTTP fueron cambiadas por HTTPS

a consultar estos datos, es posible interceptarlos con el PA y de este modo el atacante podrá acceder a ellos.

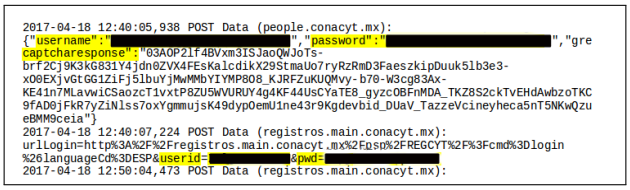


Figura 5.19: Recuperación de usuario y contraseña del CVU del CONACyT. Estas credenciales están resaltadas con amarillo. El usuario está delante del texto “username=” y la contraseña está delante del texto “password=”

PNPC

Los resultados del ataque en este servicio fueron diferentes al del CVU, porque aquí no se obtuvieron datos importantes que pudieran comprometer la privacidad del usuario. Para los escenarios 1, 3 y 4 en todos los dispositivos SSLStrip degradó el sitio a HTTP. Aunque sí hubo degradación, únicamente se visualizaba una página en blanco. Durante el ataque, algunos paquetes de datos viajaban por la red en claro, pero sólo eran los paquetes de petición y respuesta HTTP de los dispositivos.

Pruebas del PA falso con el sitio del SAT

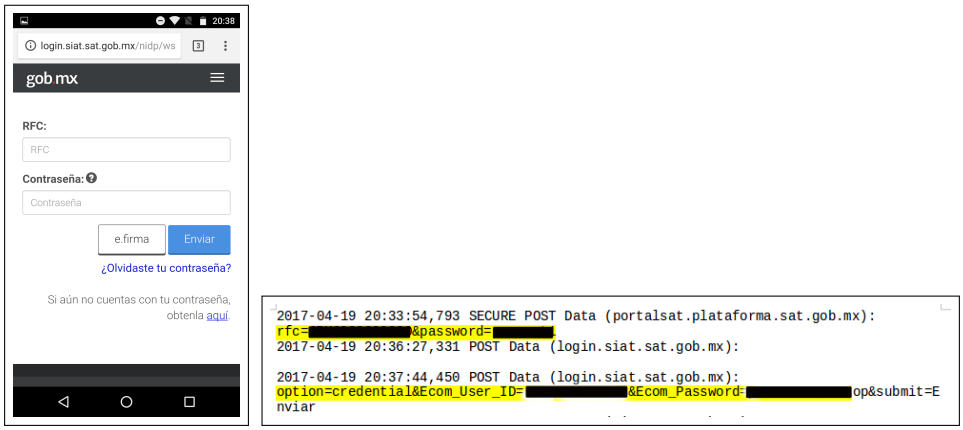
De manera similar al sitio de CONACyT, el sitio del SAT también contiene secciones que ofrecen varios servicios, y se escogieron cuatro de ellos para realizar el ataque con el PA Malicioso; “Declaraciones anuales con personas físicas", “Citas", “Trámites y servicios” y “Factura electrónica", que están bajo HTTPS.

Para poder hacer uso de cualquiera de estos servicios, es necesario contar con un usuario (el RFC) y una contraseña, con excepción de Citas, donde se deben de ingresar el número de cita y el número de confirmación. Hay que esclarecer que los datos proporcionados desde los dispositivos no son reales. Únicamente se ocuparon datos no registrados para efectos de recuperarlos en el ataque.

Resultados del ataque del SAT

En general, todos los sitios atacados por el SAT en todos los dispositivos fueron afectados por el PA Ilegítimo. Primeramente, cabe recalcar que la página principal es HTTP, por lo que la captura de la información de esta página se realizó con éxito, al interceptar parte de su código fuente y de igual forma, se pudieron obtener imágenes transmitidas con Driftnet.

En cuanto a los cuatro sitios, éstos se degradaron a HTTP y se pudieron recuperar nombres de usuarios y contraseñas en claro. Asimismo, se recuperaron números de citas y de confirmación para el caso del sitio "Citas". Por ejemplo, la figura 5.20b muestra el nombre del usuario y la contraseña en claro de la página de autenticación de "Declaraciones anuales de personas físicas" (figura 5.20a).



(a) (b)

Figura 5.20: (a) Sitio de "Declaraciones anuales de personas físicas" desde Google Chrome en el dispositivo Motorola; (b)Captura de credenciales de "Declaraciones anuales de personas físicas"

5.5.7. Ataques a la API de navegación (Webview)

Se desarrolló una aplicación que utiliza Webview para evaluar su vulnerabilidad ante este ataque de husmeo, en vista de que, como se mencionó en el Capítulo 4, es la API que se ocupa para desplegar contenido web en las aplicaciones y ocupa los motores de renderizado del sistema [52] [24] [54]. La metodología del ataque es la misma que la empleada en la subsección 5.5.2 para los navegadores, se utilizaron los mismos escenarios, los mismos dispostivos y sitios, incluídos los del IPN, Liverpool y Sears.

La aplicación fue llamada $Navegación X$, donde X representa la versión de Android; por ejemplo, $Navegación 4 4$ indica que esta aplicación está hecha principalmente para ejecutarse en la versión 4.4, y de manera similar, $Navegación 6$ significa que es para la versión 6.0, como el de la figura 5.21.

Esta aplicación, está integrada por una actividad que se puede visualizar en la figura 5.21 y tiene habilitados los permisos de `INTERNET`, `WRITE_EXTERNAL_EXTORAGE` y `ACCESS_NETWORK_STATE`. También, se habilitó Javascript porque algunos de los sitios visitados lo requieren para su óptimo funcionamiento. La utilización de esta aplicación es similar al de un navegador móvil. Las APIs

ocupadas para esta aplicación varió de acuerdo al dispositivo de forma que fuera destinado a una versión en específico. Por ejemplo, para el dispositivo con la versión 4.3, la aplicación se hizo para la API 18, mientras que para la versión 6.0, se programó para la API 23 (ver Tabla 2.1).

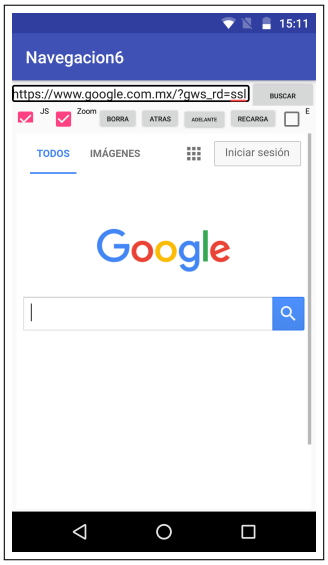


Figura 5.21: Aplicación que utiliza WebView para desplegar contenido web

Resultados del ataque a la aplicación

Se obtuvieron resultados similares de los navegadores preinstalados. De la información obtenida, se logró recuperar credenciales de autenticación.

- **Dispositivo con la versión 4.3.** La aplicación en este dispositivo resultó el más afectado, ya que en todos los sitios, hubo captura de credenciales de autenticación, con excepción de Caliente y Netflix.
- **Dispositivo con la versión 4.4.** En el dispositivo Dell, hubo captura de credenciales en Gmail, Facebook, Uptodown, Twitter, Disqus y Cinvestav.
- **Dispositivo con la versión 5.1.** En el dispositivo Samsung fue uno de los que tuvo mayor número de sitios en HTTP, ya que únicamente no hubo captura de información en Twitter, Netflix y Caliente.
- **Dispositivo con la versión 6.0.** De manera similar al teléfono inteligente con Lollipop, en el caso de la aplicación en la versión 6.0 no se obtuvieron credenciales solamente de Twitter, Netflix, Caliente.

Los resultados del ataque se muestran en las gráficas de la figura 5.22.

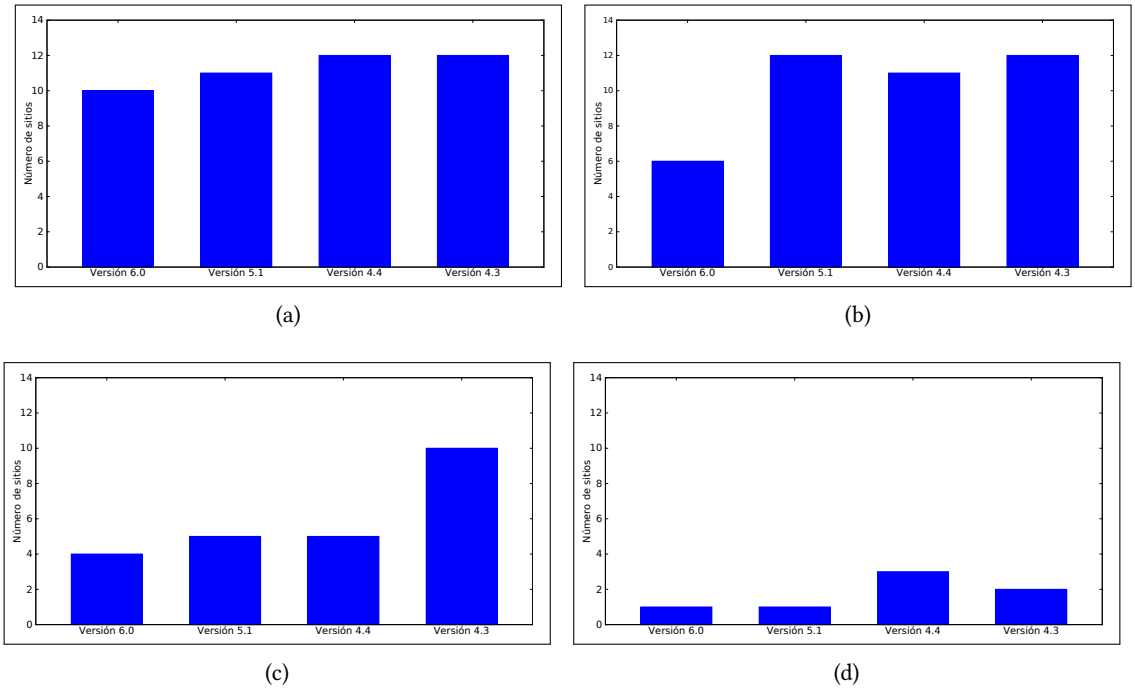


Figura 5.22: Número de sitios afectados por el ataque de husmeo a la aplicación con Webview: (a) Sitios que se degradaron a HTTP desde la aplicación que utiliza la API de navegación desde el ingreso de toda la dirección del sitio, ej. `www.sitio.com`; (b) Sitios que se degradaron a HTTP desde la aplicación que utiliza la API de navegación desde Google versión HTTP; (c) Sitios que se degradaron a HTTP desde la aplicación que utiliza la API de navegación desde el borrado de la 's' en el texto `https`; (d) Sitios que se degradaron a HTTP desde la aplicación que utiliza la API de navegación desde Google versión HTTPS

5.5.8. Prueba en la aplicación $NavegacionX$ de Sitios con HTTP con secciones en HTTPS. IPN, Liverpool y Sears.

Durante el ataque con el PA malicioso en la aplicación $NavegacionX$ con estos sitios, el correo electrónico del IPN siempre se degradó a HTTP en todos los dispositivos, de modo que se lograron capturar credenciales de autenticación. Para el sitio de Liverpool, la aplicación en los cuatro dispositivos se llegó hasta la fase donde se ingresan los datos relacionados al pago como la tarjeta de crédito, los cuáles se lograron capturar. Sin embargo, en el caso de Sears, no sucedió lo mismo, porque en el Motorola después de ingresar los datos personales del usuario, apareció un aviso que decía 'página web no disponible' que no dejó continuar con la fase en la que se capturaba la tarjeta de crédito/débito. En el dispositivo Sony y Dell de manera similar sucedió lo mismo y en Samsung

no se pudo ni siquiera ingresar los datos del usuario.

5.5.9. Pruebas en la aplicación $NavegacionX$ con el SAT y el CONACyT

De la misma manera que ocurrió con los navegadores preinstalados, en la aplicación con Webview también se realizaron pruebas con los sitios del CONACyT y el SAT ante este ataque de husmeo. Los resultados fueron los mismos que con los navegadores, en todos los sitios probados de ambas dependencias, la captura de credenciales fue exitosa.

5.5.10. Pruebas del nivel de seguridad de la aplicación $NavegacionX$ en el protocolo TLS/SSL

Del mismo modo que sucedió con la prueba realizada en la Sección 5.4, también se realizó una evaluación de la seguridad del protocolo TLS/SSL y una verficiación de los conjuntos de cifrados de esta aplicación de los cuatro dispositivos con Qualys SSL Labs.

VERSIÓN DEL DISPOSITIVO	VERSIÓN MÁXIMA DE TLS QUE SOPORTA	VULNERABLE A LOGJAM	VULNERABLE A FREAK	VULNERABLE A POODLE
Marshmallow	1.2	No	No	No
Kitkat	1.2 y SSL 3	Sí	Sí	Sí
Jelly-Bean	1.1 y SSL 3	Sí	Sí	Sí
Lollipop	1.2	No	No	No

Tabla 5.4: Resultados de la evaluación de la aplicación de Webview con Qualys SSL Labs, que muestra los navegadores que son vulnerables a los ataques Logjam, Freak y Poodle, así como la máxima versión de TLS/SSL que soportan.

En la tabla 5.4 los resultados fueron similiares a los de la Tabla 5.1, particularmente con los navegadores de fábrica. De la misma forma que los navegadores de fábrica de los dispositivos con Jelly Bean y Kitkat, la aplicación en estos dos dispositivos es vulnerable ante los ataques Logjam, Freak y Poodle, así como también tiene soporte para el inseguro SSL3.

Conjuntos de cifrado (*cipher suites*) soportados por la aplicación con Webview

La aplicación en el dispositivo con Jelly-Bean tiene los mismos conjuntos criptográficos inseguros y débiles que el Google Chrome de Fábrica de este dispositivo y tampoco tiene soporte para engrapado OCSP.

La aplicación del dispositivo con Kitkat tienen los mismos conjuntos de cifrado inseguros y débiles que los navegadores Android AOSP y el Google Chrome de fábrica del mismo y también, no

soporta el grapado OCSP.

De la misma manera, la aplicación en el dispositivo con Lollipop tiene los mismos conjuntos de Cifrado que el que poseé su versión de fábrica de Google Chrome. Y finalmente, en el dispositivo con la versión Marshmallow, esta aplicación con Webview resultó que también tiene los mismos conjuntos de cifrado de Google Chrome de fábrica de este dispositivo.

5.5.11. Problemas con Caliente y Netflix

En el caso de Caliente y Netflix, en los escenarios 1,3,4 del primer ataque, así como en los Escenarios 1 y 3 del ataque a la aplicación, sí hubo una degradación a HTTP, pero en el caso de Netflix no se visualizaba algo más allá de una página en negro. En el Sitio de Caliente, sí se visualizaba la página principal, pero al momento de iniciar sesión, la conexión se volvía HTTPS. No obstante, en ese lapso, con wireshark se capturaron los paquetes de petición y respuesta HTTP, mas no datos sensibles como credenciales de autenticación.

5.5.12. Ataque con el PA Malicioso a un sitio Wordpress

Wordpress es uno de los gestores de contenidos de internet de código abierto más utilizadas a nivel mundial. Por esta razón, se realizó una prueba de la seguridad del *front-end* de WordPress ante este ataque de PA, utilizando los navegadores de los dispositivos y la aplicación con Webview. Con esta finalidad, se instaló y configuró un sitio con el protocolo SSL/TLS.

Composición del sitio

El sitio está integrado por dos componentes principales; el primero es el sitio web que son un conjunto de páginas web públicas que cualquiera puede ver. El segundo es el portal web; que es privado y solamente un administrador y usuarios autorizados por él tienen acceso al portal. Para configurar el sitio se utilizó como servidor web Apache, MySQL como gestor de base de datos y para el manejo de SSL, se empleó un certificado autofirmado y el módulo de SSL que ofrece Apache.

Actividad del usuario administrador

Como el administrador es el único que puede acceder al portal y es el encargado de administrar el sitio, para identificarse debe de autenticarse con un usuario y contraseña desde la interfaz que muestra la figura 5.23 desde un navegador como los que tiene preinstalados Android. Una vez que está autenticado, puede hacer, entre otras acciones, generar nuevas páginas web, llamadas entradas (*post*), dar de alta a otros usuarios y autorizar o borrar comentarios.

Figura 5.23: Portal de ingreso como administrador al sitio de Wordpress desde Google Chrome en un dispositivo con Android 6.0.

Ataque desde los navegadores y la aplicación con Webview

Bajo el contexto del ataque con el PA ilegítimo se monitoreó la comunicación entre el administrador y el servidor del sitio, se verificó si es posible degradar la comunicación HTTPS. El ataque se hizo bajo los siguientes escenarios parecidos a los de los ataques anteriores, utilizando los mismos navegadores de la Tabla 5.2 y la aplicación con Webview de los diferentes dispositivos.

- **Escenario 1.** Ingresar la dirección del sitio en la barra de navegación sin el texto “https" o “http".
- **Escenario 2.** Ingresar la dirección del sitio con el texto “https".
- **Escenario 3.** Ingresar al sitio en HTTPS, y cuando esté cargada la página, borrar la letra ‘s’ de “https".

Durante la ejecución del ataque, en el marco de los Escenarios 1 y 3 se obtuvo información en claro en todos los dispositivos tanto en los navegadores como en la aplicación. Los datos sensibles como nombres de usuario y contraseña se pudieron capturar en claro en todos los dispositivos, tal y como lo muestra la figura 5.24. Asismismo, entre otra información obtenida, destaca todas las actividades que realiza el usuario en el portal después de haberse autenticado, como por ejemplo la aprobación/desaprobación de comentarios, y la captura de entradas en la página.

Wordpress tiene una característica que se llama autoguardado, que permite guardar automáticamente los cambios que el administrador sin necesidad de hacerlo de manera manual. Sin embargo, esta funcionalidad puede ser aprovechada en el ataque de husmeo, porque cuando esta operación se ejecuta sobre HTTP, todas las acciónes que haya realizado podrán ser interceptadas por SSLStrip. De esta manera, si el administrador decide eliminar una entrada que esté escribiendo antes de guardarla o publicarla, aún así se podrá interceptar a causa del autoguardado.

Figura 5.24: Captura del usuario y contraseña del administrador del sitio Wordpress, el usuario está resaltado con amarillo y está delante del texto “log=” y la contraseña se ubica delante de texto “pwd=”.

Mientras tanto, bajo el Escenario 2, no fue posible la degradación a HTTP. Sin embargo, una observación a destacar en este Escenario, es que cuando se suben fotos al sitio, la transferencia de subida se realiza sobre HTTP. Por lo tanto, con el uso de herramientas como Driftnet es posible interceptar tales imágenes, incluso si el administrador las elimina antes de haberlas publicado en alguna entrada, por la característica del autoguardado.

5.5.13. Ataques a las aplicaciones de los fabricantes

Con el mismo esquema de ataque con el PA ilegítimo, se pusieron a prueba algunas de las aplicaciones preinstaladas de los fabricantes en los cuatro dispositivos.

Aplicaciones de Motorola

Durante la ejecución de este ataque, las aplicaciones del dispositivo con la versión 6, generaron mucha información en SSL/TLS que no pudo ser degradado a HTTP. Sin embargo, hubo muchos paquetes con información en claro, como por ejemplo, los *user-agent*, así como tokens de autenticación y cookies. De las aplicaciones examinadas, en las que se recuperó más información fueron las de Ayuda, Clima y Play Música.

En la aplicación Ayuda, se pudo recuperar gran parte de la información que se transmitía a través de la red, principalmente la que está en las actividades. Asimismo, se recuperaron paquetes de petición y respuesta HTTP, dentro de los que se encuentran *user-agents*, cookies, e información del dispositivo, como la versión de Android que tiene, el modelo del fabricante y el motor de renderizado que utiliza. Además, esta aplicación permite al usuario ingresar comentarios (figura 5.25a), los cuáles fueron capturados con el PA, tal y como lo muestra la figura 5.25b.

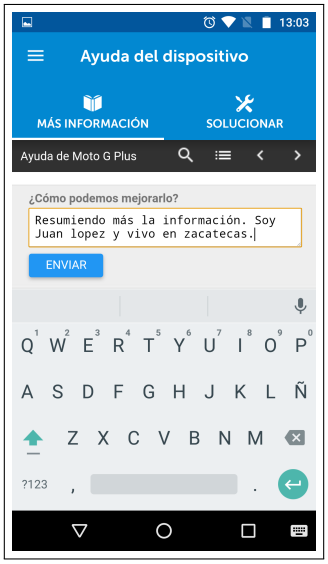
Dentro de la información de la aplicación Clima, destacan los paquetes de petición y respuesta HTTP; que contiene datos como cookies e información del dispositivo, así como tokens de autenticación. También, se encontraron imágenes codificadas en base64, las cuáles son las que forman parte de la aplicación. De la misma manera, se identificaron los nombres de las ciudades que el usuario seleccionó para obtener la información relativa al clima de las mismas, que en este caso fueron Cd. de México, Zacatecas y Cuernavaca, como la que se muestra en la figura 5.26. Con esto, se tiene una idea de cuáles son los lugares donde ha estado el usuario.

En la aplicación Play Música, también se encontraron *user-agents*, con información parecida a las aplicaciones anteriores, así como enlaces url a las páginas web de esta aplicación, principalmente relacionadas a ayuda de cómo usar la aplicación. Del mismo modo, se hallaron imágenes codificadas en base64, pero no resultaron nada más que íconos de la aplicación.

Durante la examinación de las aplicaciónes Fotos, Cámara y Radio FM, solamente se recuperaron las peticiones/respuesta HTTP, que incluyen los *user-agents*. Mientas tanto, las aplicaciones en las que no se logró recuperar información importante en claro fueron Moto, Mensajes, Play Películas y Contactos ya que éstos únicamente generaron información en HTTPS que no pudieron ser degradados, así como paquetes del protocolo ARP y conexiones DNS.

Aplicaciones de Dell

De manera similar al dispositivo Motorola, muchas de las aplicaciones de la tableta con la versión 4.4 generan información en HTTPS que no pudo ser degradada a HTTP. Sin embargo, en algunas de ellas se consiguió información en claro. Por ejemplo, en la aplicación de Amazon se obtuvieron paquetes de petición HTTP que contienen, además de información general del dispositivo, *user-agents*, cookies y token de sesión, así como lo muestra la figura 5.27.
Otra aplicación es Amazon Audible, de la que también se recuperaron los paquetes de petición/respuesta HTTP, donde venía el *user-agent* . Camcard es otra aplicación de la que se pudo recuperar



(a)

```
2017-05-01 13:03:27,211 POST Data (help.motorola.com):
topicid=jcb07012013835&language=es-us&menu-id=dh4&application=device-
help&carrier=global&country=es-us&model=3116&os=60&referrer=http%3A%2F%2Fhelp.motorola.com
%2Fhc%2F3116%2F60%2Fglobal%2Fes-us%2Findex.html%3Ft%3Djcb07012013835&topictitle=Usar
+comandos+de+voz+y+buscar&comments=Resumiendo+m%C3%A1s+la+informaci%C3%B3n.+Soy+Juan+lopez
+y+vivo+en+zacatecas.
2017-05-01 13:11:24,015 POST Data (help.motorola.com):
topicid=jcb0425131228&language=es-us&menu-id=dh4&application=device-
help&carrier=global&country=es-us&model=3116&os=60&referrer=http%3A%2F%2Fhelp.motorola.com
%2Fhc%2F3116%2F60%2Fglobal%2Fes-us%2Findex.html%3Ft%3Djcb0425131228&topictitle=++++++
+Cambiar+entre+aplicaciones+y+pesta%C3%B1as+de+Chrome++++++&ishelpful=1&comments=
2017-05-01 13:12:52,509 POST Data (help.motorola.com):
```

(b)

Figura 5.25: (a) Comentarios en la aplicación “Ayuda"de Motorola; (b) Captura de comentarios de la aplicación “Ayuda".

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1"><title>
.

El tiempo actual para Zacatecas - Pron&#243;stico de AccuWeather para Zacatecas M&#233;xico

        (ES)

</title><link href="http://www.googletagservices.com" rel="preconnect" crossorigin="" /><link href="http://securepubads.g.doubleclick.net" rel="preconnect'
crossorigin="" /><link href="http://www.googletagmanager.com" rel="preconnect" crossorigin="" />
    <script type="text/javascript">

        var TILE_SIZES = {
            AD_TOP: [[320, 50]],
            AD_MID: [[320,50]],
            AD_BOTTOM: [[300, 250]],
            AD_OVERLAY: [[300,50]]
        };
```

Figura 5.26: Información de la aplicación Clima, relacionada al clima de la ciudad que el usuario seleccionó

Figura 5.27: Información en claro de la aplicación Amazon

información como imágenes de la misma aplicación codificada en base 64, pero no imágenes del usuario. En el resto de las aplicaciones solamente se generó información en HTTPS, consultas a servidores DNS y paquetes del protocolo ARP.

Aplicaciones de Sony

En el dispositivo con Jelly Bean, las aplicaciones generalmente producen información en HTTPS. Por otro lado, algunas aplicaciones generaron información en claro, frecuentemente, mucha de esta información fueron solamente paquetes con petición y respuesta HTTP, que incluye *user-agents*, como el caso de Facebook. En cambio, las aplicaciones en las que no se pudo recuperar datos fueron Movie Studio y Notas.

La aplicación de Música también muestra una tupla de idiomas, posiblemente los que la aplicación puede soportar. De la misma manera, hace conexión con el sitio de Amazon, porque el *user-agent* muestra una etiqueta llamada `X-Amz-Cf-Id`, la que, de acuerdo a la documentación de Amazon [62], es un identificador para las peticiones para establecer una conexión al sitio de esta compañía.

La aplicación Xperia Privilege transmite el nombre del paquete de la aplicación. Además, se obtuvieron algunas imágenes, principalmente íconos. De la misma manera, Xperia Lounge, transmitió información como el nombre de la compañía telefónica (Telcel), así como algunos íconos. Mientras tanto, la aplicación Soporte, además de la petición/respuesta HTTP, se obtuvieron imágenes de la misma aplicación. En Sony select, se capturaron íconos y los paquetes de la petición HTTP.

Aplicaciones de Samsung

En las aplicaciones de Samsung (versión 5.1) fueron pocas aplicaciones donde se recuperó información en claro. Por ejemplo, las aplicaciones Mis Archivos, Cámara, Correo electrónico, Grabadora de Voz, Radio y Contactos solamente se obtuvo información en HTTPS, ARP o DNS.

En las aplicaciones Calendario, Notas y Mensajes la información en claro que se pudo recuperar fueron únicamente las peticiones HTTP, dentro de las que incluían los *user-agent*.

La aplicación Galaxy Apps es una tienda de aplicaciones como Google Play, que enseña al usuario información relacionada a algunas aplicaciones que puede descargar por medio de íconos de la aplicación y el nombre de ésta. La mayor parte de la información generada fue degradada a HTTP. De la información recuperada, se obtuvieron los paquetes de petición/respuesta HTTP, en las que incluyen *user-agents* y cookies, las imágenes de la aplicación (íconos) fueron capturadas con Driftnet.

Habría que destacar que cuando se descargó una aplicación desde esta tienda, los datos viajaban en claro durante la descarga y de la información interceptada, destaca el nombre de la aplicación, algunos datos del autor de la misma aplicación, como su nombre y correo electrónico.

Capítulo

6

Conclusiones y trabajo a futuro

"Sólo podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer"

Alan Turing

En este trabajo de tesis se realizó un análisis y una evaluación de la seguridad de algunos mecanismos de los dispositivos Android de algunas de las versiones más utilizadas al momento del desarrollo de este trabajo, cómo funcionan y cuáles son sus fallos.

Para eso, se abordó un enfoque de la recuperación de información eliminada y se midió el consumo de recursos de las aplicaciones preinstaladas exclusivas de los fabricantes. También, se evaluaron los servicios del bluetooth y WiFi, haciendo en éste último, ataques de husmeo con un Punto de Acceso Ilegítimo y degradando el tráfico de red que se genera en HTTPS a HTTP a los navegadores preinstalados, que utilizan los motores de renderizado del sistema, y a las aplicaciones preinstaladas de los fabricantes. De la misma manera, se creó una aplicación que utiliza Webview para determinar qué tan vulnerable es esta API ante este ataque de husmeo.

Resultados obtenidos

Es posible recuperar parte de la información eliminada en el restablecimiento de fábrica y también en el borrado normal. Aunque, la cantidad de información rescatada es mayor desde el borrado normal. De esta manera, en ambos escenarios de borrado, se recuperaron imágenes, números te-

lefónicos de llamadas realizadas, nombres de contactos y metadatos de algunos documentos.

La cantidad de datos recuperados varió de acuerdo al dispositivo y del tipo de archivo. En el borrado normal, los dispositivos con las versiones 4.3 y 5.1 fueron los que tuvieron mayor número de imágenes recuperadas y metadatos de documentos. Mientras que en todos los dispositivos utilizados se recuperaron nombres de los sitios visitados desde el historial de navegación y números telefónicos.

Entre tanto, en el restablecimiento de fábrica, los dispositivos con las versiones 4.3, 4.4 y 5.1 se recuperó más información a comparación del dispositivo con la versión 6.0, el cuál fue el que presentó la menor cantidad de datos recuperados en ambos escenarios de eliminación y esto fue más notorio en el restablecimiento de fábrica.

Si se eliminan datos enviados o recibidos utilizando el servicio del bluetooth, es posible conseguir por lo menos los metadatos originales de estos archivos como el nombre, la fecha de creación, de modificación, el tipo de archivo y la dirección MAC de los dispositivos vinculados.

Los navegadores preinstalados son más propensos a la degradación HTTP que los navegadores Google Chrome actualizados, por lo tanto un atacante con un punto de acceso ilegítimo es capaz de obtener información sensible que se generen en estas aplicaciones como credenciales de autenticación. Esto es más evidente en los dispositivos con las versiones más antiguas como Jelly Bean y Kitkat, al presentar la mayor cantidad de sitios que fueron degradados a HTTP. Aunque esto también está presente en los navegadores de los teléfonos inteligentes de las versiones más recientes como Lollipop y Marshmallow.

Sin embargo, los navegadores Google Chrome actualizados también tienen sus fallos particulares ante este tipo de ataques, porque fueron los que tuvieron más captura de datos en los sitios de la subsección 5.5.5 del capítulo 5 y además, su pantalla de inicio “artículos sugeridos” es HTTP, al menos en el momento de realizar este ataque.

De acuerdo con los resultados obtenidos en base a las figuras 5.11 y 5.22 del capítulo 5, la API Webview es más vulnerable ante el ataque de husmeo con un PA ilegítimo que las aplicaciones de los navegadores en los dispositivos 4.3 a 6.0. Esto significa que las aplicaciones que utilizan esta API también son vulnerables.

Un sitio Wordpress que tiene habilitado el protocolo SSL/TLS también es vulnerable ante el ataque de husmeo, porque se puede degradar a HTTP al menos en todos los navegadores probados y

también si se utiliza desde la aplicación con Webview.

Algunas aplicaciones preinstaladas por los fabricantes producen tráfico HTTP que puede ser interceptado por el punto de acceso ilegítimo. Dentro de esta información viaja información como *user-agents*, cookies y tokens. Dentro de las aplicaciones que generaron más tráfico en HTTP se encuentran Clima y Ayuda de Motorola y Galaxy Apps de Samsung.

Trabajo a futuro

El presente trabajo de tesis implica el desarrollo de los siguientes puntos a tratar como trabajo a futuro:

- Hacer el análisis de seguridad ante el ataque de husmeo con el PA ilegítimo con la versión actualizada del navegador propio de Samsung y de la misma manera, realizar lo mismo con otros navegadores propios de otros fabricantes de dispositivos Android.
- El estudio del ataque de husmeo se puede ampliar con el mecanismo de redes VPN que ofrece Android para determinar qué tan vulnerable es ante este ataque.
- Llevar a cabo ataques de inyección, como el efectuado por Choi et al. [5], en las aplicaciones que transmitan información vía HTTP, sobre todo en la aplicación Galaxy Apps de Samsung y Clima y Ayuda de Motorola.
- Encontrar nuevas formas de degradar la comunicación HTTPS a HTTP.
- Hacer un estudio de qué información eliminada se puede recuperar desde la memoria RAM realizando eliminación de información desde los enfoques del borrado normal y del restablecimiento de fábrica.
- Los proveedores de contenido son una superficie de ataque, por lo tanto se plantea atacar estos componentes de las aplicaciones preinstaladas de algunos fabricantes de forma remota desde el PA ilegítimo.

Glosario

adb — Acrónimo de *Android Debug Bridge*, es una herramienta que se compone de un demonio llamado adbd, que está en el dispositivo, un servidor llamado adb server en la estación de trabajo (típicamente una computadora personal) y una línea de comandos adb, con la que se puede interactuar con el dispositivo a través de una terminal.

Aplicación web — Aplicación que está hecha para usar tecnología web como HTML, CSS y javascript

Autofirmado — Certificado digital que no está firmado por una autoridad certicadora, sino por propia entidad que hace uso del certificado.

Autoridad Certificadora (CA) — Se encarga de expedir certificados de llava pública a cada entidad, confirmando la identidad del propietario del certificado.

BLKDISCARD — Comando utilizado para quitar sectores de un dispositivo de almacenamiento como una memoria flash. Por defecto, desechará todos los bloques en el dispositivo.

BLKSECDISCARD — Comando similar a BLKDISCARD, con la diferencia de que proporciona un borrado seguro.

Bootloader Cargador de arranque en inglés, es un mecanismo que proporciona el código de arranque del procesador. Es responsable de encontrar y cargar la imágen de arranque, inicializar la RAM, cargar el kernel, manejar las actualizaciones del *firmware* y arrancar en el modo *recovery*.

Bootloader Boot Control Mecanismo utilizado del *Recovery* para comunicarse con el *Bootloader*.

Bricked Dispositivo móvil que ya no funciona debido a una corrupción en el *firmware* o un problema en el hardware producido por una mala configuración.

Buffer overflow Término en inglés de "Desbordamiento de búfer", es una situación anómala que se da cuando un programa que está escribiendo datos a un arreglo, excede su límite y sobreescribe sus posiciones de memoria adyacentes.

Certificado digital Estructura de datos que asocian las llaves públicas con una identidad. El certificado es verificado por un tercero de confianza llamada Autoridad Certificadora que avala que la información contenida en el certificado es verídica, proporcionando así, el servicio de autenticación. La gran mayoría de los certificados están bajo el estándar X.509.

Conjunto de cifrado (*Cipher Suite*) Es una composición de primitivas criptográficas utilizadas para negociar las configuraciones de seguridad para una conexión de red que utiliza los protocolos SSL y TLS.

ElGamal Algoritmo de criptografía de llave pública que basa su seguridad en el problema del logaritmo discreto.

Exploit	También llamada *Explotación*, es un programa o un *script* desarrollado para tomar ventaja de una vulnerabilidad o un *bug* y así el atacante podrá ganar un control total o parcial de un sistema computacional.
Explotación del día cero	Del inglés *zero-day exploit*, es un programa o *script* que toma ventaja de una vulnerabilidad que aún no ha sido publicada.
Freak	*Exploit* que permite a un atacante interceptar las comunicaciones HTTPS y obliga al cliente y al servidor a utilizar cifrados débiles. De esta manera, el atacante podrá obtener datos sensibles.
HTTPS	Implementación del protocolo SSL/TLS en HTTP para ofrecer los servicios de confidencialidad e integridad en las aplicaciones web.
JNI	Acrónimo de *Java Native Interface*, framework que da facultad a código Java de hacer llamadas y ser llamado por bibliotecas escritas en lenguajes como C, C++ y ensamblador.
Logjam	Vulnerabilidad que permite a un atacante de intruso de en medio degradar conexiones vulnerables TLS a criptografía de 512 bits, la cuál es insegura.
Montar	Acción de hacer accesible un dispositivo de almacenamiento externo en un sistema de archivos.
Poodle	Acrónimo de *Padding Oracle On Downgraded Legacy Encryption*, es un *exploit* que toma ventaja de las conexiones SSL 3.0. El atacante sólo necesitará a lo más 256 peticiones SSL 3.0 para revelar un byte de los mensajes cifrados.

Recovery	Consola de recuperación que contiene código que arranca en un entorno Linux con herramientas para instalar, recuperar o reparar el sistema operativo Android.
Root	En sistemas Linux, es un usuario que poseé privilegios que le dan la facultad de realizar casi cualquier operación en el sistema operativo. También se le conoce con el nombre de superusuario.
RSA	Acrónimo de Rivest, Shamir y Adleman. Es un algoritmo criptográfico de llave pública que basa su seguridad en el problema de factorización de enteros.
Spear phishing	Ataque que a través de correo electrónico o mensaje SMS obtiene acceso no autorizado a datos confidenciales, centrado en un grupo en específico.
SSL	Acrónimo de *Secure Socket Layer*. Es un protocolo criptográfico que hace uso de TCP, creado para proporcionar comunicaciones seguras sobre una red de computadoras, brindando así los servicios de confidencialidad e integridad.
SUID	Acrónimo de *Set owner User ID*. Conjunto de permisos otorgados a un archivo indicanto que el usuario que ejecute el archivo tendrá los mismos permisos que el usuario que lo creó.
TLS	Acrónimo de *Transport Layer Security*, es el sucesor del protocolo SSL.
udevd	Demonio de Linux que escucha los eventos del kernel.
vold	Demonio que Android utiliza para sustituir al demonio udevd.

Bibliografía

[1] Google and Open Handset Alliance. Bluetooth. URL: https://developer.android.com/about/dashboards/index.html (consultado el 28-08-2017).

[2] Elenkov, N. Revisting android disk encryption, 2014. URL: https://nelenkov.blogspot.mx/2014/10/revisiting-android-disk-encryption.html (consultado el 09-02-2017).

[3] Lanier, Z. Mulliner, C. Ridley, S.A. Drake, J.J., Fora, P.O. and Wicherski, G. *Android Hacker's Handbook*. Wiley, USA, first edition, 2014. ISBN: 978-1-118-60864-7.

[4] Lazaro-Dominguez, F. *Investigación forense de dispositivos móviles Android*. Ra-Ma ediciones de la U, Mexico, first edition, 2015. ISBN: 978-958-762-429-8.

[5] Eom, J.H. Park, M.V., Choi, Y.H. and Chung, T.M. Dangerous Wi-Fi access point: attacks to benign smartphone applications. *Personal and Ubiquitous Computing*, 18(6):1373–1386, 2013. Springer International Publishing.

[6] Robinson, G. and Weir,R.S. Understanding Android Security. *Global Security, Safety and Sustainability: Tomorrow's Challenges of Cyber Security*, 4(CCIS 534):189–199, 2015. Springer.

[7] Anónimo. Vault 7: CIA Hacking Tools Revealed, 2017. URL:https://wikileaks.org/ciav7p1/ (consultado el 25-03-2017).

[8] Simon, L. and Anderson, R. Security Analysis of Android Factory Resets. In *Proceedings of 4th Workshop on Mobile Security Technologies (MoST)*, San Jose, USA, 2015. URL: https://www.cl.cam.ac.uk/~rja14/Papers/fr_most15.pdf.

[9] Pegueroles, J. Jódar, E. and Vera del Campo, J. Estudio práctico de mecanismos de seguridad en dispositivos Android. In *Reunión Española sobre Criptología y Seguridad de la Información* , pages 259–263, Alicante, España, 2014. ISBN: 978-84-9717-323-0.

[10] Marinakis, G. Ntantogian, C., Apostolopoulos, D. and Xenakis, C. Evaluating the privacy of Android mobile applications under forensic analysis. *Elsevier Computers Security*, 42:66–76, 2014. Elsevier.

[11] Ntantogian, C. Apostolopoulos, D., Marinakis, G. and Xenakis, C. Discovering authentication credentials in volatile memory of android mobile devices. *IFIP Advances in information and Communication Technology*, 399:178–185, 2013. Springer Berlin Heidelberg.

[12] Skorobogatov, S. The bumpy road towards iPhone 5c NAND mirroring, 2016. URL: `https://arxiv.org/pdf/1609.04327v1.pdf`.

[13] Choi, C. Choi, J., Sung, W. and Kim, P. Personal information leakage detection method using the inference-based access control model on the android platform. *Pervasive and Mobile Computing*, 24:138–149, 2015. Elsevier.

[14] Paar, C. and Pelzl, J. *Understanding Cryptography*. Springer, USA, first edition, 2010. ISBN: 978-3-642-44649-8.

[15] Daemen, J. and Rijmen, V. *The Design of Rjndael: AES - The Advanced Encryption Standard*. Springer, Belgium, first edition, 2002. ISBN: 9783642076466.

[16] Ristic, I. SSL and TLS deployment best practices, 2017. URL: `https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices` (consultado el 06-08-2017).

[17] Vanstone, S.A. Menezes, A.J. and Oorschot, P.C.V. *Handbook of Applied Cryptography*. CRC Press, Inc., USA, first edition, 1996. ISBN: 9780849385230.

[18] Girones, J.T. *El gran libro de Android*. Alfaomega, Mexico, third edition, 2013. ISBN: 978-607-622-494-6.

[19] Dubey, A. and Misra, A. *Android Security Attacks and Defenses*. CRC Press, USA, first edition, 2013. ISBN: 978-1-4398-9647-1.

[20] Tamma, R. and Tindall, D. *Learning Android Forensics*. PACKT Publishing, UK, first edition, 2015. ISBN: 978-1-78217-457-8.

[21] Stallings, W. *Operating Systems, Internals and Design Principles*. Pearson, USA, eight edition, 2015. ISBN: 978-0-13-380591-8.

[22] Elenkov, N. *Android Security Internals. An In-Depth Guide to Android's security Architecture*. No Starch Press, Inc., USA, first edition, 2015. ISBN:978-1-59327-581-5.

[23] Android Open Source Project. Android interfaces and architecture. URL: https://source.android.com/devices/#Hardware-Abstraction-Layer (consultado el 16-04-2017).

[24] Yaghmour, K. *Embedded Android.* O'Reilly, USA, first edition, 2013. ISBN:978-1449308292.

[25] Levin, J. *Android Internals A Confectioner's Cookbook. Volume I: The Power User's View.* Technologeeks.com, USA, first edition, 2015. ISBN: 978-0-9910555-2-4.

[26] Google and Open Handset Alliance. Wifimanager. URL: https://developer.android.com/reference/android/net/wifi/WifiManager.html (consultado el 15-04-2017).

[27] Edge, J. Returning bluez to android, 2014. URL: https://lwn.net/Articles/597293/ (consultado el 11-04-2017).

[28] Google Inc. and Open Handset Alliance. Bluetooth. URL: https://developer.android.com/guide/topics/connectivity/bluetooth.html (consultado el 12-04-2017).

[29] Google Inc. and Open Handset Alliance. Intents y filtros de intents. URL: https://developer.android.com/guide/components/intents-filters.html?hl=es-419 (consultado el 09-04-2017).

[30] Google Inc. and Open Handset Alliance. Proveedores de contenido. URL: https://developer.android.com/guide/topics/providers/content-providers.html?hl=es-419 (consultado el 10-04-2017).

[31] Android Open Source Project. Security. URL: https://source.android.com/security/?hl=es (consultado el 24-05-2017).

[32] Google Inc. and Open Handset Alliance. Permisos del sistema. URL: https://developer.android.com/guide/topics/security/permissions.html?hl=es (consultado el 24-05-2017).

[33] Google Inc. and Open Handset Alliance. Permission. URL: https://developer.android.com/guide/topics/manifest/permission-element.html (consultado el 23-05-2017).

[34] Gurski, M. A. Privacy-enhanced mail (pem), 1995. URL: https://www.csee.umbc.edu/~woodcock/cmsc482/proj1/pem.html (consultado el 14-06-2017).

[35] Spada, F.E. Wei, M., Grupp, L. M. and Swanson, S. Reliably erasing data from flash-based solid state drives. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, pages 8–8, San Jose, California, 2011. USENIX Association. ISBN: 978-1-931971-82-9.

[36] Corsair. Usb flash wear-leveling and life span, 2007. URL: https://www.scribd.com/document/7010228/CORSAIR-USB-Flash-Wear-Leveling-and-Life-Span-FAQs (consultado el 25-05-2017).

[37] Denholm, T. Comparing secure nand erase methods. Technical report, 2016. URL: http://www.farelettronica.it/web/wp-content/uploads/2016/02/Comparing-Secure-NAND.pdf.

[38] Centro Criptológico Nacional. *Guía de Seguridad de las TIC (CCN-STIC-453, Seguridad de los dispositivos móviles: Android.* Gobierno de España, Ministerio de la Presidencia, España, 2013. URL: https://www.ccn-cert.cni.es/series-ccn-stic/guias-de-acceso-publico-ccn-stic/11-ccn-stic-453-seguridad-en-android/file.htm.

[39] Android Open Source Project. Full-disk encryption. URL: https://source.android.com/security/encryption/full-disk (consultado el 15-02-2017).

[40] Galindo, V. Trusted execution environment, millions of users have one, do you have yours? URL: https://poulpita.com/2014/02/18/trusted-execution-environment-do-you-have-yours/ (consultado el 13-06-2017).

[41] Coojimans, T. Secure Key Storage and Secure Computation in Android. Master's thesis, Radbound University Nijmegen, Nijmegen, the Netherlands, 2014. URL: http://www.ru.nl/publish/pages/769526/scriptie_tim_cooijmans.pdf.

[42] Sony Mobile Comunications Inc. Sony Common Criteria Security target for Xperia Devices. Technical report, oct 2016. URL: https://www.fmv.se/Global/Sony-CC-SecurityTarget_v1.1.pdf(331559)_TMP.pdf.

[43] Ltd Samsung Electronics Co. Samsung kernel cryptographic module fips 140-2 non-propietary security policy. Technical report, jul 2016. URL: http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp2674.pdf.

[44] Tamma, R. Bommisetty, S. and Mahalik, H. *Practical Mobile Forensics.* PACKT Publishing, UK, first edition, 2014. ISBN: 978-1-78328-831-1.

[45] Hossain, M.S. Hassan, S.S., Bibon, S.D. and Atiquzzaman, M. Security threats in bluetooth technology. *Computers & Security*, 2017. ISSN: 0167-4048.

[46] Armis Inc. The attack vector "blueborne" exposes almost every connected device. URL: https://www.armis.com/blueborne/#/technical (consultado el 14-09-2017).

[47] Seri, G. and Vishnepolsky, G. Blueborne. the dangers of bluetooth implementations: Unveiling zero day vulnerabilities and security flaws in modern bluetooth stacks. Technical report, 2017. URL: http://go.armis.com/hubfs/BlueBorne%20Technical%20White%20Paper-1.pdf?t=1505319664351.

[48] Gupta, A. *Learning Pentesting for Android Devices.* PACKT Publishing, UK, first edition, 2014. ISBN: 978-1-78328-898-4.

[49] Dixit, A. Verma, P. *Mobile Device Exploitation Cookbook.* PACKT Publishing, UK, first edition, 2016. ISBN: 978-1-78355-872-8.

[50] Frichot, C. Alcorn, W. and Orru, M. *The Browser Hacker's Handbook.* Wiley, USA, first edition, 2014. ISBN: 978-1-118-66209-0.

[51] Marlinspike, M. New tricks for defeating SSL in practice, 2009. URL: https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf (consultado el 02-07-2017).

[52] Kotipalli, S.R. and Imran, M.A. *Hacking Android.* PACKT Publishing, UK, first edition, 2016. ISBN: 978-1-78588-314-9.

[53] Chin, E. and Wagner, D. *Bifocals: Analyzing WebView Vulnerabilities in Android Applications*, pages 138–159. Springer International Publishing, Cham, 2014. ISBN: 978-3-319-05149-9.

[54] Daoyuan, W. and Chang, R. K. C. *Analyzing Android Browser Apps for file:// Vulnerabilities*, pages 345–363. Springer International Publishing, Cham, 2014. ISBN: 978-3-319-13257-0.

[55] Cuadros, A. Sun, S. and Beznosov, K. Android rooting: Methods, detection and evasion. 2015. ISBN: 978-1-4503-3819-6/15/10.

[56] Howarth, F. Is rooting your phone safe? the security risks of rooting devices, 2015. URL: https://insights.samsung.com/2015/10/12/is-rooting-your-phone-safe-the-security-risks-of-rooting-devices/ (consultado el 15-07-2017).

[57] Vaughan-Nichols, S.J. The dirty cow linux bug: A silly name for a serious problem, 2016. URL: http://www.zdnet.com/article/the-dirty-cow-linux-security-bug-moos/ (consultado el 18-07-2017).

[58] Rosenberg, D. Reflections on trusting trustzone - qsee trustzone kernel integer overflow vulnerability. 2014. ISBN:978-1-4503-2957-6.

[59] Tamma, R. Mahalik, H. and Bommisetty, S. *Practical Mobile Forensics*. PACKT Publishing, UK, second edition, 2016. ISBN: 978-1-78328-831-1.

[60] Internet Engineering Task Force (IETF). Transport Layer Security (TLS) Extensions: Extension definitions, 2011. URL: https://tools.ietf.org/html/rfc6066#section-8 (consultado el 07-08-2017).

[61] Amazon.com. Top sites in mexico, 2017. URL: http://www.alexa.com/topsites/countries/MX (consultado el 10-03-2017).

[62] Amazon web services. Request and response behavior for custom origins, 2017. URL: http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorCustomOrigin.html (consultado el 01-08-2017).

[63] Lookout. Pegasus for android, technical analysis and findings of chrysaor. Technical report, april 2017. URL: https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-android-technical-analysis.pdf.

[64] Metha, N. Bodzak, K. Chang, W. Cannings, R.,, Woloz, J. and Ruthven, M. An investigation of chrysaor malware on android, 2017. URL: https://android-developers.googleblog.com/2017/04/an-investigation-of-chrysaor-malware-on.html (consultado el 08-09-2017).

Apéndice

A

Pegasus, el malware espía en Android y iOS

A continuación se presenta de manera breve el funcionamiento de uno de los malware más recientes que permite a un atacante monitorear toda la actividad que un usuario realiza en su teléfono inteligente si utiliza Android o iOS, Pegasus, o también llamado Chrysaor. El presente apéndice sigue muy de cerca la documentación presentada por Lookout [63] y Google [64].

Chrysaor es un malware creado por la empresa israelí NSO, que en un principio fue creado para el sistema operativo iOS de Apple y después salió la versión para Android. En este apéndice solamente se hara énfasis en la versión de Android.

Para que Chrysaor pueda ser almacenado en el teléfono, previamente el usuario deberá haber accedido a un enlace desde un mensaje SMS enviado por el atacante, es decir, debió haber sido víctima de un *spear phishing*.

Este malware es capaz de filtrar datos de aplicaciones como Whatsapp, Skype, Facebook, Viber, Kakao, Twittwer, Gmail, la aplicación de correo electrónico de la AOSP, los navegadores Android AOSP y Google Chrome, también puede vigilar el audio a través del micrófono, realizar *keylogging*, hacer capturas de pantalla y deshabilitar las actualizaciones del sistema. De la misma manera, puede autodestruirse para no dejar rastro o si no logra sus objetivos.

A.1. Instalación

El malware esencialmente es una aplicación *com.network.android* que tiene sus variantes, y que permanece latente hasta que se reinicia el dispositivo. Durante el reinicio del sistema, se lanza un intent llamado `android.intent.action.BOOT_COMPLETED`, es aquí donde Pegasus entra en acción y obtiene un conjunto inicial de opciones de configuración.

Este conjunto de opciones lo obtiene realizando un análisis gramatical en una dirección URL para encontrar una cadena llamada "rU81PXbn". Para realizar esto, debió haber accedido primero al historial del navegador. Ya obtenido el conjunto de opciones, la aplicación borra la URL del historial del navegador. También, obtiene la configuración inicial desde los siguientes archivos, si están presentes en el dispositivo; `/data/myappinfo` y `/system/ttg`.

A.2. Acceso a root

Cuándo la aplicación es instalada, Chrysaor intenta obtener el acceso al usuario *root* del dispositivo con el uso del *exploit* framaroot. En caso de que este *exploit* no tenga éxito, entonces intentará utilizar el archivo binario del súperusuario en el directorio `/system/czk`.

A.3. Métodos de comunicación

Chrysaor se comunica con un servidor a través de SMS, del protocolo HTTP y del protocolo *Message Queue Telemetry Transport* (MQTT).

A.3.1. Comunicación sobre HTTP

Para conectarse al servidor por este protocolo, Chrysaor envía una petición HTTP desde la aplicación, que contiene dos campos; "SessionId1" y "SessionId2". El primer campo es un token cifrado con AES, usado por el servidor para generar respuestas cifradas e identificar el dispositivo, mientras que el segundo, es un arreglo de bytes generado de forma aleatoria, es la llave AES para cifrar datos en la respuesta HTTP.

Estos paquetes de petición HTTP, contienen otros campos cifrados con la llave de "SessionId2", los cuáles, cuando son descifrados, contienen archivos XML con información del dispositivo, como el modelo, la versión de Android, entre otros. También, pueden contener información filtrada del dispositivo, como por ejemplo la del calendario, que se muestra en la Figura A.1,

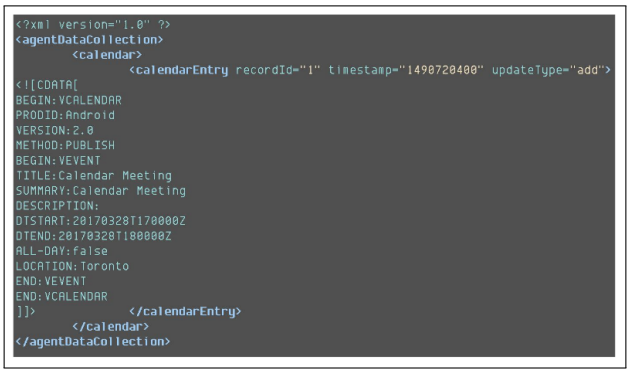


Figura A.1: Información filtrada de la aplicación Calendario. Fuente:

Mientras tanto, en la respuesta HTTP enviada desde el servidor, viajan archivos XML cifrados que contienen instrucciones que el servidor envía a la aplicación para que ejecute una acción en particular, como tomar una captura de pantalla, una foto con la cámara, envíar los registros de llamadas, entre otros.

A.3.2. SMS

Otra forma en la que la aplicación puede recibir instrucciones del servidor es a través de mensajes SMS, que aparentan ser códigos de autenticación de Google. Dentro de las instrucciones que envía el servidor, viene el número telefónico al que la aplicación debe enviar mensajes SMS, los cuáles incluyen un subconjunto de la información disponible en los paquetes de comunicación HTTP. Cabe aclarar que estos mensajes están ocultos.

A.3.3. *Message Queue Telemetry Transport* (MQTT)

Hay muchas restricciones para que el atacante pueda comunicarse a través de este medio, por ejemplo, MQTT debe estar configurado para ejecutarse en la red WiFi o en la red celular. Si Chrysaor no está explícitamente configurado para que no utilice MQTT en estos escenarios, entonces no ocurrirá la conexión MQTT. En caso de que sí esté permitido el uso de MQTT sobre la red, el proceso de recibo y envío de instrucciones es el mismo que en la comunicación por HTTP o por SMS.

A.4. Obtención de datos y funcionalidad de vigilancia

Como se mencionó al principio de este apéndice, Chrysaor es capaz de monitorear la actividad de aplicaciones de comunicación y mensajería. Para lograr esto, antes de lograr el acceso a *root* verifica

si existen las bases de datos de aplicaciones de mensajería. De ser esto así, adquiere privilegios de *root* y posteriormente modifica los permisos de las bases de datos haciéndolos de lectura y escritura para todos los usuarios, después, obtiene información como las conversaciones por chat y los correos electrónicos. Después de copiar esa información, vuelve a cambiar los permisos a como estaban antes de modificarlos.

A.4.1. Vigilancia de audio

Esta funcionalidad de Chrysaor es lanzada cuando una llamada es recibida desde un número especificado por el atacante, y cuando esto sucede, el adversario puede capturar el audio recibido desde el micrófono del dispositivo silenciosamente.

Sin embargo, esta característica puede ser activada únicamente bajo ciertas condiciones; que la pantalla esté bloqueada o apagada, que el reenvío de llamadas no esté activado, que no esté activada la aplicación de música, que no estén conectados los auriculares, que el estado de telefonía no esté inactivo, entre otras.

A.4.2. Capacidad para grabar la cámara

Chrysaor contiene la funcionalidad de capturar el contenido visual a través de capturas de pantalla o con la cámara frontal/trasera del usuario. La captura de pantalla fue implementada dos veces. La primera implementación consiste en revisar si existe el archivo binario `screencap` en `/system/bin`, si esto se cumple, Chrysaor hace una captura de pantalla y lo guarda en formato PNG a `/data/data/com.network.android/bqul4.dat`

En caso de que el dispositivo no tenga el binario `screencap`, en su lugar, Chrysaor utiliza el binario `take_screen_shot`, ubicado en el directorio `res/raw` y almacena de manera temporal la captura de pantalla en formato PNG a `/data/data/com.network.android/tss64.dat`. Finalmente, las imágenes que fueron capuradas exitosamente son comprimidas a formato JPG.

De manera similar, Chyrsaor tiene la habilidad de usar la cámara para tomar fotos, las cuáles son guardadas en formato PNG para después ser comprimidas en formato JPG.

A.4.3. *Keylogging*

Chrysaor puede registrar el contenido que el usuario ingresa en el teclado. Para realizar esto, Chrysaor escribe el archivo ELF `libk` en el directorio `/data/local/tmp/libuml.so`, donde es ejecutado e inyectado en el id del proceso del teclado antes de ser borrado. Por lo tanto, los caracteres de las

teclas presionadas son escritas a `/data/local/tmp/ktmu/ulmndd.tmp` antes de ser movidas a un archivo de estampa de tiempo en `/data/local/tmp/ktmu/finidk.<tiempo_actual>`.

A.5. Autodestrucción

Chrysaor es capaz de destruirse a sí mismo bajo alguna de las siguientes situaciones:

- El identificador SIM MCC es inválido. Esto es para prevenir de que sea ejecutado en dispositivos de prueba y emuladores que podrían no estar conectados a las redes celulares
- Existe un "archivo antídoto". Esto sucede si existe un arhivo en el directorio `/sdcard/MemosForNotes`.
- No se ha conectado al servidor en más de 60 días.
- Recibe un comando del servidor para borrarse a sí mismo.

Apéndice

B

Consumo de Recursos de Aplicaciones

A continuación se presentan los resultados de las mediciones de las aplicaciones mencionadas en la Sección 5.3 del Capítulo 5.

	CPU			Memoria (en MB)		
Nombre de la aplicación	Aplicación abierta	En el fondo	Aplicación cerrada	Aplicación abierta	En el fondo	Aplicación cerrada
Xperia Link	26.0 %	0.0 %	0.0 %	38.3	30.2	6
Películas	25.0 %	0.0 %	0.0 %	57.2	46.6	46.5
Facebook	25.0 %	0.80 %	1.0 %	75.0	69.0	42.9
Xperia Lounge	23.0 %	0.0 %	0.0 %	98.5	83.0	75.0
Soporte Sony	23.0 %	0.10 %	0.0 %	143.6	125.3	0.0
Twitter	16.0 %	0.0 %	1.1 %	55.0	22.6	0.0
Correo electrónico	16.0 %	0.0 %	0.0 %	55.0	22.6	0.0
Reloj	15.0 %	0.0 %	0.0 %	43.3	33.3	0.0
Música	13.0 %	0.10 %	0.0 %	51.0	48.0	0.0
Mensajes	13.0 %	0.10 %	0.0 %	51.0	48.0	0.0
Cámara	10.0 %	0.0 %	0.0 %	90.0	89.0	88.0
Sony Select	9.0 %	0.30 %	0.4 %	40.0	31.6	31.6
Xperia Privilege	7.0 %	0.30 %	0.0 %	118	121.1	113
Album	6.90 %	0.0 %	0.0 %	74.5	62.2	62.0

Tabla B.1: Medición del CPU y Memoria de aplicaciones del Sony Xperia SP (versión 4.3)

	CPU			Memoria (en MB)		
Nombre de la aplicación	Aplicación abierta	En el fondo	Aplicación cerrada	Aplicación abierta	En el fondo	Aplicación cerrada
Smart Manager	21.0 %	0.0 %	0.0 %	21.5	23.9	24.3
Mensajes	13.0 %	0.7 %	0.2 %	41.2	36.5	11.5
Calendario	13.0 %	0.0 %	0.0 %	17.2	16.1	7.1
Reloj	10.0 %	0.0 %	0.0 %	14.0	13.6	2.8
Notas	10.0 %	2.0 %	0.0 %	36.1	26.4	0.0
Mis archivos	9.8 %	0.0 %	0.0 %	27.6	21.8	0.0
Grabadora de voz	6.3 %	1.2 %	0.0 %	23.1	21.4	0.0
Contactos	3.1 %	3.0 %	0.3 %	34.0	29.6	8.6
Cámara	3.0 %	0.0 %	0.0 %	28.0	21.6	0.0
Radio	3.0 %	1.3 %	0.0 %	21.6	22.6	0.0
Correo electrónico	2.3 %	0.5 %	0.0 %	65.6	61.0	13.0

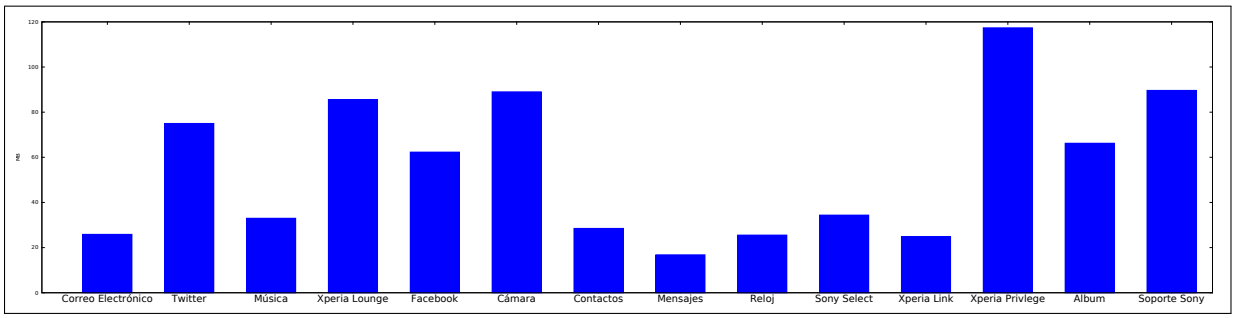
Tabla B.2: Medición del CPU y Memoria de aplicaciones del Samsung Galaxy Grand Prime (versión 5.1)

	CPU			Memoria (en MB)		
Nombre de la aplicación	Aplicación abierta	En el fondo	Aplicación cerrada	Aplicación abierta	En el fondo	Aplicación cerrada
Amazon Music	52.0 %	18.0 %	3.7 %	131	58.8	53.1
Amazon Audible	40.0 %	0.9 %	0.0 %	81.3	69.4	0.0
Polaris Xlsx	27.0 %	2.0 %	0.0 %	69.3	51.7	0.0
Skitch	27.0 %	0.0 %	0.0 %	90.6	55.8	0.0
Dell Cast	25.0 %	3.1 %	0.0 %	65.2	52.8	0.0
Facebook	25.0 %	18.0 %	0.10 %	92	79.5	56.1
Camcard	21.0 %	0.30 %	0.0 %	52.6	38.7	0.0
Correo Electrónico	20.0 %	0.0 %	0.0 %	30.4	25.0	0.0
Dropbox	19.0 %	0.0 %	0.0 %	69.0	69.0	9.8
Cámara	19.0 %	0.0 %	0.0 %	69.3	67.1	0.0
Polaris Doc	17.0 %	2.40 %	0.0 %	83.8	60.7	0.0
Calendario Google	15.0 %	0.70 %	0.0 %	71.8	53.6	7.8
MaxxAudio	10.0 %	1.0 %	0.0 %	16.4	13.4	13.8
Polaris Pptx	8.0 %	0.7 %	0.0 %	70.5	53.2	0.0
Amazon Apps	7.0 %	1.0 %	0.0 %	109.7	87.6	17.4
Evernote	0.30 %	0.10 %	0.0 %	51.8	46.6	0.0
Amazon Local	0.0 %	0.0 %	0.0 %	29.7	23.8	0.0

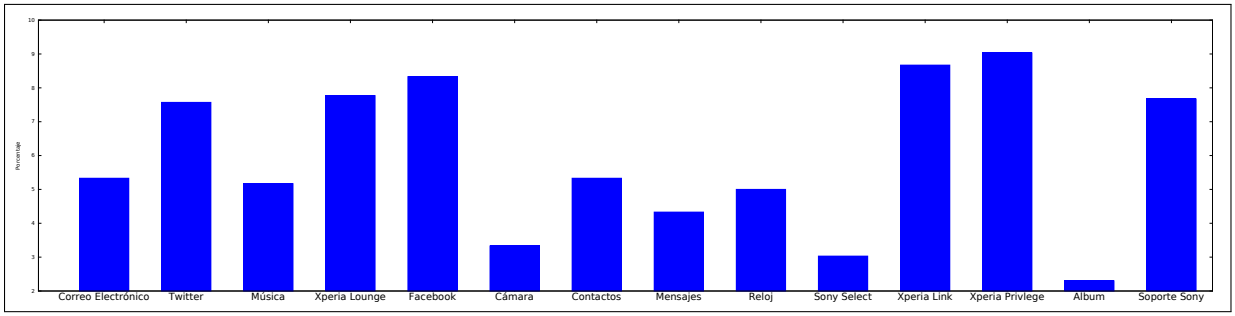
Tabla B.3: Medición del CPU y Memoria de aplicaciones del Dell Venue 7 (versión 4.4)

	CPU			Memoria (en MB)		
Nombre de la aplicación	Aplicación abierta	En el fondo	Aplicación cerrada	Aplicación abierta	En el fondo	Aplicación cerrada
Fotos	16.0 %	0.0 %	0.0 %	168	88.5	0.0
Ayuda	12.0 %	0.0 %	0.0 %	72.8	68.5	4.5
Teléfono	10.0 %	10.0 %	10.0 %	27.8	28.5	28.9
Play Música	8.0 %	3.0 %	1.0 %	77.2	69.3	0.0
Play Películas	6.2 %	6.2 %	6.2 %	50.3	69.3	0.0
Reloj	5.0 %	0.0 %	0.0 %	24.2	20.7	0.0
Mensajes	4.7 %	4.7 %	1.4 %	40.9	32.8	8.5
Cámara	4.0 %	0.0 %	0.0 %	70.5	25.5	5.9
Calendario	3.0 %	1.0 %	0.0 %	44.7	38.9	0.0
Clima	2.0 %	3.90 %	3.9 %	36.8	34.5	27.4
Radio FM	0.0 %	1.40 %	0.0 %	29.2	33.4	2.4
Moto	0.0 %	0.0 %	0.0 %	7.8	7.0	0.0

Tabla B.4: Medición del CPU y Memoria de aplicaciones del Motorola Moto G (versión 6.0)

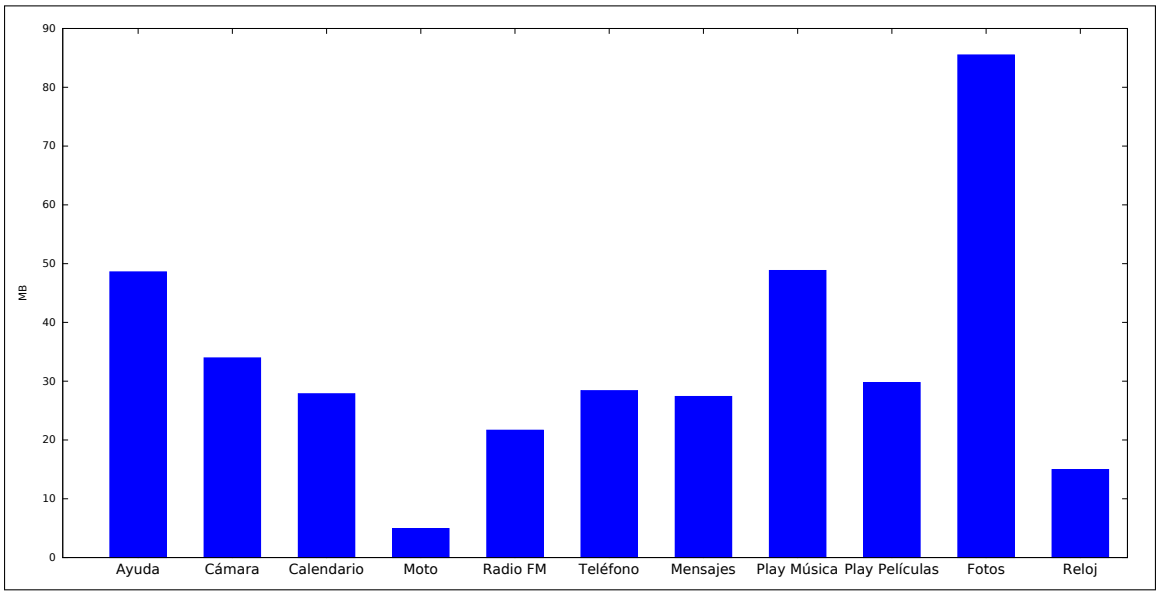


(a)

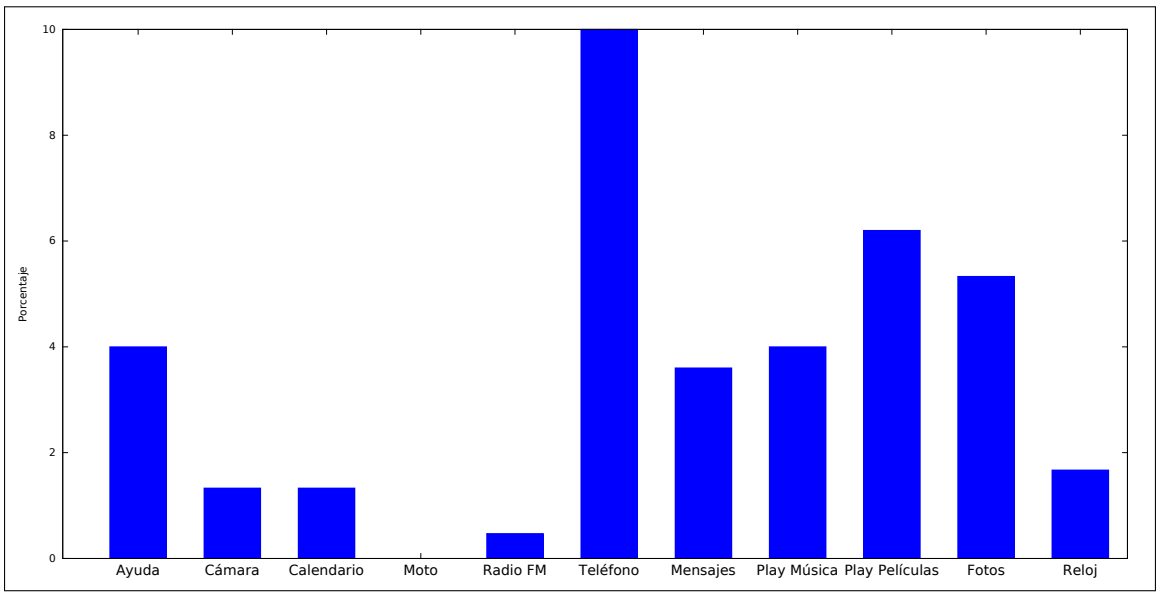


(b)

Figura B.1: Consumo de recursos de las aplicaciones de Sony Xperia SP (versión 4.3) (a) Consumo de memoria (en MB); (b)Consumo de CPU (en porcentaje)

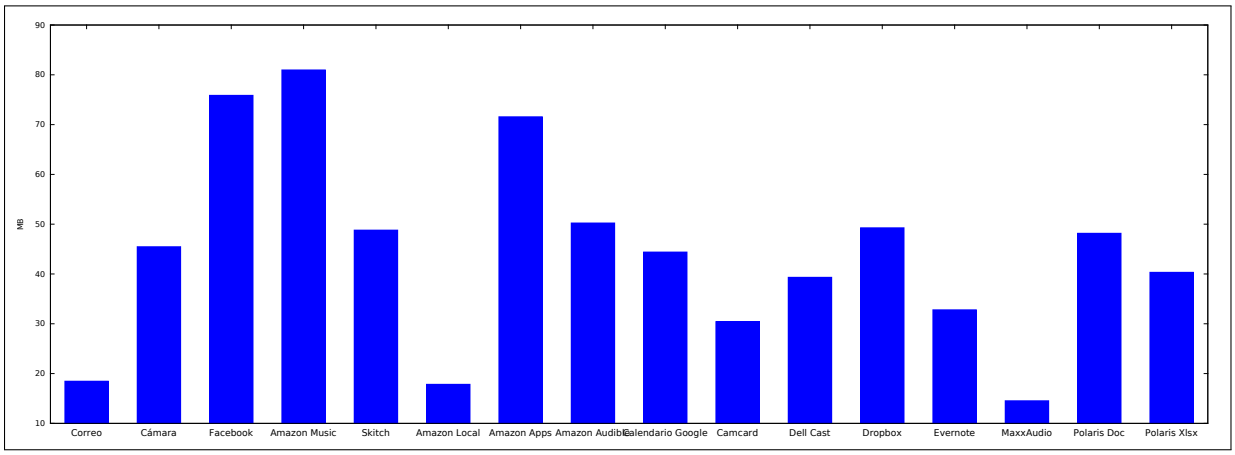


(a)

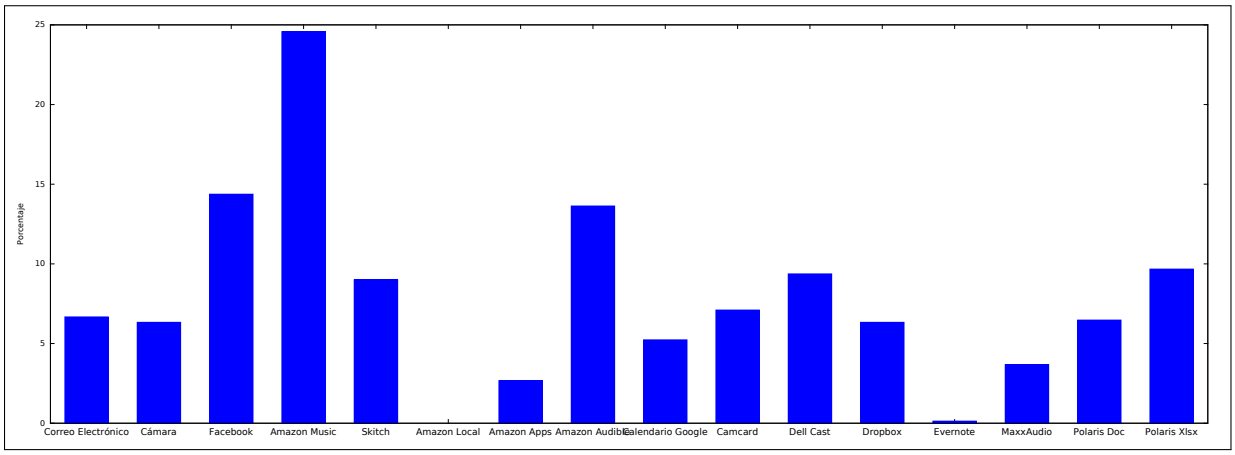


(b)

Figura B.2: Consumo de recursos de las aplicaciones de Moto G 4ta generación (versión 6.0) (a) Consumo de memoria (en MB); (b)Consumo de CPU (en porcentaje)

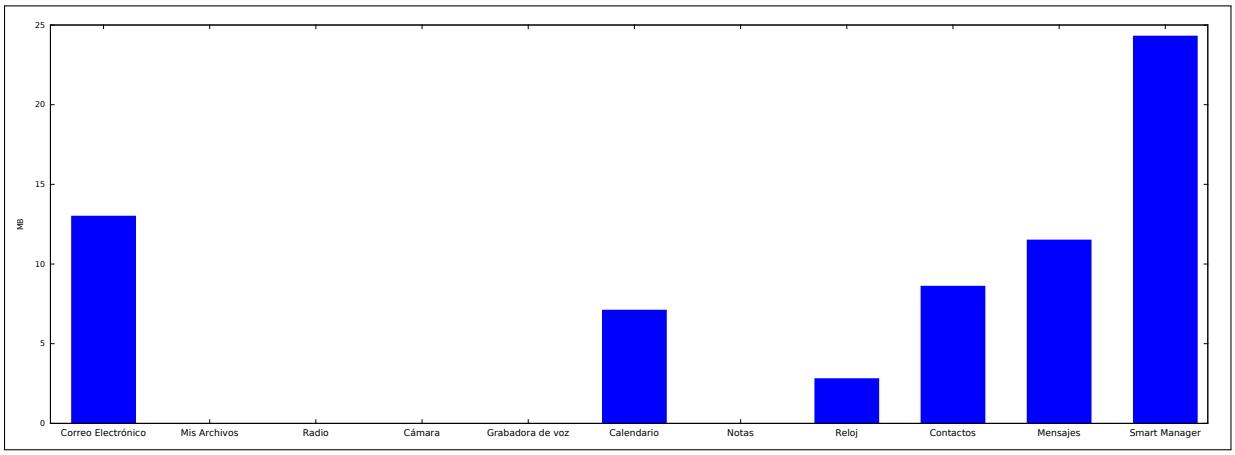


(a)

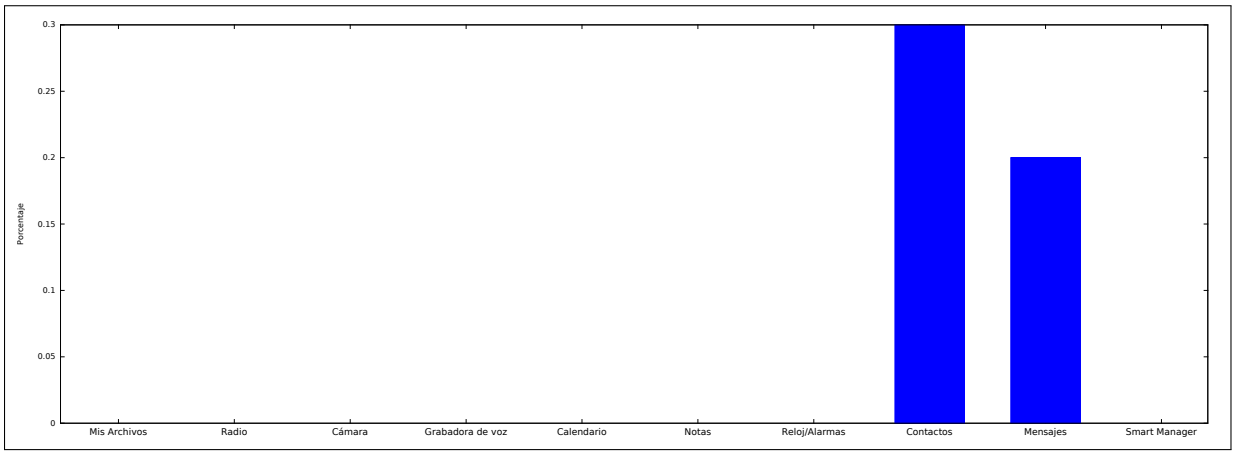


(b)

Figura B.3: Consumo de recursos de las aplicaciones de Dell Venue 7 (versión 4.4) (a) Consumo de memoria (en MB); (b)Consumo de CPU (en porcentaje)



(a)



(b)

Figura B.4: Consumo de recursos de las aplicaciones de Samsung Galaxy Grand Prime (versión 5.1) (a) Consumo de memoria (en MB); (b)Consumo de CPU (en porcentaje)