UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

ANÁLISIS DE VULNERABILIDADES DE LAS APLICACIONES DE CONTROL REMOTO EN DISPOSITIVOS MÓVILES CELULARES CON SISTEMA OPERATIVO ANDROID, UTILIZANDO HERRAMIENTAS INFORMÁTICAS OPEN SOURCE.

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO
DE INGENIERA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

LIZETH CAROLINA ARIAS LLIVICOTA

DIRECTOR: ING. BOLÍVAR JÁCOME

Quito, Julio 2017

DERECHOS DE AUTOR

© Universidad Tecnológica Equinoccial. 2017

Reservados todos los derechos de reproducción

FORMULARIO DE REGISTRO BIBLIOGRÁFICO

PROYECTO DE TITULACIÓN

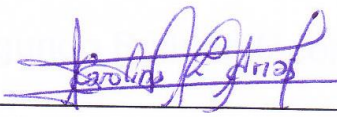
DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	172182207-8
APELLIDOS Y NOMBRES:	ARIAS LLIVICOTA LIZETH CAROLINA
DIRECCIÓN:	VELASCO CALLE “H” N74-43
EMAIL:	carolina_199334@hotmail.com
TELÉFONO FIJO:	02 2493433
TELÉFONO MOVIL:	0984903793

DATOS DE LA OBRA	
TITULO:	Análisis de vulnerabilidades de las aplicaciones de control remoto en dispositivos móviles celulares con sistema operativo Android, utilizando herramientas informáticas OPEN SOURCE
AUTOR O AUTORES:	CAROLINA ARIAS
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	JULIO 2017
DIRECTOR DEL PROYECTO DE TITULACIÓN:	ING. BOLÍVAR JACOME
PROGRAMA	**PREGRADO** [X] **POSGRADO** [ ]
TITULO POR EL QUE OPTA:	INGENIERA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
RESUMEN: Mínimo 250 palabras	El presente documento muestra el análisis de las vulnerabilidades que presentan las aplicaciones de control remoto en el sistema operativo Android. Mediante la utilización de la herramienta Open Source MobSF se evaluaron las posibles amenazas a las que están expuestos los dispositivos móviles celulares. Para realizar el análisis se utilizó la metodología OWASP, la cual permitió conocer las principales vulnerabilidades presentadas en dispositivos móviles celulares

	en los últimos años. La herramienta APK Extractor permitió extraer el código fuente de las distintas aplicaciones, para recolectar la información necesaria que facilitó la realización del análisis estático y dinámico de las mismas. Mediante la aplicación de la metodología se logró indagar en el código fuente de las aplicaciones de control remoto y monitorear en tiempo real cuales son los procesos que se ejecutan en segundo plano y obtener como resultado un informe donde se visualizan cuáles son las amenazas que hacen vulnerable a los dispositivos móviles celulares.
PALABRAS CLAVES:	Vulnerabilidades móviles, aplicaciones móviles de control remoto, análisis estático móvil, análisis dinámico móvil.
ABSTRACT:	This document shows the vulnerability analysis of remote control applications on the Android operating system. The use of the Open Source MobSF tool evaluated the possible threats to which mobile cellular devices are exposed. To perform the analysis, the OWASP methodology was used, which allowed to know the main vulnerabilities presented in cellular mobile devices in recent years. The APK Extractor tool allowed extracting the source code of the different applications, to collect the necessary information that facilitated the static and dynamic analysis of the same ones. Through the application of the methodology it was possible to investigate in the source code of the remote control applications and to monitor in real time which are the processes that run in the background and to obtain as a result a report where they are visualized

	which are the threats that make mobile cellular devices vulnerable.
KEYWORDS	Mobile vulnerabilities, mobile remote control applications, mobile static analysis, mobile dynamic analysis.

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución

f: 

ARIAS LLIVICOTA LIZETH CAROLINA

172182207-8

DECLARACIÓN Y AUTORIZACIÓN

Yo, **LIZETH CAROLINA ARIAS LLIVICOTA**, CI: 172182207-8 autor del proyecto titulado: **ANÁLISIS DE VULNERABILIDADES DE LAS APLICACIONES DE CONTROL REMOTO EN DISPOSITIVOS MÓVILES CELULARES CON SISTEMA OPERATIVO ANDROID, UTILIZANDO HERRAMIENTAS INFORMÁTICAS OPEN SOURCE.**, previo a la obtención del título de **INGENIERA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN** en la Universidad Tecnológica Equinoccial.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la BIBLIOTECA de la Universidad Tecnológica Equinoccial a tener una copia del referido trabajo de graduación con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, 06 de Julio del 2017.

f _____

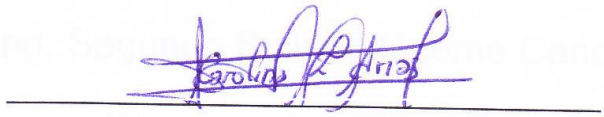
ARIAS LLIVICOTA LIZETH CAROLINA

172182207-8

DECLARACIÓN

Yo **ARIAS LLIVICOTA LIZETH CAROLINA**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

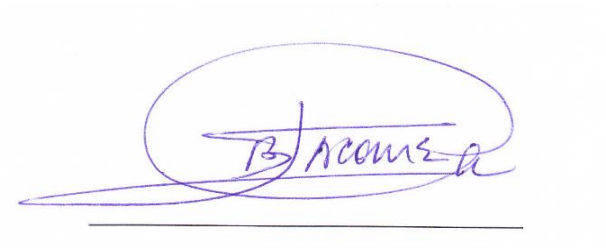


Lizeth Carolina Arias Llivicota

C.I. 172182207-8

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título "**Análisis de vulnerabilidades de las aplicaciones de control remoto en dispositivos móviles celulares con sistema operativo Android, utilizando herramientas informáticas OPEN SOURCE.**", que, para aspirar al título de **Ingeniera en Informática y Ciencias de la Computación** fue desarrollado por **Lizeth Carolina Arias Llivicota**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería e Industrias; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 19, 27 y 28.



Ing. Segundo Bolívar Jácome Canchig

DIRECTOR DEL TRABAJO

C.I.1707004618

DEDICATORIA

Quiero agradecer a Dios por darme las fuerzas necesarias para poder culminar mi vida universitaria.

A mi madre por ser una persona incondicional en mi vida, apoyándome en los momentos buenos y malos durante todo el transcurso de mi vida estudiantil y personal.

A mi padre por ser el apoyo fundamental para terminar mis estudios y a mis hermanos por estar presente en cada logro y fracaso para ser una persona mejor, ya que gracias a ellos pude salir adelante a pesar de los diferentes tropezones que tuve a lo largo de mi carrera.

ÍNDICE DE CONTENIDOS

PÁGINA

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE FIGURAS

PÁGINA

RESUMEN

El presente documento muestra el análisis de las vulnerabilidades que contienen las aplicaciones Team Viewer, Sure Universal y Peel Smart Remote en el sistema operativo Android. Mediante la utilización de la herramienta Open Source MobSF se evaluaron las posibles amenazas a las que están expuestos los dispositivos móviles celulares. Para realizar el análisis se utilizó la metodología OWASP, la cual permitió conocer las principales vulnerabilidades presentadas en dispositivos móviles celulares en los últimos años. La herramienta APK Extractor permitió extraer el código fuente de las distintas aplicaciones, para recolectar la información necesaria que facilitó la realización del análisis estático y dinámico de las mismas. Mediante la aplicación de la metodología se logró indagar en el código fuente de las aplicaciones de control remoto y monitorear en tiempo real cuales son los procesos que se ejecutan en segundo plano y obtener como resultado un informe donde se visualiza cuáles son las amenazas que hacen vulnerables a los dispositivos móviles celulares.

Palabras clave: Vulnerabilidades móviles, aplicaciones móviles de control remoto, análisis estático móvil, análisis dinámico móvil.

ABSTRACT

This document shows the vulnerability analysis of the Team Viewer, Sure Universal and Peel Smart Remote applications on the Android operating system. The use of the Open Source MobSF tool evaluated the potential threats to which mobile cellular devices are exposed. To perform the analysis, the OWASP methodology was used, which allowed to know the main vulnerabilities presented in cellular mobile devices in recent years. The APK Extractor tool allowed extracting the source code of the different applications, to collect the necessary information that facilitated the static and dynamic analysis of the same ones. Through the application of the methodology it was possible to investigate in the source code of the remote control applications and to monitor in real time which are the processes that run in the background and to obtain as a result a report where it is visualized which are the threats that make mobile cellular devices vulnerable.

Keywords: Mobile vulnerabilities, mobile remote control applications, mobile static analysis, mobile dynamic analysis.

1. INTRODUCCIÓN

Actualmente utilizar varias aplicaciones libres en dispositivos móviles celulares con sistema operativo Android, ocasiona que la información de los usuarios se encuentre expuesta y vulnerable a cualquier tipo de ataque informático o de uso no autorizado.

Los dispositivos móviles celulares con sistema operativo Android han permitido que las personas y empresas los utilicen para realizar diversas tareas y a su vez guardar información, la misma que, por medio de aplicaciones, malware y antivirus falsos puede ser afectada por apps que se encuentran en la plataforma digital Google Play (Agustín & Carlos, 2013).

Las aplicaciones en su mayoría son cargadas sin tener un control adecuado antes de publicarlas, por ende cualquier desarrollador de software puede crear una aplicación sin un control de seguridad con la intensión de acceder a la información del dispositivo, lo cual indica que no todas las aplicaciones de redes sociales, juegos, antivirus, aplicaciones de control remoto, entre otras, son seguras (Agustín & Carlos, 2013).

Por lo antes indicado, se analizará las aplicaciones de control remoto en los dispositivos móviles con sistema operativo Android, ya que se realizó una encuesta a los docentes y estudiantes de la Universidad Tecnológica Equinoccial para conocer con qué frecuencia este tipo de apps se utiliza.

Las herramientas Open Source como VEGA, MobSF, VTS para Android, Nessus, entre otras, permiten realizar un análisis profundo de las vulnerabilidades que pueden contener las distintas aplicaciones, mostrando de esta manera los procesos que realizan o indicando a qué tipo de información se puede acceder.

El objetivo de este trabajo es un análisis de las diferentes vulnerabilidades que existen en las aplicaciones de control remoto en dispositivos móviles celulares con sistema operativo Android, utilizando herramientas informáticas Open Source. Para cumplir dicho propósito, primeramente se investigará los tipos de vulnerabilidades que existen en la actualidad para los dispositivos móviles celulares que utilicen el sistema operativo Android, a continuación se analizará los diferentes permisos y código malicioso que pueda existir dentro de este tipo de aplicaciones. Finalmente, con los resultados obtenidos, se brindaran recomendaciones de seguridad para reducir la filtración de información en dispositivos móviles celulares que utilicen aplicaciones de control remoto comprometidas.

2. MARCO TEÓRICO

2.1 ESTADO DEL ARTE DE ANÁLISIS DE VULNERABILIDADES EN ANDROID

Android es un sistema operativo de código abierto con más de mil millones de usuarios activos para todos sus dispositivos móviles. Para la protección de la información y de los dispositivos, Android incorpora varios mecanismos de seguridad, entre los más destacados se encuentran: un entorno sandbox al nivel Kernel para prevenir el acceso al file-system y otros recursos, una API de permisos que controla los privilegios de las aplicaciones sobre el dispositivo, mecanismos de seguridad a nivel del desarrollo de aplicaciones, y su plataforma de distribución digital (Google Play) (Christian Camilo Urcuqui López, 2016).

El 25% de las aplicaciones Android tienen permisos que pueden generar vulnerabilidades de seguridad. Así lo asegura la empresa de seguridad móvil TrustGo que analizó 2.3 millones de aplicaciones para el sistema operativo Android; el último trimestre de 2013, se detectó un total de 511.000 apps considerados de ‘alto riesgo’ para los usuarios, una calificación que se aplica cuando un programa es capaz de hacer pagos no autorizados, robar datos o modificar los ajustes del usuario (Foresti, Yung, & Martinelli, 2012).

Durante los últimos años el desarrollo de software malicioso (malware) para Android se ha incrementado; esto se debe a distintas razones, por ejemplo: la filosofía de Android, su posicionamiento en el mercado de móviles y a la cantidad de información sensible que se produce en estas tecnologías. Existen distintas herramientas y técnicas para el análisis de amenazas para este sistema operativo (Christian Camilo Urcuqui López, 2016).

Entre las técnicas más representativas se encuentran:

- Análisis estático, técnica para evaluar los comportamientos maliciosos en el código fuente, datos o archivos binarios sin ejecutar directamente la aplicación.
- Análisis dinámico, métodos que estudian el comportamiento del malware en tiempo real, esta técnica permite analizar los procesos, sockets, conexiones, entre otros.

También se utilizó como lenguaje de desarrollo Python y las herramientas de machine learning, para abarcar el proceso de evaluación de los algoritmos de acuerdo a los permisos que contenga el archivo AndroidManifest.xml y de este modo comprobar si la aplicación es benigna o maligna (Christian Camilo Urcuqui López, 2016).

No existe ningún software 100% seguro, y cuánto mayor es su cuota de mercado más fácil es que los usuarios, expertos de seguridad y piratas informáticos descubran nuevas vulnerabilidades para atacar a los dispositivos móviles. Es prácticamente imposible llevar un seguimiento de todas las vulnerabilidades de Android y comprobarlas una a una. Para facilitar esta tarea existen aplicaciones que se encargan de recopilar todas estas vulnerabilidades y comprobar si están presentes o no en el celular. Una de estas aplicaciones es VTS for Android (Casas, 2017).

VTS for Android es una aplicación gratuita y de código abierto desarrollada para descubrir si nuestro smartphone o tablet es vulnerable, y de serlo, a qué fallos de seguridad está expuesto. Esta aplicación analiza todas las aplicaciones instaladas en el dispositivo, detallando las vulnerabilidades existentes en cada app y desplegando una ventana en la que indica por medio de enlaces cómo se pueden corregir dichas amenazas o la información que ha sido afectada sin que el usuario se de cuenta (Casas, 2017).

2.2 SISTEMA OPERATIVO ANDROID

Los dispositivos móviles con sistema operativo Android son los más utilizados en Latinoamérica, con un porcentaje del 48.96% según estudios realizados por ESET, esto se debe a que las personas ocupan estos dispositivos como una herramienta útil tanto para su desempeño personal como laboral (Llaven, 2015).

Los incidentes de seguridad en dispositivos móviles con sistema operativo Android se han incrementado de manera alarmante debido a la preferencia de sus usuarios, esto los hace más vulnerables de ser atacados, presentando un alto riesgo de vulnerabilidad en la violación de información confidencial de las personas que administran estos dispositivos (Llaven, 2015).

En la figura 1 se puede observar que las vulnerabilidades en dispositivos móviles en el año 2015, con un 58% crecieron en un 2% con respecto al año 2014, con 56%; de igual manera el malware en el año 2015, con un 55%, creció un 3% con respecto al año 2014, con un 58%. Los casos de vulnerabilidades que se mediatizaron, son relacionados con una falla de seguridad en OpenSSL y con el protocolo de cifrado HTTPS (ESET Latinoamérica, 2016).

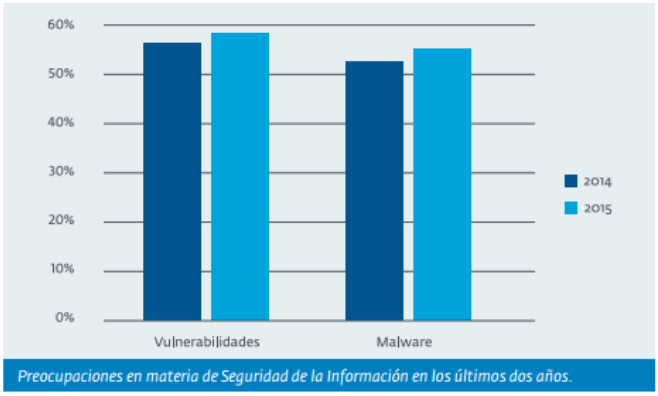


Figura 1. Porcentaje de la seguridad en Android los 2 últimos años
(ESET Latinoamérica, 2016)

El equipo de investigación de ESET Latinoamérica manifiesta que el sistema operativo Android se convirtió en el más utilizado en los equipos móviles como Smartphone y Tablets del mundo, y a su vez Android apunta a un uso cada vez mayor de este sistema, lo que permite explicar el aumento y consolidación de diversas vulnerabilidades o amenazas informáticas que afectan a dicha plataforma (ESET Latinoamérica, 2014).

También los dispositivos móviles con sistema operativo Android se vieron afectados por vulnerabilidades de mayor impacto, como es Stagefright, un fallo de seguridad a través del cual se puede hackear un teléfono enviando contenido multimedia en forma de mensajes de texto o a través de un archivo con formato MP3 o MP4 (ESET Latinoamérica, 2016).

A pesar de ello, los dispositivos móviles son cada vez más utilizados para desempeñar diferentes tipos de actividades personales o laborales. Del mismo modo, estos incidentes aclaran por qué las vulnerabilidades de software no solo siguen siendo la principal preocupación de las empresas de la región, sino también una tendencia creciente (Pérez, 2015).

Este crecimiento va acompañado de un aumento directamente proporcional con la cantidad de códigos maliciosos que se desarrollaron para Android. Asimismo, la evolución de algunas amenazas para este sistema operativo y el descubrimiento de ciertas vulnerabilidades, demuestran el creciente interés de los cibercriminales por atacar este segmento (Pérez, 2015).

2.3 MODELO DE SEGURIDAD DE ANDROID

La implementación del modelo de seguridad de Android se lleva a cabo a lo largo de toda la arquitectura del sistema; los aspectos más importantes son (Romano & Luna, 2013):

- Sandbox: Android implementa esta app para forzar a que cada aplicación solo pueda tener acceso irrestricto a sus propios recursos.
- Signing: todas las aplicaciones Android deben estar firmadas digitalmente de forma tal que sus claves privadas solo sean conocidas por sus respectivos desarrolladores.
- Permisos: una aplicación declara estáticamente, en su archivo AndroidManifest, el conjunto de permisos que necesita para obtener las capacidades adicionales que no tiene por defecto.
- Delegación de permisos: utiliza mecanismos que permiten a una aplicación dada delegar sus propios permisos a otra por medio de intents o uri.

2.3.1 AMENAZAS PARA EL SISTEMA OPERATIVO ANDROID

Las preocupaciones de los dispositivos móviles pueden determinarse a través de distintos factores, como las tendencias en materia de amenazas informáticas o los incidentes de seguridad más comunes. Por ende la principal preocupación son las “vulnerabilidades de software y sistemas” con el 58% de las respuestas afirmativas, seguido por el “malware” (54%) y, en el tercer puesto, el “acceso indebido la información” (46%) (ESET Latinoamérica, 2016). En la Figura 2 se puede observar las preocupaciones de seguridad de la información mencionadas anteriormente.

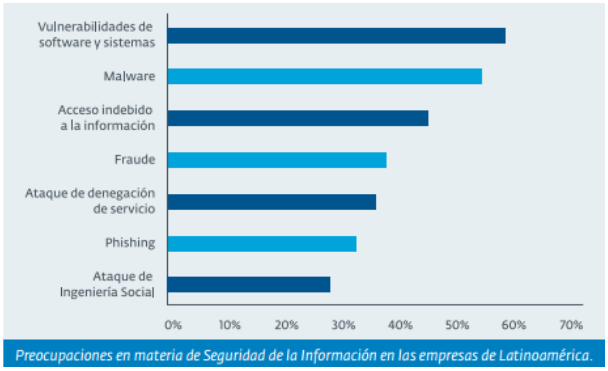


Figura 2. Crecimiento de amenazas en S.O. Android
(ESET Latinoamérica, 2016)

El fraude informático se produjo por el acceso indebido a la información, de este modo se explica el aumento de la explotación de vulnerabilidades que generalmente tiene como consecuencia la perdida de información, razón por la que ambos riesgos pudieron haber crecido proporcionalmente (Ibáñez & Navarro, 2014).

2.4 VULNERABILIDADES EN LOS CELULARES

Los investigadores de ESET calculan que existen alrededor de 2 mil millones de dispositivos que están bajo la administración de OMA-DM(Open Mobile Alliance - Device Manager), entidad que permite a las operadoras, lanzar actualizaciones, y a su vez OTA (OverThe Air) que permite controlar el sistema operativo de los dispositivos móviles. No obstante, este sistema también puede ejecutar una serie de comandos remotos que, de ser usados de forma maliciosa, podrían comprometer la seguridad de los dispositivos y la información que almacenan, e incluso podrían suponer el control remoto del mismo (Albors, 2014).

2.4.1 WI-FI

La conexión a internet es una de las cosas más habituales en nuestra vida diaria, uno de los principales métodos de conexión son las redes Wi-Fi. El sistema Wi-Fi es una de las tecnologías que más se usa para conectarse a internet, aunque esto no implica que sea el método más seguro. Actualmente las redes Wi-Fi tienen grandes problemas de seguridad, ya que una red inalámbrica viene determinada por varios aspectos, los cuales se pueden configurar desde el router (Laudon & Laudon, 2014).

2.4.2 LOS DISPOSITIVOS MÓVILES PUEDEN CONTENER MALWARE

Los usuarios pueden descargar aplicaciones que contienen malware los cuales son programas informáticos maliciosos con el fin de robar información privada. Los consumidores descargan estos programas sin saberlo, ya que pueden estar disfrazados como un juego, parche de seguridad, utilidad o aplicación. Es difícil que los usuarios noten la diferencia entre una aplicación legítima y una que contenga un software malicioso (Staff, 2016).

La detección de malware ha sido un área de investigación relevante en los últimos años. Si bien esta investigación se centró principalmente en los ordenadores de escritorio, en los últimos tiempos, con la mejora en las capacidades de los teléfonos móviles, éstos han sido objeto de varias investigaciones (Urquijo, 2012).

2.4.3 CATEGORÍAS DE LOS MALWARE QUE AFECTAN A LOS CELULARES

Los principales malware que afectan a los dispositivos móviles se clasifican en (Urquijo, 2012) :

- **Robo de privacidad**: sustrae datos de carácter privado de las víctimas.
- **Industria del malware**: los creadores de malware pueden obtener beneficio económico a través de la venta en el mercado negro de la información sustraída de las víctimas.
- **Cuota de consumo**: consumo sin permiso expreso de servicios de tarificación especial.
- **Puerta trasera**: se ejecutan en segundo plano y abren algún puerto para acceder o controlar los celulares.

En la figura 3 se puede observar las muestras de malware analizadas con respecto al robo de información personal. Cabe resaltar que los dispositivos

móviles celulares almacenan una gran cantidad de información personal, datos de usuarios y contraseñas, cuentas de correo electrónico o de redes sociales.

Figura 3. Malware dentro de la plataforma Android
(Urquijo, 2012)

2.4.4 RANSOMWARE

Una de las actividades más rentables del mundo del ciber crimen, se consolidó en plataformas móviles con nuevas técnicas para bloqueo de los equipos, por medio de aplicaciones móviles como WhatsApp o Facebook para aumentar el alcance de campañas de malware multiplataforma haciendo uso de viejas técnicas de ingeniería social (ESET Latinoamérica, 2016).

2.4.5 VIRUS, GUSANOS Y TROYANOS

Los virus son aquellas aplicaciones que se replican a sí mismas, normalmente tras una acción concreta (ejemplo: el usuario ejecuta un programa) (Kruegel, 2008).

Los gusanos, por su parte, tienen el mismo objetivo que el virus, replicarse a sí mismo, sin embargo, utiliza la red para enviar las copias (Kruegel, 2008).

Por último, los troyanos son programas escondidos dentro de programas legítimos que ejecutan fragmentos de código maliciosos (Kruegel, 2008).

2.5 APLICACIONES DE CONTROL REMOTO

Actualmente existen diferentes tipos de aplicaciones que permiten convertir a los dispositivos móviles en un control remoto conectándose por medio de Wi-FI, infrarrojo o Bluetooth.

A continuación se detalla algunas de las aplicaciones de control remoto utilizadas en los dispositivos móviles.

2.5.1 TEAM VIEWER

Team Viewer es una aplicación intuitiva, rápida y segura para el control remoto de un computador a través de la red Wi-Fi, además todas las conexiones están encriptadas y protegidas para el acceso por parte de terceros (Lahaie & Leberfinger, 2013). En la Figura 4 se muestra la pantalla inicial de Team Viewer en donde se verifica el id y contraseña que se necesita para conectarse con otro dispositivo.

Figura 4. Ventana principal de TeamViewer
(Lahaie & Leberfinger, 2013)

2.5.2 SURE UNIVERSAL

SURE es una solución de control remoto universal gratis para electrodomésticos y medios digitales. También controla Smart TV y streamers de medios, para enviar imágenes y videos desde el teléfono a un televisor, además muestra un panel principal para la conexión a dispositivos con infrarrojo, Wi-Fi y otros sistemas (Acevedo, 2016) . La Figura 5 muestra la pantalla donde se configuran los dispositivos.

Figura 5. Aplicación SURE Universal (Acevedo, 2016)

2.5.3 PEEL SMART REMOTE

Esta aplicación convierte al teléfono inteligente o tableta en un control remoto de televisión, ya que puede reemplazar a los controles de Samsung TV, pero también de las LG TV, Sony TV, DirecTV, e incluso Apple TV (Jaramillo, 2016). La Figura 6 muestra la pantalla de las opciones que contiene esta app.

Figura 6. Aplicación PEEL SMART REMOTE.
(Jaramillo, 2016)

2.6 HERRAMIENTAS OPEN SOURCE PARA EL ANÁLISIS DE VULNERABILIDADES.

Estas herramientas analizan las vulnerabilidades y amenazas presentes en las aplicaciones para dispositivos móviles, identificando los permisos, librerías, dominios, malware, entre otros, para verificar la manipulación de los datos almacenados en los celulares e identificar los recursos que requieren dichas apps.

En el internet existen diferentes aplicaciones que permiten el estudio de las diferentes vulnerabilidades, entre las que se pueden mencionar.

2.6.1 APK EXTRACTOR

Es una herramienta útil con la que se puede extraer cualquier archivo APK de todas las aplicaciones instaladas en el dispositivo. Este programa tiene un funcionamiento simple, únicamente se debe abrir su interfaz para acceder a la lista de aplicaciones que tenemos instaladas en la memoria del

celular o en la de la tarjeta micro SD y mantener pulsado sobre la app de la que se quiere extraer su APK (Herrera, 2015).

2.6.2 MARCO DE SEGURIDAD MÓVIL (MOBSF)

Es una herramienta de código abierto que permite realizar test de penetración automatizado, análisis estático y dinámico tanto en aplicaciones Android como en iOS. Puede utilizarse para el análisis de seguridad rápido y eficaz de las apps móviles, siendo compatible con los binarios (APK e IPA) y el código fuente comprimido. Esta aplicación pretende minimizar el tiempo que con un conjunto de programas se puede realizar: decodificación, depuración y revisión de código. Además puede realizar pruebas de seguridad de la API Web para analizar cabeceras de seguridad e identificar vulnerabilidades específicas en XXE (Entidad externa Xml) (Dalziel & Abraham, 2016).

2.7 MÉTODOS DE ANÁLISIS DE VULNERABILIDADES

Existen diferentes metodologías para el análisis de las aplicaciones de control remoto, entre las que se puede mencionar.

2.7.1 PROYECTO ABIERTO DE SEGURIDAD EN APLICACIONES WEB (OWASP)

El Proyecto de Seguridad OWASP móvil es un recurso centralizado destinado a dar a los desarrolladores y equipos de seguridad las herramientas que necesita para construir y mantener aplicaciones móviles seguras. El objetivo es clasificar los riesgos de seguridad móvil y proporcionar controles de desarrollo para adquirir y mantener apps que puedan ser confiables reduciendo su impacto o la probabilidad de explotación de vulnerabilidades (Totzek-Hallhuber, 2017).

2.7.1.1 Top 10 de vulnerabilidades en aplicaciones móviles OWASP 2016

El objetivo del proyecto Top 10 es conocer acerca de la seguridad en dispositivos móviles mediante la identificación de algunos de los riesgos críticos que enfrentan las aplicaciones móviles. A continuación se detallan las diez categorías de OWASP que están centradas en el estudio de las vulnerabilidades (Chell, Erasmus, Colley, & Whitehouse, 2015).

- M1: uso incorrecto de la plataforma.
- M2: almacenamiento de datos inseguro.
- M3: comunicación insegura.
- M4: autentificación insegura.
- M5: criptografía insuficiente.
- M6: autorización no segura.
- M7: calidad del código del cliente.
- M8: codificación de código.
- M9: ingeniería inversa.
- M10: funcionalidad extraña.

2.7.2 METODOLOGÍA ABIERTA DE EVALUACIÓN PARA LA SEGURIDAD ANDROID (OASAM)

El objetivo de esta metodología es ser un marco de referencia de análisis de vulnerabilidades en aplicaciones Android, elaborando una taxonomía completa y consistente, que sirva de apoyo no solo a los desarrolladores de aplicaciones, sino también a los encargados de buscar vulnerabilidades en las mismas (Li, Holtkamp, Wang, & Han, 2009)

2.7.2.1 Controles de seguridad de OASAM

Existen diferentes secciones para poder evaluar las vulnerabilidades dentro de esta metodología, las cuales se describen a continuación (Li, Holtkamp, Wang, & Han, 2009) :

- **OASAM-INFO** (Information Gathering). Obtención de información y definición de superficie de ataque
- **OASAM-CONF** (Configuration and Deploy Management). Análisis de la configuración e implantación
- **OASAM-AUTH** (Authentication). Análisis de la autenticación
- **OASAM-CRYPT** (Cryptography). Análisis del uso de criptografía
- **OASAM-LEAK** (Information Leak). Análisis de fugas de información sensible
- **OASAM-DV** (Data Validation). Análisis de gestión de la entrada de usuario
- **OASAM-IS** (Intent Spoofing). Análisis de la gestión en la recepción de intents[1]
- **OASAM-UIR** (Unauthorized Intent Receipt). Análisis de la resolución de intents

2.7.3 METODOLOGÍA PARA ANÁLISIS DE RIESGOS DE VULNERABILIDADES Y AUDITORÍAS DE DISPOSITIVOS.

Esta metodología trata de la evaluación de los dispositivos móviles cuyo fin es detectar errores y fallos en el sistema, y mediante un informe detallado se puede mostrar los resultados verificados y las recomendaciones a tomar, enfocándose en la protección total de los recursos expuestos a posibles amenazas de seguridad informática (Altarejos, 2017)

[1] Intents: Objeto de acción que se usa para solicitar una acción de otro componente de la aplicación en el sistema Android.

2.7.3.1 Fases de la metodología

A continuación se detallan las diferentes fases de la metodología que permite aplicarla a los riesgos informáticos en entornos compuestos por dispositivos móviles (Altarejos, 2017)

- **Identificación:** consiste en identificar y definir los sistemas a auditar.
- **Análisis:** análisis de los servicios de red en escucha o puertos activos y servicios inalámbricos en uso.
- **Acceso:** identificar las vulnerabilidades del dispositivo.
- **Resultados:** describir los procesos realizados hasta llegar a obtener el objetivo, técnicas utilizadas y las herramientas asociadas.
- **Informes**: luego de obtener los resultados, se emite un informe indicando el establecimiento de las medidas preventivas de refuerzo y/o corrección de los problemas obtenidos.

3. METODOLOGÍA

3.1 MÉTODOS DE INVESTIGACIÓN

Para el desarrollo de este proyecto se utilizaron varios métodos de investigación científica durante las diferentes fases del desarrollo del trabajo de titulación, cuya aplicación se resume en la Tabla 1.

Tabla 1. Resumen de métodos de investigación

MÉTODOS DE INVESTIGACIÓN		APLICACIÓN	FASE DE LA INVESTIGACIÓN
Teóricos	Inductivo	-Identificación del problema a resolver	Introducción
	Deductivo	- En el análisis del código de la aplicación, utilizando herramientas Open Source.	Metodología
	Analítico - Sintético	-Selección de las fuentes de información y elaboración de contenidos sistematizados para los temas.	Marco Teórico
Empíricos	Experimentación	-Utilización de la herramienta Open Source para el análisis de vulnerabilidades en las aplicaciones de control remoto.	Análisis de Resultados

3.2 METODOLOGÍA PARA EL ANÁLISIS DE VULNERABILIDADES

Para el análisis de amenazas y vulnerabilidades de las aplicaciones de control remoto se utilizó la metodología OWASP la cual está dividida en las siguientes fases:

- Fase 1: recopilación de información de las aplicaciones de estudio.
- Fase 2: análisis estático.
- Fase 3: análisis dinámico.

3.1.1 FASE 1: RECOPILACIÓN DE INFORMACIÓN DE LAS APLICACIONES DE ESTUDIO

En esta fase se identificaron las características de la aplicación, tales como: su sistema operativo, interfaces, compatibilidad, funcionamiento, entre otras. Las actividades que se realizaron en esta fase son:

- Se identificaron las aplicaciones que se van analizar.
- Se reconoció el sistema operativo con el cual funciona correctamente la aplicación.
- Se implementó un laboratorio básico de pruebas para llevar a cabo el análisis estático.
- Se establecieron las interfaces de comunicación que se utilizan con las aplicaciones.
- Se detallaron los requisitos con los cuales la aplicación tiene un funcionamiento adecuado.
- Se conoció el funcionamiento básico de la aplicación.
- Se investigaron las herramientas para extraer el código fuente de las aplicaciones.

3.1.2 FASE 2: ANÁLISIS ESTÁTICO

En esta fase se realizó un análisis del código fuente de la aplicación, utilizando ingeniería inversa con la ayuda de descompiladores que facilitaron la extracción del código del archivo APK, para lo cual se realizaron las siguientes actividades:

- Se identificaron las herramientas de software para el análisis de código.
- Se extrajo el código fuente de la aplicación con la herramienta seleccionada.
- Se detalló el certificado con el cual trabaja la aplicación.

- Se reconocieron los algoritmos con los que trabaja el certificado.
- Se verificaron los permisos de la aplicación.
- Se determinaron las librerías o métodos que utiliza la aplicación.
- Se describieron y revisaron los niveles de seguridad.
- Se analizó la información a la cual puede acceder la aplicación.
- Se examinó la gestión de la conexión de la aplicación.
- Se examinó si se inserta sentencias SQLite.

3.1.3 FASE 3: ANÁLISIS DINÁMICO

En esta fase se realizó un análisis de la aplicación en tiempo real, utilizando herramientas que monitorean su comportamiento dentro del dispositivo móvil o en un emulador. A continuación se detallan las actividades que se ejecutaron en esta fase:

- Se identificó la herramienta que ayuda a monitorear la aplicación.
- Se cargó la aplicación en la herramienta seleccionada.
- Se analizaron los sistemas de ficheros que ocupa la aplicación.
- Se determinaron los algoritmos que utiliza la aplicación al momento de la ejecución.
- Se verificó la maniobra de la aplicación por ejemplo: inyección de código.
- Se investigaron los métodos del protocolo HTTP que utiliza la aplicación.
- Se revisaron los permisos que utiliza la aplicación durante la ejecución.

4 RESULTADOS Y DISCUSIÓN

Para el análisis de vulnerabilidades de las aplicaciones móviles, se realizó una encuesta a los docentes y estudiantes de la Universidad Tecnológica Equinoccial, con el objetivo de conocer qué tipo de apps utilizan con mayor frecuencia.

La Tabla 2 muestra los resultados de la encuesta realizada, en la cual se puede observar que las aplicaciones de control remoto tienen mayor frecuencia de uso; por esta razón fueron seleccionadas para el análisis de vulnerabilidades.

Tabla 2. Encuestas de aplicaciones

		APLICACIONES			
		App Música	**App Control Remoto**	**App Películas**	**App Juegos**
Frecuencia	**Permanentemente**	27%	57%	13%	30%
	Parcialmente	43%	30%	50%	47%
	Nada	30%	13%	37%	23%
TOTAL		**100%**	**100%**	**100%**	**100%**

4.1 FASE 1: RECOPILACIÓN DE LA INFORMACIÓN

En esta fase se seleccionaron para el análisis las aplicaciones de control remoto, ya que son las más utilizadas por los usuarios para controlar dispositivos como: PC, televisores, alarmas, proyectores, entre otros, de acuerdo a los resultados presentados en la Tabla 2. La Tabla 3 muestra las apps de este tipo.

Tabla 3. Aplicaciones de control remoto

APLICACIÓN	SISTEMA OPERATIVO	INTERFACES DE COMUNICACIÓN	No. DISPOSITIVOS	FUNCIONALIDADES BÁSICAS
SURE	Android 4.2 o superior	Infrarrojo (IR) Wi-Fi	2 dispositivos o más.	Controla: TV, Proyectores Receptores de AV Electrodomésticos
TEAM VIEWER	Android, iOS	Red Lan Wi-Fi	2 dispositivos.	Computadoras para: Enviar / recibir archivos Chatear Videoconferencias Accesos remotos
PELL SMART REMOTE	Android 4.2 o superior	Infrarrojo (IR)	2 dispositivos o más.	Controla: TV, Decodificadores, DVDs, Calefactores
Universal TV	Android 3.2 o superior	Infrarrojo (IR) Wi-Fi	2 dispositivos o más.	Control TV: LG,Smart TV Android, Apple
IR Remote Control	Android 4.3 o superior	Infrarrojo (IR)	2 dispositivos o más.	Controla TV: LG, Panasonic, Samsung, Sony
Smart IR Remote	4.3 o superior	Infrarrojo (IR)	2 dispositivos o más.	Controla TV Samsung

Para el proceso de análisis se escogieron las aplicaciones SURE, TEAM VIEWER y PELL SMART REMOTE, ya que estas apps son utilizadas con mayor frecuencia porque sirven para controlar diferentes dispositivos en una sola aplicación según los estudios realizados a los usuarios por la revista PCWorld, las mismas que funcionan en el sistema operativo Android y utilizan las interfaces de comunicación Wi-Fi o Infrarrojo (IR).

La Tabla 4 muestra algunas herramientas que permiten analizar el código fuente de las aplicaciones seleccionadas.

Tabla 4. Herramientas para analizar Apk

HERRAMIENTA	SISTEMA OPERATIVO	TIPO DE ANALISIS	APLICACIONES QUE MANEJA	TIPO DE LICENCIA
Mobile Security Framework (MobSF)	Windows,Android o iOS	Estático y Dinámico	Aplicaciones Android, Windows o iOS	Libre
Quick Android Review Kit o QARK	Android	Estático	Todas las aplicaciones Android	Libre
SandDroid	Android	Estático y Dinámico	Todas las aplicaciones Android	Libre
Reverse.it	Android	Estático y Dinámico	Todas las aplicaciones Android	Libre
Dexter	Android Marshmallow	Dinámico	Aplicaciones Android compatibles con el sistema operativo.	Libre y Pagado
JavaDecompilers	Android	Estático	Todas las aplicaciones Android	Libre

A su vez, para tener todos los archivos necesarios para un correcto análisis de vulnerabilidades se requiere extraer las Apk de las aplicaciones de control remoto, en la Tabla 5 se muestra un conjunto de herramientas utilizadas para este fin.

Tabla 5. Herramientas para obtener las Apk

HERRAMIENTA	SISTEMA OPERATIVO	APLICACIONES QUE MANEJA	ACCESO ROOT
ML Manager	Android 4.1 o superior	Todas las aplicaciones Android Android	NO
APK Extractor	Android		NO
AppSend	Android		SI
APK Downloader	Android		NO

4.1 FASE 2: ANÁLISIS ESTÁTICO

Para la realización de las pruebas de vulnerabilidades en los dispositivos móviles se requieren los siguientes equipos:

- **Celular inteligente**
  - Sistema operativo: Android 4.4 o superior.
  - Almacenamiento interno: 2 GB.
  - Procesador: Dual Core de 1,2 GHz o superior.
  - Conectividad: Wi-Fi 802.

- **Computador personal**

  Para un correcto funcionamiento de la herramienta que ayuda al análisis de las aplicaciones se necesita como requerimientos mínimos:
  - Sistema operativo: Windows 7 o superior.
  - Ram: 4GB.
  - Procesador: Intel Core i5 o superior.
  - Disco duro: 256 GB o más.

De las herramientas mencionadas en las tablas 4 y 5, se escogió a MobSF y APK Extractor ya que permiten extraer la APK de cada aplicación de control remoto y todo su código fuente para poder realizar un correcto análisis y verificar cuales son los permisos, librerías, certificados, algoritmos y dominios peligrosos que puede contener cada una de éstas.

4.1.1 CONFIGURACIÓN DE LA HERRAMIENTA MOBSF PARA EL ANÁLISIS ESTÁTICO

Para la configuración de MobSF, se instaló Python2.7, el cual permite obtener las librerías necesarias para poder realizar el análisis de vulnerabilidades, utilizando el comando **python setup.py,** el cual permite configurar y descargar todos los paquetes dentro de esta aplicación.

Una vez finalizada la instalación, se configuró Python para habilitar todos los paquetes faltantes como por ejemplo: Django, pyOpenSSL, Tornado, entre otros, utilizando el comando: **python.exe –m pip install –r requirements.txt.**

Una vez instalado Python, se debe copiar en el disco **C** la carpeta de MobSF, abrir un nuevo símbolo del sistema y colocar el siguiente comando: **manage.py runserver,** el cual permite levantar la plataforma MobSF y conocer el puerto para conectarse al explorador como se muestra en la Figura 7.

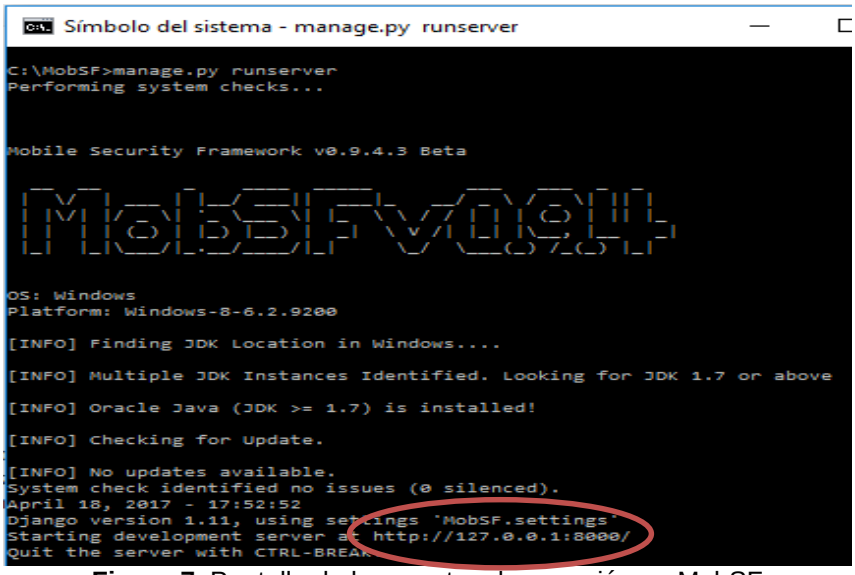


Figura 7. Pantalla de los puertos de conexión en MobSF

Después, instalar la herramienta APK Extractor en el dispositivo móvil para extraer las Apk de las aplicaciones de control remoto. En la Figura 8 se muestra la pantalla inicial en donde se puede observar todas las App que se encuentran instaladas en el celular.

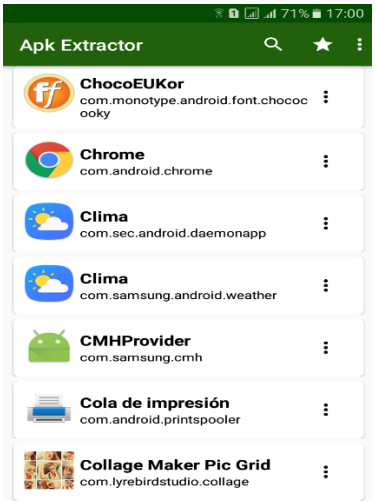


Figura 8. Pantalla principal de Apk Extractor

En la Figura 9 se puede observar cómo obtener las Apk, dando clic derecho en una de las App, el archivo se guarda dentro de la carpeta **ExtractedApks** del dispositivo móvil.

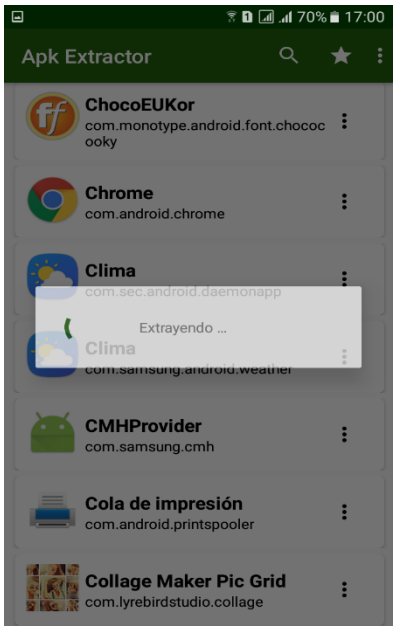


Figura 9. Pantalla de extracción de las APK

En la Figura 10 se muestra la pantalla principal de la herramienta MobSF, en donde se pueden subir los APK de las aplicaciones que van a ser analizadas.

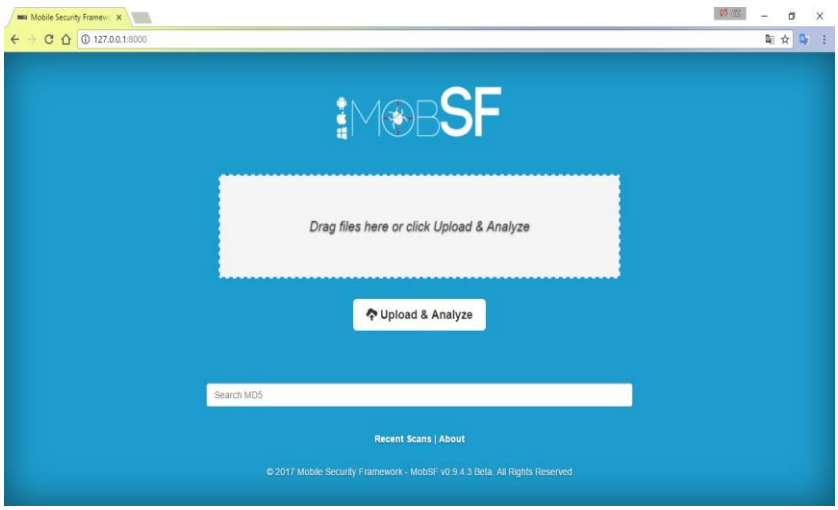


Figura 10. Pantalla de la interfaz de la herramienta MobSF

4.1.2 CASO DE ESTUDIO: APLICACIÓN TEAM VIEWER

La Figura 11 muestra la pantalla del código extraído de TeamViewer a través de la herramienta MobSF.

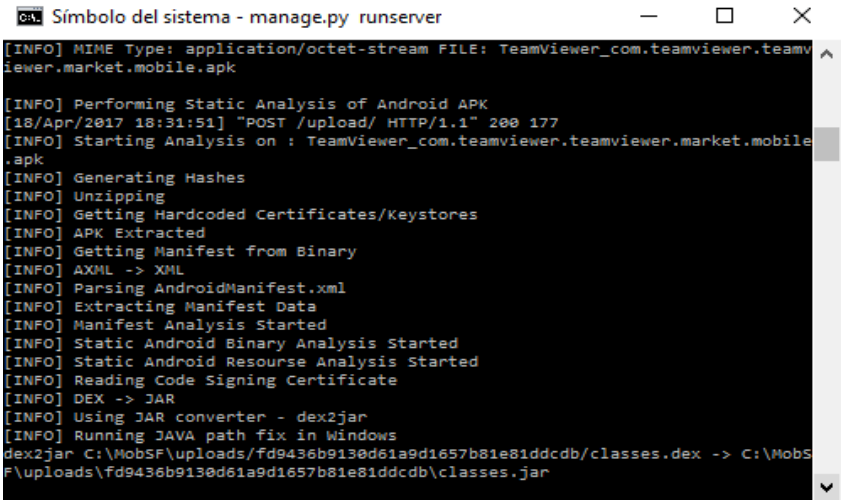


Figura 11. Pantalla de extracción de código de TeamViewer

La Figura 12 muestra la pantalla de la información general de la aplicación, en donde se puede observar el nombre de la App, su tamaño y el tipo de algoritmo que utiliza para los diferentes cifrados.

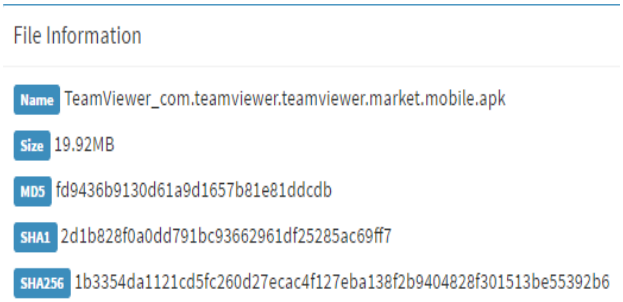


Figura 12. Pantalla de información general de TeamViewer

La Figura 13 muestra la pantalla del certificado de la aplicación, en donde se puede apreciar que el algoritmo SHA1 es vulnerable, ya que tiene

resistencia de colisión y a su vez no tiene una protección segura para la información.

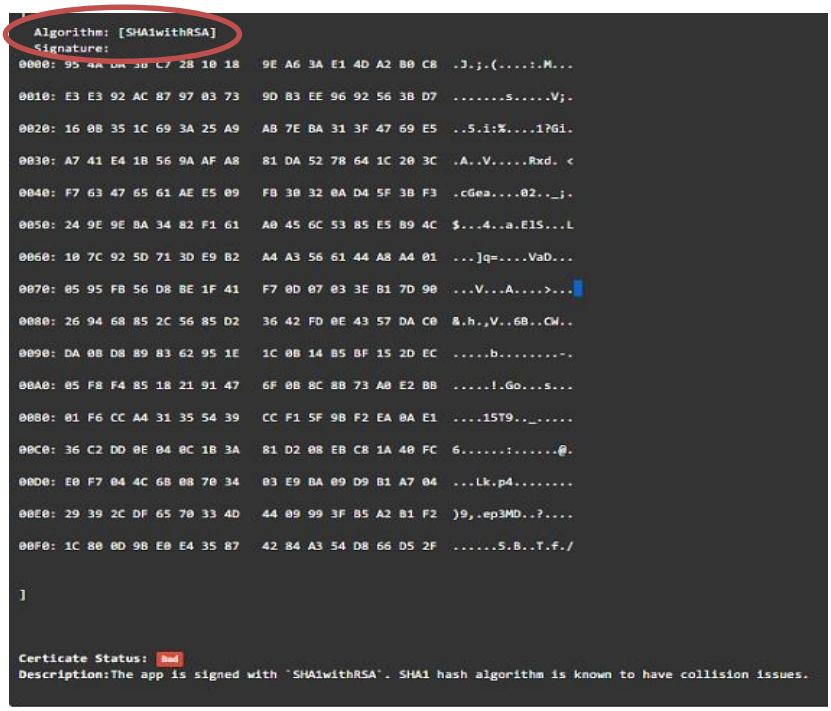


Figura 13. Pantalla del certificado de TeamViewer

En la Tabla 6 se muestran los permisos que necesita la aplicación para su funcionamiento con su respectiva descripción, indicando el nivel de vulnerabilidad de cada uno de ellos.

Tabla 6. Permisos de TeamViewer

PERMISOS	DESCRIPCIÓN	VULNERABILIDAD
android.permission.READ_PHONE_STATE	Permite que la aplicación acceda a las funciones del dispositivo.	Alto
(android.permission.ACCESS_WIFI_STATE	Permite que una aplicación vea la información sobre el estado de Wifi.	Bajo
android.permission.WAKE_LOCK	Permite que una aplicación impida que el teléfono se bloquee.	Alto
android.permission.ACCESS_NETWORK_STATE	Permite que una aplicación vea el estado de todas las redes.	Bajo
android.permission.INTERNET	Permite que una aplicación cree sockets de red.	Alto
com.teamviewer.teamviewer.market.mobile.permission.C2D_MESSAGE	Permite que la aplicación reciba notificaciones	Firma de la aplicación
android.permission.WRITE_EXTERNAL_STORAGE	Permite que una aplicación escriba en la tarjeta SD.	Alto

En la Tabla 7 se indican las características de la librería que puede perjudicar al dispositivo móvil al momento de utilizar la aplicación TeamViewer.

Tabla 7. Librería de la aplicación TeamViewer

PROBLEMA	SEVERIDAD	DESCRIPCIÓN
Estándar **ELF** encontrado construido sin la protección.	Alta	El desbordamiento de pila puede aumentar en datos almacenados en la pila de una función, porque obliga al atacante a obtener el control del puntero de instrucción sin darse cuenta el usuario.

En la Tabla 8 se muestran las características de los códigos maliciosos encontrados en esta aplicación.

Tabla 8. Análisis de código malicioso en TeamViewer

PROBLEMA	SEVERIDAD	DESCRIPCIÓN
(com.teamviewer.remotecontrollib.activity.ConnectInterfaceActivity) is not Protected.	Alta	Se comparte con otras aplicaciones en el dispositivo, por lo que es accesible a cualquier otra aplicación.
Se debe comprobar el nivel de protección del permiso (com.google.android.gms.gcm.GcmReceiver) Permission: com.google.android.c2dm.permission.SEND [android:exported=true]	Alta	Está protegido por un permiso que no está definido en la aplicación analizada. Como resultado, el nivel de protección del permiso debe comprobarse donde se define, una aplicación malintencionada puede solicitar y obtener el permiso e interactuar con el componente.
auk.java (Random random=new Random();while ((n2 = random.nextInt()) == 0));	Alta	La aplicación utiliza un generador de números aleatorios inseguro.
ix.java (var1_1 = Environment.getExternalStorageDirectory();)	Alta	Cualquier aplicación puede leer/escribir datos en almacenamiento externo.
MessengerCompat.java (int n2 = aup2.hashCode();)	Alta	Esta aplicación utiliza Java Hash Code y las operaciones criptográficas definidas en este paquete incluyen el cifrado, la generación y el acuerdo de claves.

La Figura 14 muestra los dominios con los cuales interactúa la aplicación y a su vez su estado conociendo de este modo si es bueno o peligroso dicho dominio.

Figura 14. Pantalla de dominios de TeamViewer

4.1.3 CASO DE ESTUDIO: APLICACIÓN SURE

En la Figura 15 se muestra la pantalla en donde se extrae todo el código de SURE por medio de la herramienta MobSF.

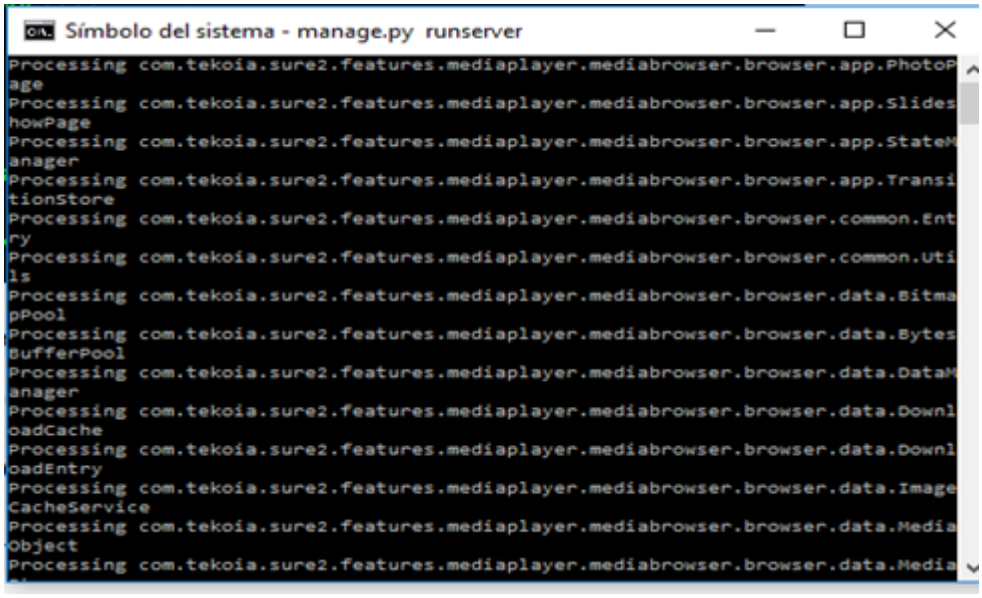


Figura 15. Pantalla de extracción del código SURE

En la Figura 16 se puede observar la pantalla con la información general de la aplicación, en donde se indica el nombre de la App, su tamaño y los algoritmos de encriptación que utiliza.

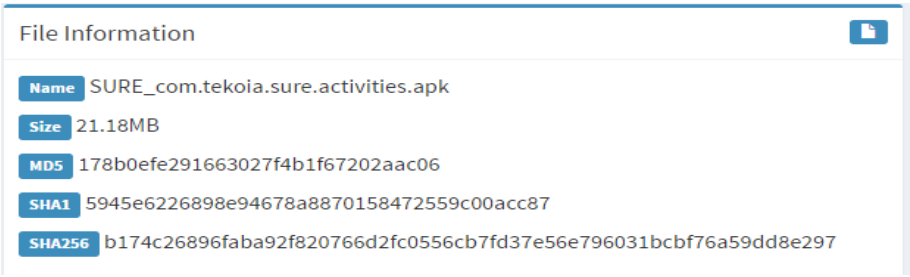


Figura 16. Pantalla de información general de Sure

La Figura 17 se muestra la pantalla con el certificado de la aplicación SURE, en donde se puede apreciar que el algoritmo SHA256 no es muy vulnerable, ya que calcula códigos únicos y no pueden decodificarse en una ingeniería inversa.

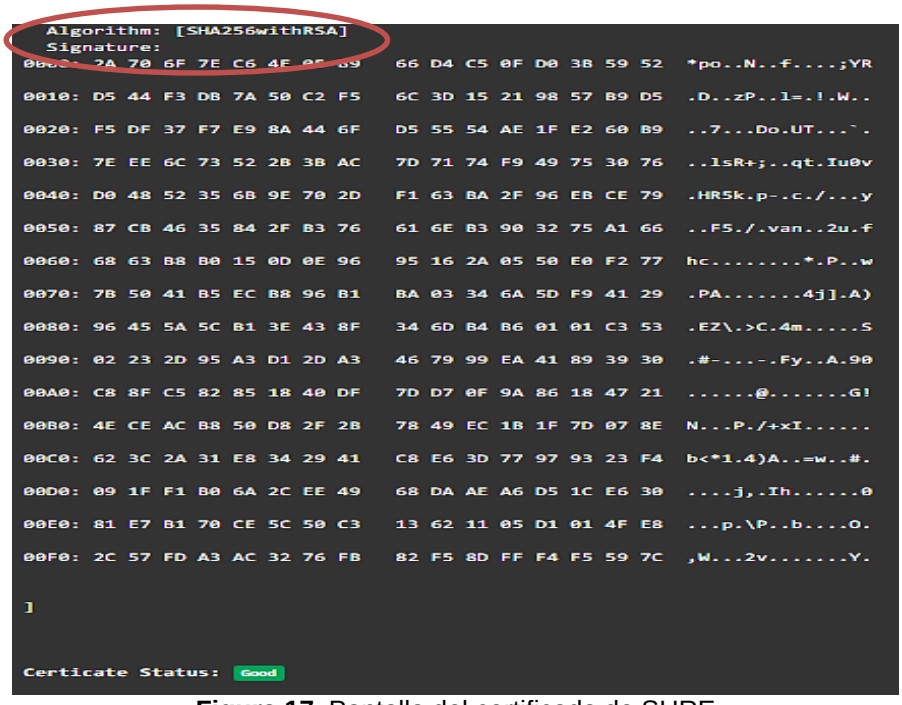


Figura 17. Pantalla del certificado de SURE

En la Tabla 9 se observan los distintos permisos que posee la aplicación con su respectiva descripción, indicando el nivel de vulnerabilidad.

Tabla 9. Permisos de la aplicación SURE

PERMISOS	DESCRIPCIÓN	VULNERABILIDAD
android.permission.ACCESS_FINE_LOCATION	Acceda a fuentes de localización como el Posicionamiento Global en el teléfono, ya que ocupa WIFI y está disponible en todo momento.	Alto
android.permission.READ_EXTERNAL_STORAGE	Permite que una aplicación lea la tarjeta SD.	Alto
android.permission.READ_PHONE_STATE	Una aplicación con este permiso puede determinar el número y la serie del teléfono.	Alto
com.tekoia.sure.activities.permission.C2D_MESSAGE	Permite que la aplicación reciba notificaciones	Firma o sello de la aplicación
android.permission.DISABLE_KEYGUARD	Permite que una aplicación inhabilite el bloqueo de teclas y cualquier seguridad de contraseña asociada.	Alto
android.permission.ACCESS_WIFI_STATE	Permite que una aplicación vea la información sobre el estado de Wifi.	Bajo
android.permission.ACCESS_COARSE_LOCATION	Acceda a fuentes de ubicación aproximada, como la base de datos de la red móvil.	Alto
android.permission.WAKE_LOCK	Permite que una aplicación impida que el teléfono se vaya a dormir.	Alto
android.permission.CHANGE_WIFI_MULTICAST_STATE	Permite que una aplicación reciba paquetes no directamente dirigidos a su dispositivo.	Alto
android.permission.CHANGE_WIFI_STATE	Permite que una aplicación se conecte y desconecte de puntos de acceso Wi-Fi y realice cambios.	Alto
android.permission.ACCESS_NETWORK_STATE	Permite que una aplicación vea el estado de todas las redes.	Bajo
android.permission.INTERNET	Permite que una aplicación cree sockets de red.	Alto

En la Tabla 10 se muestran las características de los códigos maliciosos encontrados en la aplicación SURE.

Tabla 10. Análisis de código malicioso de la aplicación SURE

PROBLEMAS	SEVERIDAD	DESCRIPCIÓN
Se pueden realizar copias de seguridad de los datos de la aplicación: [android:allowBackup=true]	Media	Permite a los usuarios que han habilitado la depuración USB copiar datos de la aplicación fuera del dispositivo.
(com.tekoia.sure2.features.mediaplayer.mediabrowser.browser.app.Browser) is not Protected. An intent-filter exists.	Alta	La presencia de filtros INTENT indica que la Actividad se exporta explícitamente.
(com.tekoia.sure.activities.Reopen Activity) is not Protected. [android:exported=true]	Alta	Se comparte con otras aplicaciones en el dispositivo, por lo que es accesible a cualquier otra aplicación del dispositivo
AuthenticationData.java (this.internalRouteAuthSecret = this.generateRandomSecret();)	Alta	La aplicación utiliza un generador de números aleatorios inseguro.
EnvironmentCompat.java (object = Environment.getExternalStorageState())	Alta	La aplicación puede leer / escribir en almacenamiento externo.
NanoHTTPD.java (var7_11 = this.tempFileManager.createTempFile();)	Alta	La información confidencial nunca debe ser escrita en un archivo temporal.
a.java (sslErrorHandler.proceed();;sslErrorHandler.cancel();	Alta	WebView ignora los errores del certificado SSL y acepta cualquier certificado SSL. Esta aplicación es vulnerable a ataques MITM

La Figura 18 indica la pantalla con los distintos dominios con los que interactúa la aplicación.

Figura 18. Pantalla de dominios de SURE

4.1.4 CASO DE ESTUDIO: APLICACIÓN PEEL SMART REMOTE

Para el análisis de está aplicación, se inicializa la herramienta MobSF para extaer el código fuente, como se indica en la Figura 19.

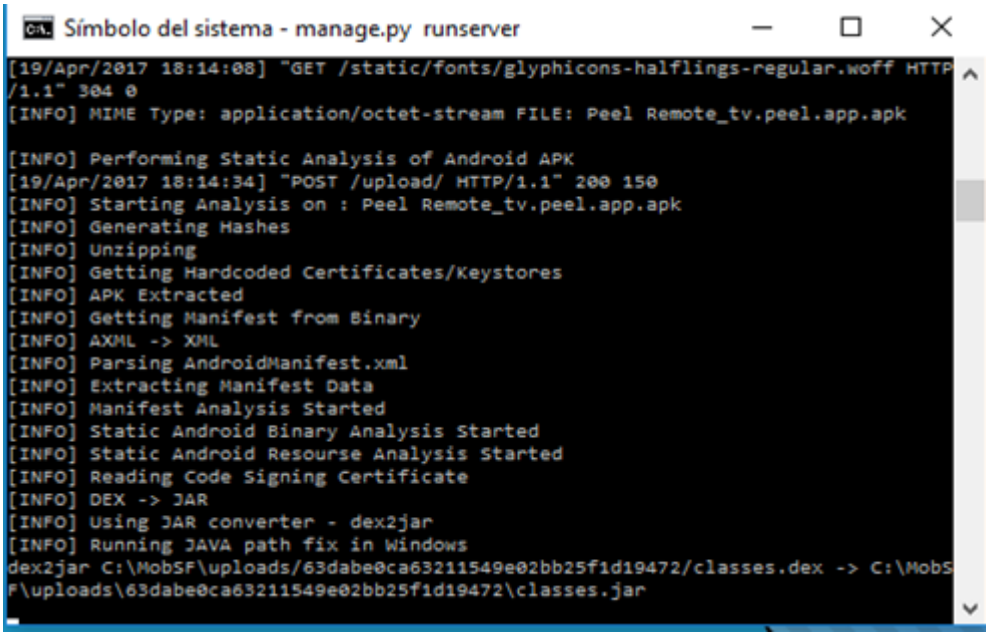


Figura 19. Pantalla de extracción de código de Peel Smart Remote

La Figura 20 muestra la pantalla con la información general de la aplicación, indicando el nombre de la App, su tamaño y el algoritmo de encriptación.

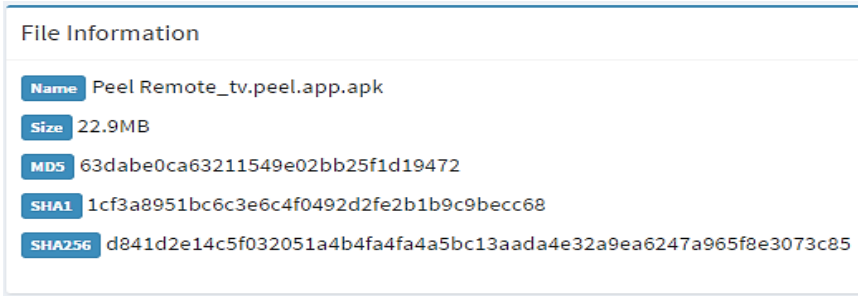


Figura 20. Pantalla de la información general de Peel Smart Remote

La Figura 21 indica la pantalla del certificado de la aplicación, en donde se puede apreciar que el algoritmo SHA1 es vulnerable ya que tiene resistencia

de colisión y no utiliza una protección segura de información clasificada y sensible.

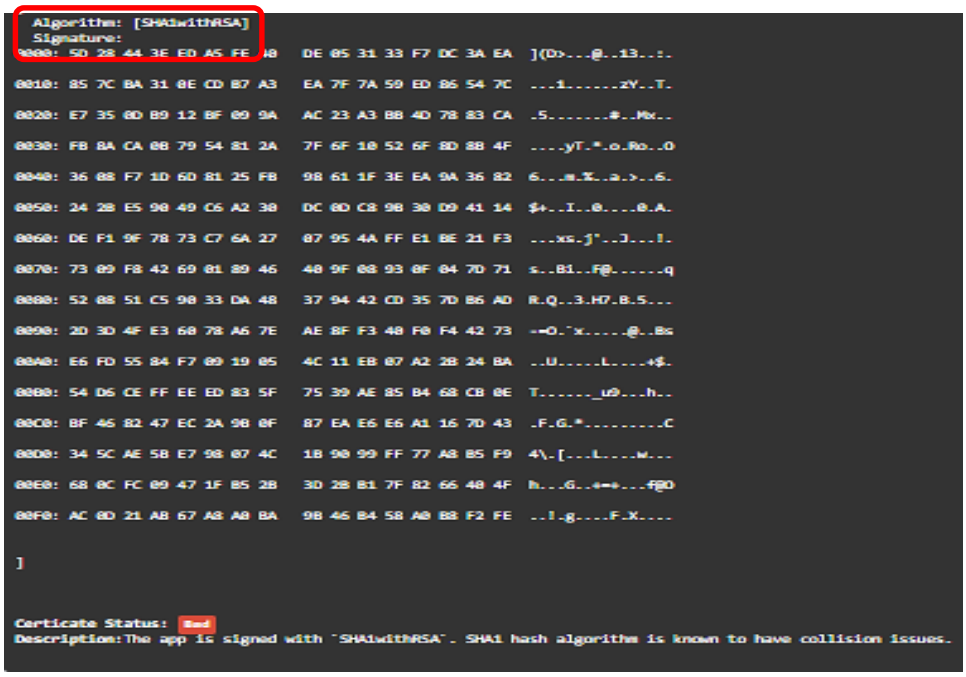


Figura 21. Pantalla del algoritmo de Peel Smart Remote

En la Tabla 11 se observan los distintos permisos que posee la aplicación PEEL SMART REMOTE con su respectiva descripción y estado.

Tabla 11. Permisos de la aplicación PEEL SMART REMOTE

PERMISOS	DESCRIPCIÓN	VULNERABILIDAD
android.permission.READ_CALENDAR	Permite que una aplicación lea todos los eventos del calendario almacenados en el teléfono.	Alto
android.permission.ACCESS_COARSE_LOCATION	Las aplicaciones maliciosas pueden usar esto para determinar aproximadamente dónde se encuentra.	Alto
android.permission.BLUETOOTH	Permite que una aplicación vea la configuración del Bluetooth, haga y acepte conexiones con dispositivos emparejados.	Alto
android.permission.WRITE_CALENDAR	Las aplicaciones maliciosas pueden utilizarlo para borrar o modificar los eventos del calendario o para enviar mensajes de correo electrónico a los invitados.	Alto
android.permission.INTERNET	Permite que una aplicación cree sockets de red.	Alto
android.permission.EXPAND_STATUS_BAR	Permite que la aplicación expanda o contraiga la barra de estado.	Bajo
android.permission.BLUETOOTH_ADMIN	Permite que una aplicación configure el Bluetooth del teléfono, descubra y empareje con dispositivos remotos.	Alto
android.permission.ACCESS_FINE_LOCATION	Acceda a fuentes de localización, como el Sistema de Posicionamiento en el teléfono, donde esté disponible.	Alto
android.permission.DEVICE_POWER	Permite que la aplicación encienda o apague el teléfono.	Firma de la aplicación
android.permission.ACCESS_NETWORK_STATE	Permite que una aplicación vea el estado de todas las redes.	Bajo
android.permission.GET_TASKS	Permite que aplicaciones maliciosas descubran información privada sobre otras aplicaciones que se ocupen actualmente o recientemente.	Alto
android.permission.WRITE_EXTERNAL_STORAGE	Permite que una aplicación escriba en la tarjeta SD.	Alto
android.permission.RECEIVE_BOOT_COMPLETED	Permite que una aplicación se inicie tan pronto como el sistema haya terminado de arrancar.	Bajo
android.permission.READ_PHONE_STATE	Una aplicación con este permiso puede determinar el número y serie del teléfono, una llamada está activa, etc.	Alto
tv.peel.app.permission.C2D_MESSAGE	Permite que la aplicación reciba notificaciones	Firma de la aplicación
android.permission.SYSTEM_ALERT_WINDOW	Las aplicaciones maliciosas pueden ocupar toda la pantalla del teléfono sin dejar que muestre las alertas del sistema	Alto
android.permission.WAKE_LOCK	Permite que una aplicación impida que el teléfono se bloquee.	Alto
android.permission.CHANGE_WIFI_STATE	Permite que una aplicación se conecte y desconecte de puntos de acceso Wi-Fi y realice cambios en las redes configuradas.	Alto
android.permission.RECORD_AUDIO	Permite que la aplicación acceda a la ruta de grabación de audio.	Alto
android.permission.READ_CONTACTS	Permite que una aplicación lea todos los datos de contacto almacenados en su teléfono y enviar sus datos a otras personas.	Alto
android.permission.MODIFY_AUDIO_SETTINGS	Permite que la aplicación modifique los ajustes globales de audio, como el volumen y el enrutamiento.	Alto

En la Tabla 12 se muestran las características de la librería que utiliza la aplicación Peel Smart Remote al momento de su ejecución.

Tabla 12. Librería de la aplicación PEEL SMART REMOTE

PROBLEMA	DESCRIPCIÓN	SEVERIDAD
Estándar **Elf** encontrado construido sin la protección.	El desbordamiento de pila puede aumentar en gran medida la dificultad de explotar un desbordamiento de almacenamiento de datos dentro de la aplicación.	Alta

En la Tabla 13 se muestran las características de los códigos maliciosos encontrados en esta aplicación.

Tabla 13. Análisis de código malicioso de PEEL SMART REMOTE

PROBLEMAS	SEVERIDAD	DESCRIPCIÓN
Utility.java (return"mounted".equals(Environment.getExternalStorageState())	Alta	La aplicación puede leer / escribir en almacenamiento externo
DefaultDiskStorage.java (object = new FileOutputStream(this.mTemporaryFile))	Alta	Se crea un archivo temporal para información confidencial, nunca debe ser escrita en un archivo temporal.
ReactDatabaseSupplier.java (SQLiteDatabase.execSQL("CREATE TABLE catalystLocalStorage (key TEXT PRIMARY KEY, value TEXT NOT NULL))	Alta	Los archivos pueden contener información confidencial codificada como nombres de usuario, contraseñas, claves, etc.

La Figura 22 indica la pantalla con los distintos dominios con los cuales interactúa la aplicación.

Figura 22. Pantalla de dominios de Peel Smart Remote

4.2 FASE 3: ANÁLISIS DINÁMICO

En esta fase se necesitan herramientas que ayuden a monitorear el comportamiento de las aplicaciones de control remoto. La Tabla 14 muestra las características de estos programas.

Tabla 14. Herramientas para monitorear las aplicaciones móviles

HERRAMIENTA	SISTEMA OPERATIVO	CARACTERÍSTICAS	TIPO DE APLICACIÓN
APP Use	Android	Permite realizar pruebas personalizadas	Todas las aplicaciones Android
Androl4b	Ubuntu	Utiliza máquina virtual mediante ingeniería inversa	Todas las aplicaciones Android
Android Malware Analysis Toolkit	Linux	Se puede realizar el análisis en línea sin instalar ninguna aplicación	Todas las aplicaciones Android
Inspeckage	Android	Inspecciona paquetes de Android como http, SQLite, ganchos, carpetas, etc	Todas las aplicaciones Android
StaDynA	Android	Análisis de aplicaciones de seguridad en presencia de características dinámicas de actualización de código	Todas las aplicaciones Android
NowSecure Lab Automated	Android o iOs	Herramienta empresarial para la prueba de seguridad de aplicaciones para móviles,	Aplicaciones Android o iOs

Del conjunto de herramientas mencionadas en la tabla anterior se seleccionó al programa Inspeckage porque permite verificar los distintos procesos que realiza las aplicaciones de control remoto, también se puede verificar los registro y archivo que se crean, Hashes, SQLite, HTTP, sistema de archivos, entre otros.

4.2.1 CONFIGURACIÓN DE LA HERRAMIENTA MOBSF PARA EL ANÁLISIS DINÁMICO

Para configurar el emulador de Android se debe importar en VirtualBox el archivo “MobSF_VM_03” como se indica en la Figura 23 y 24.

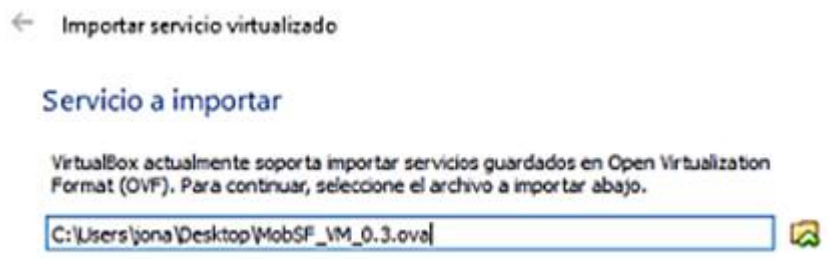


Figura 23. Selección de la máquina virtual MobSF

Figura 24. Importación de máquina virtual MobSF

Al finalizar la importación del emulador de Android, es necesario instalar tres aplicaciones que permiten configurar los permisos del dispositivo a nivel de super usuario, los mismos que se detallan a continuación:

- La aplicación **SuperSu** permite tener acceso al emulador como super usuario y tener acceso total a todos los recursos del dispositivo móvil. La Figura 25 muestra la pantalla para habilitar la opción "**EnableSuperuser**".

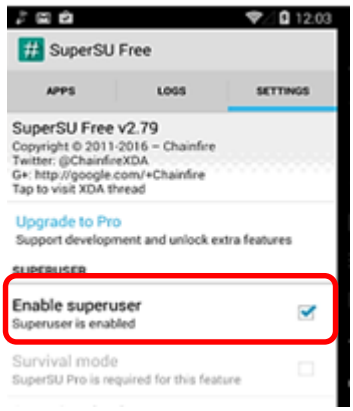


Figura 25. Instalación de la aplicación SuperSu

- La aplicación **Inspeckage**, monitorea las aplicaciones e indica la IP en la cual se va a verificar su ejecución, ver Figura 26.

Figura 26. Aplicación PEEL SMART REMOTE

- La aplicación **Xposed Framework** permite activar los módulos para el análisis dinámico, los mismos que ayudan a visualizar el código de la aplicación. La Figura 27 indica la pantalla para activar la última versión de Framework.

Figura 27. Aplicación Xposed Framework

Una vez activado Framework dar clic en el botón “Instal/Update” para instalar todos los módulos, como se muestra en la Figura 28.

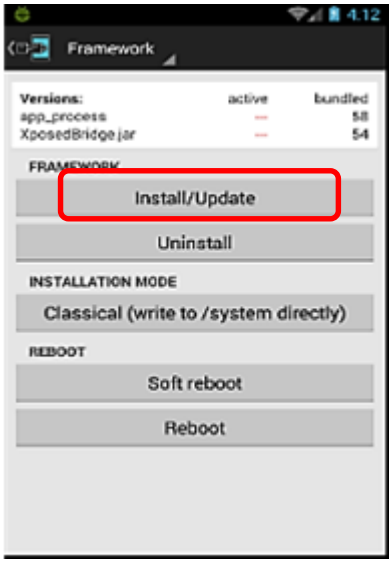


Figura 28. Instalación de Framework

Finalmente, en la Figura 29 se muestra la pantalla de los módulos que se deben activar para invocar a las aplicaciones instaladas anteriormente.

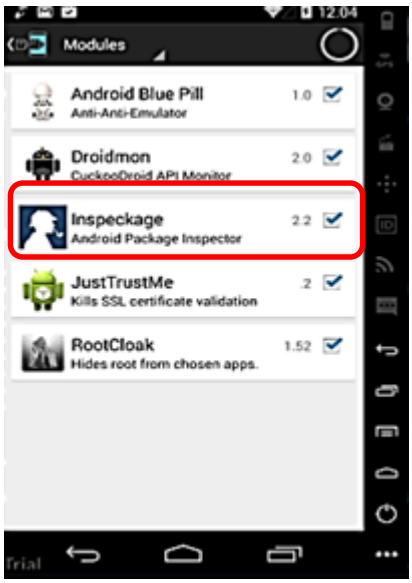


Figura 29. Módulos de Framework

4.2.2 CASO DE ESTUDIO: APLICACIÓN TEAM VIEWER

La Figura 30 muestra la pantalla en donde se carga la APK de la aplicación, obtenida a través del programa Apk Extractor disponible en el dispositivo móvil.

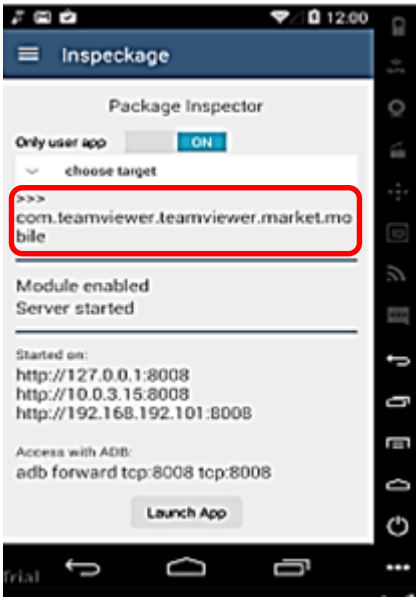


Figura 30. Carga de TeamViewer

La Figura 31 muestra la pantalla de las carpetas que se crean al momento de la ejecucion de Team Viewer, las cuales se utilizan para guardar información o los recursos necesarios para esta aplicación.

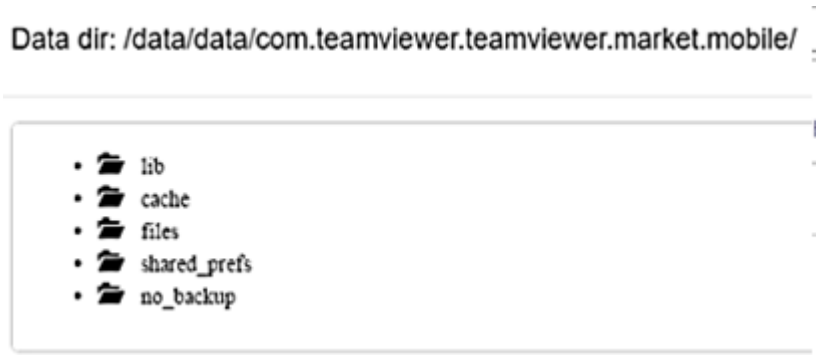


Figura 31. Pantalla de carpetas de TeamViewer

La Figura 32 muestra la pantalla de los get y put (requests) que utiliza Team Vieewer al momento de llamar a cada permiso o método de la aplicación.

Figura 32. Pantalla de los Get y Put de Team Viewer

La Figura 33 muestra la pantalla de los distintos algoritmos que se están ejecutando con TeamViewer, y que son necesarios para conocer la forma de encriptación de claves o usuarios.

Figura 33. Pantalla de los algoritmos de TeamViewer

La Figura 34 muestra la pantalla de los archivos que se crean o se modifican al momento de ejecutarse la aplicación en el dispositivo móvil.

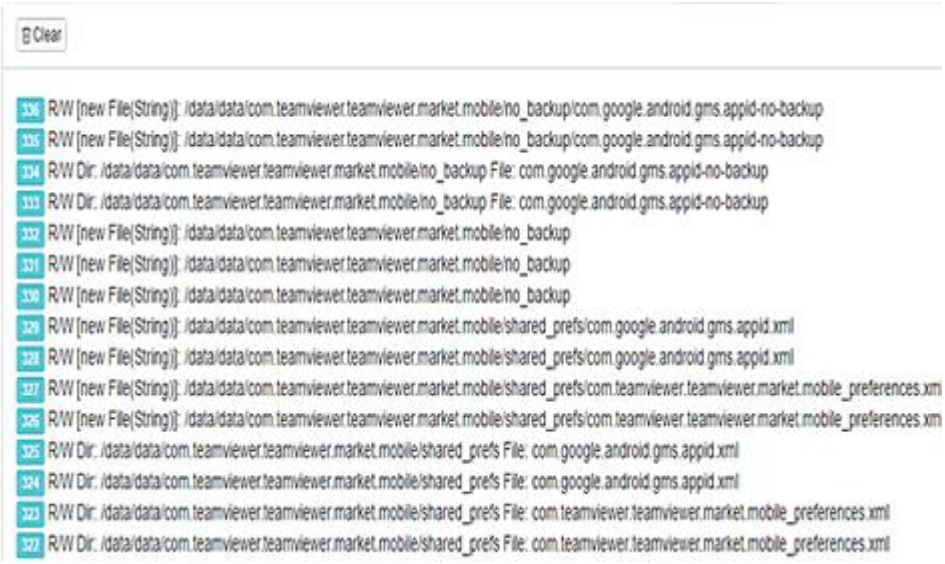


Figura 34. Pantalla de archivos que genera TeamViewer

La Figura 35 indica la pantalla de las actividades que inicia al momento de su ejecución.

Figura 35. Pantalla de información general de Team Viewer

4.2.3 CASO DE ESTUDIO: APLICACIÓN SURE

La Figura 36 muestra la pantalla en donde se carga el Apk de la aplicación de control remoto SURE, la cual se obtuvo por medio del programa Apk Extractor disponible en el dispositivo móvil.

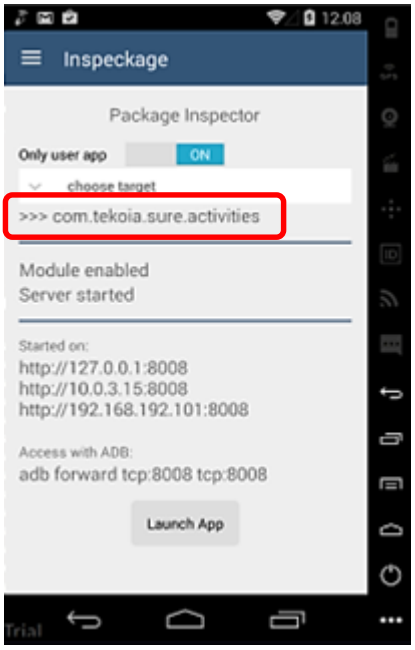


Figura 36. Carga de la aplicación SURE.

La Figura 37 indica la pantalla con los métodos get y put de los requests que utiliza la aplicación al momento de su ejecución.

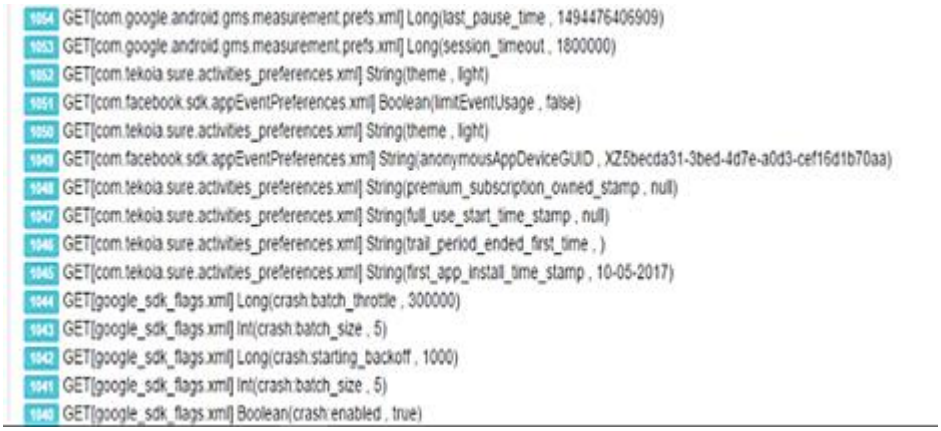


Figura 37. Pantalla de ejecución de métodos get y put de SURE

La Figura 38 muestra la pantalla con la dirección de las carpetas que accede la aplicación cuando necesita utilizar alguna actividad o recurso.

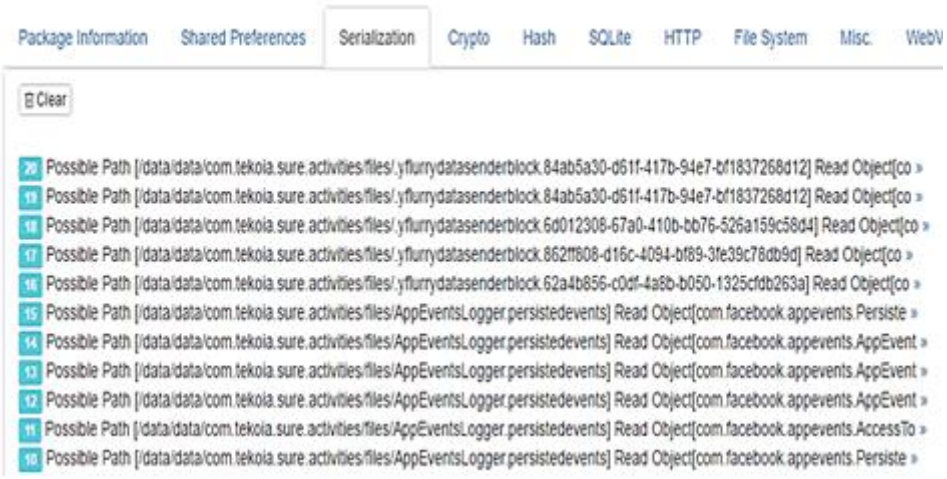


Figura 38. Pantalla de direcciones de carpetas que accede SURE

La Figura 39 muestra la pantalla de encriptación de las diferentes claves utilizando el esquema de cifrado AES al momento de acceder a los puntos Wi-FI.

Figura 39. Pantalla de encriptación de claves

La Figura 40 muestra la pantalla con los diferentes algoritmos que trabaja la aplicación cuando está en ejecución.

Figura 40. Pantalla de algoritmos de SURE

La Figura 41 muestra la pantalla con las sentencias SQL que realiza al momento de crear o llamar a las tablas para que la aplicación pueda funcionar correctamente.

Figura 41. Pantalla de sentencias SQL de SURE

La Figura 42 indica la pantalla con los HTTPs que la aplicación invoca cuando se realiza los diferentes procesos dentro de SURE.

Figura 42. Pantalla de HTTPs de SURE

En la Figura 43 se puede observar las carpetas o archivos de lectura y escritura a los cuales puede acceder la aplicación al momento de su ejecución.

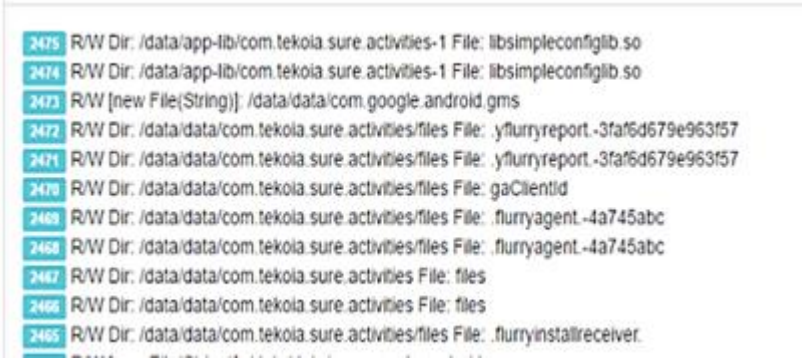


Figura 43. Pantalla de carpetas temporales de Sure

La Figura 44 muestra la pantalla de los URL que la aplicación llama cuando se esta ejecutando o realizando algún proceso.

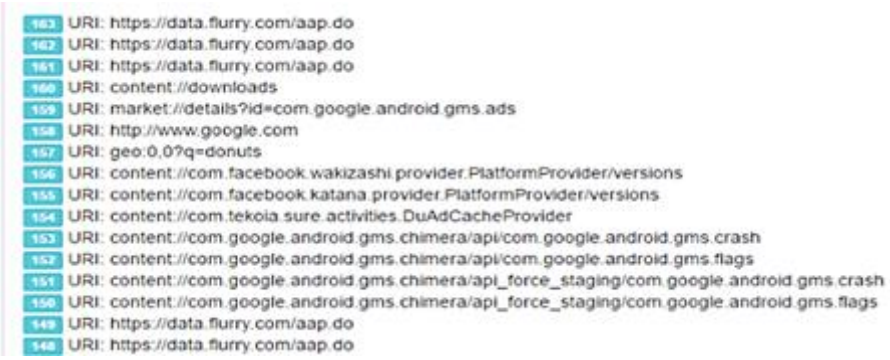


Figura 44. Pantalla de URLS de SURE

La Figura 45 indica la pantalla con los servicios que inicializa la aplicación para su correcta ejecución.

Figura 45. Pantalla de servicios de SURE

4.2.4 CASO DE ESTUDIO: APLICACIÓN PEEL SMART REMOTE

La Figura 46 muestra la pantalla con el Apk de la aplicación para iniciar el análisis dinámico, obtenido a través de la aplicación Apk Extractor que se encuentra disponible en el dispositivo móvil.

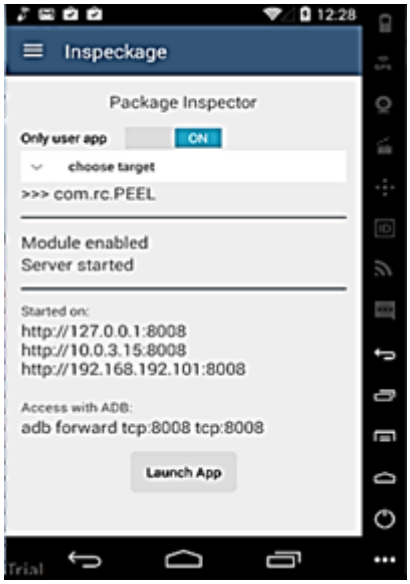


Figura 46. Carga del Apk de Peel Smart Remote

La Figura 47 indica las carpetas en donde se guarda la información o se leen los archivos necesarios para el funcionamiento de esta aplicación.

Figura 47. Pantalla de las carpetas de Peel Smart Remote

La Figura 48 muestra la pantalla de los métodos get y put de los requests, que utiliza la aplicación.

Figura 48. Pantalla de los métodos get y put de Peel Smart Remote

La Figura 49 muestra la pantalla de la encriptación de las claves mediante el esquema de cifrado AES al momento de utilizar ingeniería inversa.

Figura 49. Pantalla de encriptación de claves de Peel Smart Remote

La Figura 50 muestra la pantalla con los algoritmos que utiliza la aplicación al momento de su ejecución.

Figura 50. Pantalla de los algoritmos de Peel Smart Remote

La Figura 51 muestra los HTTPs que utiliza la aplicación cuando se esta ejecutando un proceso o necesita un servicio.

Figura 51. Pantalla de HTTPs de Peel Smart Remote

La Figura 52 muestra la pantalla de las direcciones de las carpetas que crea o a las cuales accede la aplicación.

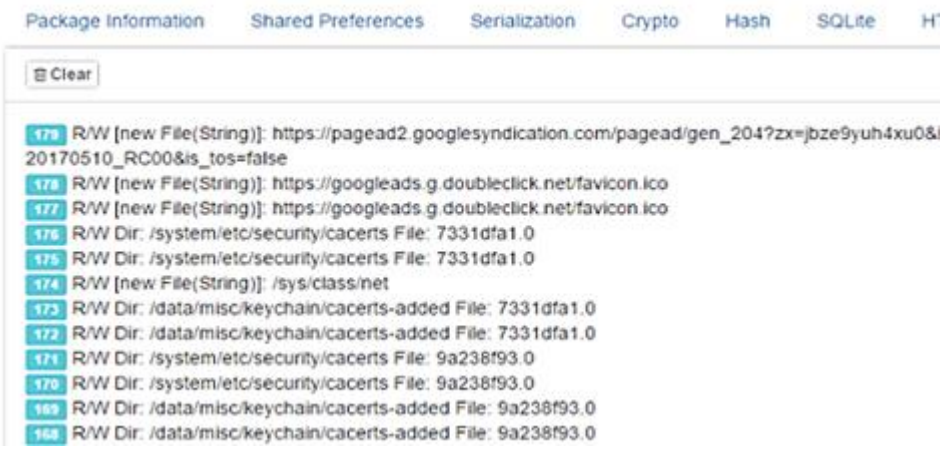


Figura 52. Pantalla de direcciones de las carpetas de Peel Smart Remote

La Figura 53 muestra la pantalla con los URL que utiliza al momento de llamar a un servicio.

Figura 53. Pantalla de URL de Peel Smart Remote

La Figura 54 muestra la pantalla de los servicios o las clases que inicia la aplicación al momento de su ejecución.

Figura 54. Pantalla de los servicios de Peel Smart Remote

4.3 INFORME DE LOS RESULTADOS OBTENIDOS DE LAS APLICACIONES ANALIZADAS.

La Tabla 15 muestra un cuadro comparativo de las vulnerabilidades obtenidas en las diferentes aplicaciones de control remoto, de acuerdo a la metodología de OWASP.

Tabla 15. Vulnerabilidades de las aplicaciones analizadas

VULNERABILIDAD / APLICACIÓN	TEAMVIEWER	SURE	PEEL SMART TV
Uso inadecuado de la plataforma.	**Medio**, impide que el teléfono se bloquee.	**Medio**, impide que el teléfono se bloquee.	**Medio**, impide que el teléfono se bloquee.
Almacenamiento de datos inseguros.	**Alto**, utiliza java hash code. Accede a otras aplicaciones y a la tarjeta SD	**Alto**, Permite que escriba en la tarjeta SD y ver la ubicación del dispositivo	**Alto** , permite que los datos guardados en la aplicación se puedan descifrar en ingeniería inversa
Comunicación insegura.	**Bajo**, solo ve los estados Wi-Fi pero no se conecta	**Medio**, permite que realice cambios o se conecte automáticament e a los puntos Wi-Fi.	**Alto**, empareja con dispositivos desconocidos mediante Bluetooth y Wi-Fi
Autentificación insegura.	**Alto**, tiene cifrado al generar claves	**Alto** , acepta cualquier permiso SSL o guarda información en carpetas temporales	**Alto**, crea carpetas temporales para claves contraseñas o usuarios.
Criptografía insuficiente.	**Alto** utiliza algoritmos md5	**Medio** , utiliza algoritmos Sha256	**Alto**, utiliza algoritmos md5
Autorización insegura.	**Alto** , accede al dispositivo y a la memoria SD, puede modificar u obtener información	**Alto** , accede a toda la información del celular y puede copiar datos para enviar a otras personas	**Alto** , accede a la localización, permite escribir en la tarjeta SD, no permite que muestre alertas
Calidad del código deficiente	**Bajo**, permite interactuar con aplicaciones que se ejecuten en ese momento.	**Alto**, información confidencial es escrita en carpetas temporales	**Alto** , permite que reciba paquetes que no son para ese dispositivo
Codificación de Código.	**Bajo,** Utiliza algoritmos para descifrar las claves o para su ejecución	**Medio**, utiliza algoritmos adecuados para no cifrar las claves en su ejecución	**Alto**, utiliza algoritmos para descifrar las claves o para su ejecución
Ingeniería Inversa.	Se obtuvo todo el código por medio de la herramienta MobSF		
Funcionalidad extraña.	No se puede comprobar La aplicación se descarga cuando está en producción		

En estudios revisados sobre las vulnerabilidades en dispositivos móviles se pudo observar que se utiliza un análisis estático y/o dinámico en las diferentes aplicaciones que existen para el sistema operativo Android comprobando de este modo los diferentes permisos o la información a la cual se puede acceder sin la autorización de los usuarios. A su vez se implementan algoritmos basados en frameworks que tienen como función predecir si la aplicación analizada es benigna o maligna de acuerdo a los permisos configurados en el archivo AndroidManifest, y de esta manera replicar y comparar la información que se filtran por medio de las apps.

Por lo mencionado anteriormente, se hizo un análisis de las vulnerabilidades con la ayuda de herramientas Mobsf que permitio extraer todo el código real de las aplicaciones TeamViewer, Sure y Peel Smart Remote, con la implementación del análisis estático, en donde se pudo comprobar que existen varias amenazas para los usuarios como por ejemplo guardando claves, usuarios, imágenes, etc., en archivos temporales los mismos que al utilizar ingeniería inversa se pueden descifrar con facilidad. Por otro lado, en el análisis dinámico se puede observar cómo se encriptan las contraseñas, los permisos que utiliza la aplicación y los métodos a los que llama para realizar un determinado proceso.

Concluyendo, tanto el análisis estático y dinámico es de mucha ayuda para analizar las vulnerabilidades que pueden existir en las diferentes aplicaciones, ya que de esta manera se puede conocer el funcionamiento real de las apps y a su vez poder proteger la información que se almacena dentro de los dispositivos móviles.

5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- El uso de dispositivos móviles, por ser una herramienta imprescindible en la vida de las personas, ha incrementado la creación de aplicaciones sin una supervisión adecuada, lo cual provoca el aumento de ataques a los celulares.

- Con el análisis estático realizado se comprobó que las aplicaciones TeamViewer, Sure y Peel Smart Remote tienen acceso a la información que contiene el celular y pueden acceder con facilidad a otras aplicaciones, mientras que en el análisis dinámico se pudo observar en tiempo real las carpetas que se crean, los algoritmos de encriptación y los permisos de acceso.

- De los resultados obtenidos en el análisis estático y dinámico de la aplicación Team Viewer se pudo observar que esta App es confiable, ya que permite tener contraseñas para poder asociarse con otro dispositivo y a su vez controlar los accesos de control remoto desde la misma aplicación.

- Del mismo modo al finalizar el análisis estático y dinámico de la aplicación Peel Smart Remote se pudo concluir que es perjudicial para los dispositivos móviles, ya que recibe direccionamiento de paquetes de los cuales no se conoce su procedencia y la encriptación de las claves lo realiza en un texto plano, permitiendo de este modo que terceras personas puedan acceder fácilmente al dispositivo sin que el usuario se de cuenta.

5.2 RECOMENDACIONES

- Habilitar los diversos tipos de comunicación como Wi-fi, bluetooth o Gps, solo cuando se los utilice o se conozca la red a la cual se van a vincular, ya que algunas aplicaciones utilizan esto para filtrar o enviar información sin que los usuarios se den cuenta.

- Tener las aplicaciones actualizadas para solucionar vulnerabilidades de seguridad y no sufrir ataques en el dispositivo, haciéndolo desde un sitio oficial o plataforma digital de aplicaciones como Google Play.

- Antes de descargar e instalar las aplicaciones, se debe investigar qué datos del usuario se utilizan y para qué fin, las políticas de acceso y acerca de posibles acuerdos con terceros.

- No otorgar permisos innecesarios o excesivos al momento de instalar una app o cuando la aplicación necesite realizar una determinada funcionalidad, ya que se debe explicar claramente los motivos por los que necesita cada permiso.

6 BIBLIOGRAFÍA

Acevedo, J. (2016). Convertir tu Android en un control remoto universal. *Movilarena*.

Agustín, R., & Carlos, L. (2013). *Descripción y análisis del modelo de seguridad de Android.* USA: UR. FI – INCO.

Albors, J. (2014). Cómo explotar vulnerabilidades en dispositivos móviles a escala global. *WeliSecurity*, 1,2.

Altarejos, C. G. (2017). *Seguridad en Smartphones: Análisis de riesgos de vulnerabilidades y auditorías de dispositivos.* Barcelona: Alba.

Casas, E. (2017). COMPRUEBA LAS VULNERABILIDADES DE TU SMARTPHONE ANDROID CON VTS FOR ANDROID. *ADLSZone*.

Chell, D., Erasmus, T., Colley, S., & Whitehouse, O. (2015). *The Mobile Application Hacker's Handbook.* Indianapolis.

Christian Camilo Urcuqui López, A. N. (2016). *Clasificadores de machine learning para análisis de malware en Android.* Colombia.

Dalziel, H., & Abraham, A. (2016). *Automated Security Analysis of Android and iOS Applications with Mobile Security Framework.* New York: Elsevier; Book Aid.

ESET Latinoamérica. (2014). Tendencias 2014: El desafío de la privacidad en Internet. 9,10.

ESET Latinoamérica. (2016). Eset Security Report Latinoamérica 2016.

ESET Latinoamérica. (2016). Tendencias 2016 Security Everywhere. *ESET LATINOAMERICA*.

Foresti, S., Yung, M., & Martinelli, F. (2012). *Computer Security – ESORICS 2012.* Italia: Springer.

Herrera, J. (2015). APK Extractor. *Movil Zona*, 1,2.

Ibáñez, L. d., & Navarro, J. M. (2014). *Seguridad nacional, amenazas y respuestas.* España: LID.

Jaramillo, J. (2016). Peel Smart Remote, Un control para TV en tu móvil. *RWWES*.

Kruegel, C. (2008). Limits of Static Analysis for Malware Detection. *IEEE Xplore Digital Library*.

Lahaie, C., & Leberfinger, D. (2013). TeamViewer Forensics. *IEEE Explorer*.

Laudon, K., & Laudon, J. (2014). *Sistemas de información gerencial: administración de la empresa digital.* Mexico: D.R.

Lech, J., JanczewskiHenry, B., & WolfeSujeet, S. (2013). *An Empirical Evaluation of the Android Security Framework.* New Zealand: Springer.

Li, G., Holtkamp, B., Wang, J., & Han, Y. (2009). OASAM - AN OPEN AND ADAPTIVE SOFTWARE ARCHITECTURE MODEL FOR SERVICE-ORIENTED APPLICATIONS IN DYNAMIC NETWORK ENVIRONMENTS. *IOS Press*. Obtenido de OASAM: metodología para el análisis de seguridad de aplicaciones Android.

Llaven, D. S. (2015). *Panorama para ingenieria en computacion e informatica.* Mexico: Patria S.A.

Pérez, B. S. (2015). *Sistemas Operativos Monopuestos.* LULU Press.

Romano, A., & Luna, C. (2013). *Descripción y análisis del modelo de seguridad de Android .* Uruguay: Pedeciba.

Staff, P. (2016). 14 predicciones sobre seguridad informática que se podrían avecinar en 2017. *PCWorld*.

Totzek-Hallhuber, J. (2017). Neue Risiken: OWASP verschlimmbessert Liste. *Silicon*.

Urquijo, B. S. (2012). *Un Nuevo Enfoque para la Seguridad en las aplicaciones de Smartphones.* Bilbao: Akal.