**UNIVERSIDAD POLITÉCNICA DE MADRID**

****

**MÁSTER UNIVERSITARIO EN CIBERSEGURIDAD**

**TRABAJO FIN DE MÁSTER**

***Desarrollo de mecanismos para auditoría de tráfico de aplicaciones Android***

**Andrea del Nido García**

**2020**

**TRABAJO FIN DE MÁSTER**

**TÍTULO:** Desarrollo de mecanismos para auditoría de tráfico de aplicaciones Android

**AUTOR:** D. Andrea del Nido García

**TUTOR 1/DIRECTOR:** D. José María del Álamo Ramiro

**Escuela Técnica Superior de Ingenieros de Telecomunicación**

**Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación**

**Escuela Técnica Superior de Ingenieros Informáticos**

**Escuela Técnica Superior de Ingeniería de Sistemas Informáticos**

**TRIBUNAL:**

**PRESIDENTE:**

**VOCAL:**

**SECRETARIO:**

**FECHA DE LECTURA:**

**CALIFICACIÓN:**

Fdo: El Secretario del Tribunal

Agradecimientos

Opcional

Resumen

Cada día más y más usuarios disponen de teléfono móviles, y hoy en día se asocia a las personas con números de teléfono. Todo este gran desarrollo de tecnología conlleva a que, como bien es sabido por todos, nuestros móviles sean algo fundamental en todo momento. La mayor parte de datos personales de los usuarios se encuentran en sus dispositivos móviles. Es por esto por lo que es realmente necesario definir unos protocolos de seguridad claros y concisos, que permitan poder hacer uso de estas tecnologías sin temer ataques a la privacidad, robo de identidad, etc.

Cada vez hay más y más aplicaciones disponibles en nuestros teléfonos móviles que hacen que nuestra vida sea algo más sencilla o incluso más entretenida en ciertos momentos. Ahora bien, es cierto que dichas aplicaciones hacen uso de datos que se almacenan en los terminales, como, por ejemplo, posicionamiento GPS, acceso a cámara, contactos, entre otros. Solicitando permiso al usuario claro, pero, ¿quién va a negar el permiso de la cámara a Candy Crash?, si no lo aceptas entonces no te será permitido hacer uso del juego.

El presente trabajo consiste en el análisis de aplicaciones Android, concretamente, el análisis y estudio de los mecanismos de seguridad que implementan las aplicaciones, centrado en el análisis de tráfico, además de la implementación y desarrollo de mecanismos capaces de sobrepasar dichos mecanismos de seguridad, ya que, como se ha comentado previamente, si algún atacante fuera capaz de interceptar las comunicaciones o suplantar las aplicaciones sería capaz de adueñarse de la información del usuario.

Abstract

Every day more and more users have mobile phones, and nowadays people are associated with phone numbers. All this great development of technology means that, as is well known by all, our mobiles are essential at all times. Most of the personal data of users is found on their mobile devices. This is why it is really necessary to define clear and concise security protocols that allow you to make use of these technologies without fear of attacks on privacy, identity theft, etc.

There are more and more applications available on our mobile phones that make our lives a little easier or even more entertaining at certain times. However, it is true that these applications make use of data that is stored in the terminals, such as, for example, GPS positioning, camera access, contacts, among others. Requesting permission from the user of course, but who is going to deny the permission of the camera to Candy Crash? If you do not accept it then you will not be allowed to use the game.

The present work consists of the analysis of Android applications, specifically, the analysis and study of the security mechanisms that the applications implement, focused on traffic analysis, in addition to the implementation and development of mechanisms capable of bypassing said security mechanisms, since, as previously commented, if an attacker were able to intercept communications or impersonate applications, they would be able to take over user information.

Contenido

[Agradecimientos 2](#_Toc38025002)

[Resumen 3](#_Toc38025003)

[Abstract 4](#_Toc38025004)

[Contenido 5](#_Toc38025005)

[Listado de figuras 7](#_Toc38025006)

[Listado de tablas 8](#_Toc38025007)

[Acrónimos 9](#_Toc38025008)

[Definiciones previas 10](#_Toc38025009)

[1 Introducción 13](#_Toc38025010)

[1.1 Introducción 13](#_Toc38025011)

[1.2 Objetivos 14](#_Toc38025012)

[2 Estado del arte 17](#_Toc38025013)

[2.1 Introducción 17](#_Toc38025014)

[2.2 Estructura de certificados 17](#_Toc38025015)

[2.3 Mecanismos de seguridad del tráfico en Android 18](#_Toc38025016)

[2.3.1 Comunicación TLS/SSL 18](#_Toc38025017)

[2.3.2 Certificate pinning 18](#_Toc38025018)

[2.4 Captura de tráfico 20](#_Toc38025019)

[2.5 Análisis de las aplicaciones Android 20](#_Toc38025020)

[3 Desarrollo 21](#_Toc38025021)

[3.1 Herramientas 21](#_Toc38025022)

[3.2 Entorno 23](#_Toc38025023)

[3.3 Plataforma 25](#_Toc38025024)

[3.4 Búsqueda de bibliotecas de certificate pinning 25](#_Toc38025025)

[3.4.1 Ejecución del sistema de detección de bibliotecas 27](#_Toc38025026)

[3.4.2 Ficheros obtenidos como resultado de la ejecución 29](#_Toc38025027)

[3.5 Implementación 30](#_Toc38025028)

[3.5.1 Caso 1: certificate pinning con paquetes 31](#_Toc38025029)

[3.5.2 Caso 2: certificate pinning X 31](#_Toc38025030)

[4 Resultados 32](#_Toc38025031)

[5 Conclusiones y líneas futuras 33](#_Toc38025032)

[Apéndices y/o anexos 34](#_Toc38025033)

[Bibliografía 35](#_Toc38025034)

Listado de figuras

[Figura 2.1. Esquema de funcionamiento certificate pinning 19](file:///C:\Users\Andrea\Desktop\Andrea\Máster%20Ciberseguridad%20UPM\TFM\Memoria\Memoria_TFM_MUCS_Andrea_del_Nido.docx#_Toc38021153)

[Figura 3.1. Esquema del entorno de trabajo 24](file:///C:\Users\Andrea\Desktop\Andrea\Máster%20Ciberseguridad%20UPM\TFM\Memoria\Memoria_TFM_MUCS_Andrea_del_Nido.docx#_Toc38021154)

[Figura 3.2. Esquema del sistema de búsqueda de bibliotecas con certificate pinning 26](file:///C:\Users\Andrea\Desktop\Andrea\Máster%20Ciberseguridad%20UPM\TFM\Memoria\Memoria_TFM_MUCS_Andrea_del_Nido.docx#_Toc38021155)

[Figura 3.3. Ejecución del script pinning\_libraries.sh 27](file:///C:\Users\Andrea\Desktop\Andrea\Máster%20Ciberseguridad%20UPM\TFM\Memoria\Memoria_TFM_MUCS_Andrea_del_Nido.docx#_Toc38021156)

[Figura 3.4. Ejecución del script find\_libraries\_pinning.sh 28](file:///C:\Users\Andrea\Desktop\Andrea\Máster%20Ciberseguridad%20UPM\TFM\Memoria\Memoria_TFM_MUCS_Andrea_del_Nido.docx#_Toc38021157)

[Figura 3.5. Ejecución del script check\_pinning\_library.py 28](file:///C:\Users\Andrea\Desktop\Andrea\Máster%20Ciberseguridad%20UPM\TFM\Memoria\Memoria_TFM_MUCS_Andrea_del_Nido.docx#_Toc38021158)

Listado de tablas

[Tabla 1.1. Valores típicos en función del tipo de dispositivo 3](#_Toc527920810)

Acrónimos

A continuación, se listan los acrónimos empleados a lo largo del desarrollo del trabajo.

|  |  |
| --- | --- |
| **HTTP** | Hypertext Transfer Protocol |
| **SSL/TLS** | Secure Socket Layer/Transport Layer Security |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **DHCP** | Dynamic Host Configuration Protocol |
| **CA** | Certification Authority |
| **PKI** | Public Key Infrastructure |
| **MiTM** | Man in The Middle |
| **APK** | Android Package |

Definiciones previas

Se resumen a continuación las definiciones de conceptos utilizados a lo largo del trabajo desarrollado.

**Modelo cliente-servidor:** este tipo de arquitectura se basa en que los servicios y recursos de un determinado software son proporcionados por un servidor y, aquellas entidades que demandan estos servicios y recursos son los clientes.

**HTTP:** es un protocolo de comunicación que permite la transferencia de datos e información comúnmente utilizado en la World Wide Web (www). En la capa de transporte es soportado por el protocolo TCP y en el nivel de red por el IP. Este protocolo sigue un modelo cliente-servidor. El cliente realiza peticiones al servidor (request) y el servidor proporciona recursos y servicios contestando mediante una respuesta (response). Este protocolo tiene definidos varios métodos para poder realizar las peticiones y en las respuestas se incluyen códigos para verificar el estado de la comunicación. El puerto estándar para este protocolo es el 80.

**HTTPS:** protocolo HTTP pero que utiliza el cifrado del protocolo SSL/TLS en las comunicaciones.El protocolo funciona igual que HTTP, pero l diferencia es que al cifrar las comunicaciones ya no se pueden realizar ataques estilo MiTM ya que al interceptar las tramas HTTPS estarían cifradas.

**DHCP:** por sus siglas en inglés, *Dynamic Host Configuration Protocol* [1], es un protocolo de red cuya arquitectura de red es el modelo cliente servidor, el cual tiene como función asignar direcciones IP a dispositivos de forma dinámica, es decir, sin que la dirección IP tenga que ser fijada por un usuario. Para ello, se dispone de una lista de direcciones DHCP que se van asignando en base al uso de cada una de ellas. Además, se almacenan ciertos parámetros de configuración necesarios para poder llevar a cabo de forma satisfactoria esta asignación de direcciones, algunos parámetros almacenados son, por ejemplo, la máquina a la que se asigna la dirección, la IP asignada, la duración de la asignación de dicha dirección, entre otros parámetros.

**SSL/TLS:** SSL es un protocolo predecesor del protocolo TLS. Estos protocolos criptográficos consisten en proporcionar confidencialidad e integridad en las comunicaciones en una red de comunicación. Tratan de garantizar que la información que es transmitida por un canal no sea modificada ni interceptada y, que los integrantes de la comunicación puedan acceder a ella. Tal y como se muestra en [2], estos protocolos son muy utilizados para la seguridad de las comunicaciones web, pero para la situación de este proyecto, hay que tener en cuenta que para dispositivos Android se utilizan estos protocolos para la comunicación cifrada.

**Certificados digitales:** los certificados en este contexto, hacen referencia a mecanismos o utilidades a partir de las cuales se puede confirmar la autenticidad de la información. Además, también se consigue mantener la integridad del mensaje y el no repudio. Este mecanismo utiliza un par de claves, la clave privada conocida únicamente por el propio usuario y, la clave pública, que es conocida por todos.

**Autoridad de certificación:** son autoridades reconocidas las cuales se encargan de generar y mantener los certificados digitales ya que de alguna manera se tienen que comprobar que sean válidos. Para saber si los certificados son válidos, se emplean cadenas de confianza de tal manera que atravesando certificados se llega al certificado raíz, y este certificado es el que ha sido generado directamente por una autoridad de certificación.

**Ataques Man in The Middle:** este ataque se basa en que el atacante se sitúa en medio de la comunicación entre los integrantes, de manera que todo el tráfico de la comunicación pasa por el atacante. De esta manera, puede obtener, observar y modificar la información y reenviarla de nuevo o manipularla de cualquier forma.

**Integridad:** este principio básico de la seguridad de la información versa sobre que la información que es transmitida no sea modificada durante el trayecto.

**Confidencialidad:** principio de la seguridad de la información que consiste en que si se intercepta la información o el mensaje sea ininteligible, de manera que únicamente los verdaderos integrantes de la comunicación puedan comprender el contenido del mensaje.

**Autenticidad:** este principio consiste al recibir información de alguna de las partes de la comunicación, sea enviada realmente por ese integrante y no por otra entidad que no pertenece a la comunicación.

**Bash:**

**Python:**

# Introducción

## Introducción

Cada vez más y más la población hace uso de terminales móviles. De hecho, habitualmente es más común que en las familias cada uno de los miembros disponga de un dispositivo móvil y no, por ejemplo, ordenadores de sobremesa o portátiles. En estos dispositivos se genera, recolecta y almacena grandes cantidades de información personal del usuario (y de sus contactos, personas próximas) que puede ser consultada y utilizada por las aplicaciones del móvil.

Además de lo comentado anteriormente, es necesario tener en cuenta la existencia de las redes públicas Wi-Fi, en las cuales cualquiera puede conectarse y hacer uso de sus servicios y, además, las redes de uso común como pueden ser las de las universidades o centros de trabajo, etc. Tener todo esto en cuenta supone darse cuenta de que algún individuo podría ser capaz de interceptar el tráfico de los diferentes terminales conectados a una red. Si este atacante consiguiera interceptar tráfico, esto significaría que es capaz de obtener en claro los mensajes y sería capaz de obtener datos que ofrecen las aplicaciones.

Este trabajo se centra en la investigación y análisis del método de seguridad que trata de impedir ataques de suplantación de certificados (Certificate Pinning), los cuales son necesarios para evitar los ataques mencionados previamente. Además, el trabajo se centrará en implementar diferentes métodos que consigan romper Certificate Pinning para conseguir suplantar los certificados de las aplicaciones de un terminal móvil y así, ser capaces de realizar ataques MiTM.

## Objetivos

El principal objetivo de este Trabajo Fin de Master consiste en primer lugar, investigar y estudiar los diferentes mecanismos de implementación más utilizados y comunes de Certificate Pinning, ya que estos serán los que aparezcan en la gran mayoría de aplicaciones Android, además, después de este estudio, se diseñarán soluciones que permitan sobrepasar dichos mecanismos mediante el empleo de técnicas de instrumentación dinámica de las aplicaciones. A continuación, se listas los objetivos más concretos.

* Estudiar y comprender los diferentes ataques que se pueden producir a las aplicaciones Android mediante el tráfico de red.
* Analizar las posibles soluciones ya aplicadas a los ataques producidos.
* Estudio del Certificate Pinning, su funcionamiento e implementación en las diferentes aplicaciones de sistemas Android.
* Análisis y estudio de las herramientas más comunes para poder realizar ataques de tráfico en aplicaciones Android.
* Estudio e implantación del entorno de trabajo.
* Selección y estudio de herramientas que se puedan utilizar para sobrepasar el mecanismo de seguridad Certificate Pinning de Android.
* Analizar diferentes aplicaciones que dispongan de soluciones Certificate Pinning mediante técnicas de instrumentación dinámica para conseguir romper esta seguridad y, tratar de encontrar de patrones para automatizar los ataques.
* Diseño de posibles soluciones para poder sobrepasar el mecanismo de seguridad de las aplicaciones Android basado en el método de Certificate Pinning.
* Pruebas del correcto funcionamiento de los métodos o soluciones implementadas.
* Análisis de los resultados obtenidos frente a anteriores propuestas de trabajo sobre ataques a las aplicaciones Android.
* Documentación de todo el trabajo realizado.

# Estado del arte

## Introducción

Las principales referencias en las que se ha basado este trabajo para continuar con la línea en la que se dejaron

¿Apartado de seguridad en las comunicaciones?

## Estructura de certificados

## Comunicación segura: TLS/SSL

## Mecanismos de seguridad de comunicaciones en Android

En esta sección del trabajo se explican los principales mecanismos que se utilizan en los dispositivos móviles para poder comunicarse entre ellos, ya sea para transferencia de ficheros o comunicaciones entre las aplicaciones que residen en el terminal con sus respectivos servidores para solicitar recursos.

Este apartado se divide en los diferentes mecanismos de transferencia y comunicación y mecanismos de seguridad que se utilizan para proteger el tráfico de datos entre dispositivos y servidores. Se explicarán los siguientes: comunicación TLS/SSL, certificate pinning (objeto principal de estudio de este trabajo), PONER MÁS

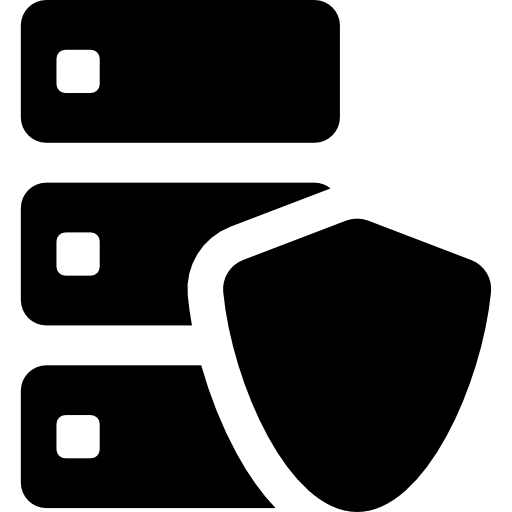
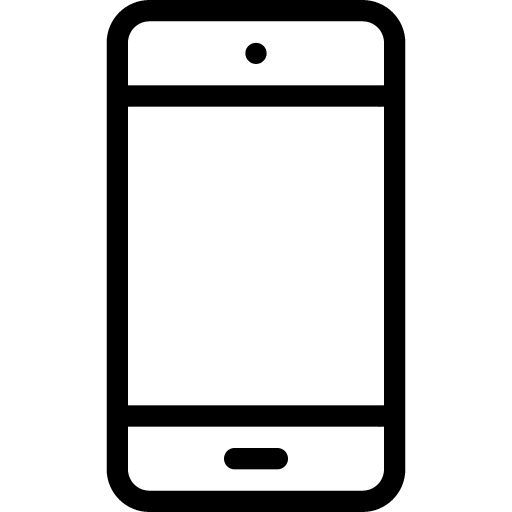
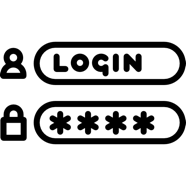
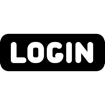
### Certificate pinning

El certificate pinning o traducido, fijación de certificados, es un mecanismo a partir del cual se relaciona un host con su respectiva clave pública de tal manera que trata de conseguir solucionar o mitigar el problema que se produce en la verificación de las cadenas de certificados. Por lo general, la confianza en los certificados SSL se reduce simplemente a los certificados raíz, es decir, se escalan certificados hasta alcanzar el que es fiable (certificado raíz). Estos certificados vienen por defecto en el sistema operativo, navegadores y demás dispositivos, pero el problema realmente llega a la hora de verificar el certificado del propio dispositivo. Únicamente se hacen comprobaciones de los certificados del servidor.

La fijación de certificados se refiere al proceso por el cual se importan certificados en un dispositivo o máquina host en su depósito de certificados de confianza de manera que, no es necesario confiar en los certificados de las CA (Autoridades de Certificación), es decir, mitiga el riesgo de que las CA se vean comprometidas.

En el caso de Android, las aplicaciones llevan incluidos desde el código de la propia aplicación una copia del certificado de tal manera que, se compara este certificado anclado con el certificado del servidor, para así evitar problemas con la validez de las CA y de los certificados raíz. Ahora bien, un problema evidente es que es necesario actualizar los certificados manualmente al caducar.

En la figura que se muestra a continuación, se representa en la parte derecha un terminal móvil Android que dispone de aplicaciones. Como se muestra, dicha aplicación en el terminal contiene un certificado que corresponde al del servidor con el que se conecta el dispositivo móvil. El primer paso, es el “saludo” por parte del terminal indicando que quiere establece una comunicación con el servidor, esta fase (1), se conoce como *TLS Handshake*. Una vez producido el saludo, el servidor envía su certificado al terminal, fase (2) y este, verifica que corresponde con el certificado que tiene almacenado mediante el uso de la clave, fase (3). Cuando ya se ha comprobado la validez del certificado, entonces ya puede realizarse una comunicación entre el servidor y el terminal móvil, fase (4).



4

**Terminal móvil**

**Verificación de la clave fijada**

3

**Envío de certificado**

1

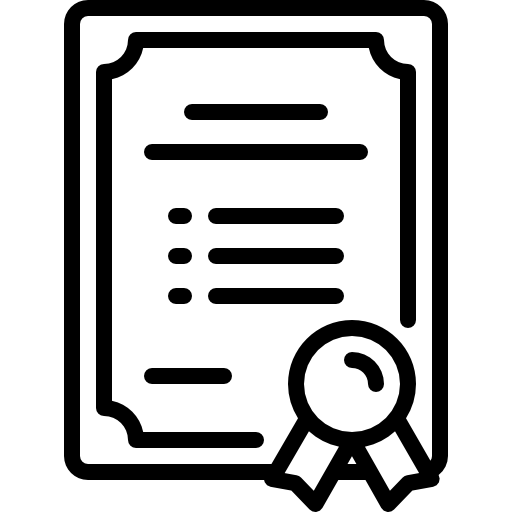
**TLS Handshake**

**Servidor**

2

**App**

**Comienzo de la comunicación**



**Certificado**

Figura 2.1. Esquema de funcionamiento certificate pinning

AÑADIR LAS BIBLIOTECAS 🡪 MÉTODOS QUE SE HAN IDENTIFICADO PARA HACER EL CERTIFICATE PINNING

## Captura de tráfico

## Análisis de las aplicaciones Android

# Desarrollo

## Herramientas

En este apartado se explican las herramientas que se han utilizado para poder desarrollar la implementación de las soluciones sobre la fijación de certificado en las aplicaciones Android.

* **VirtualBox:** es un software de código libre cuya funcionalidad es la de ser un host hipervisor [3], es decir, permite crear múltiples máquinas virtuales de diferentes sistemas operativos sin tener que crear particiones o modificar el propio sistema de la máquina anfitrión. Para el trabajo desarrollado, ha sido necesario su uso para poder crear dos máquinas virtuales con sistemas operativos Parrot y Android x86, que permitan realizar el estudio, la implementación y pruebas necesarias sobre las soluciones a certificate pinning.
* **Parrot:** es un sistema operativo basado en la distribución Debian GNU de Linux [6]. Este sistema operativo se utiliza como sistema base a partir del cual se obtendrá la imagen de Docker (en la que se encuentra parte del trabajo previo) y con lo que se trabajará para desarrollar los mecanismos para lograr hacer bypass de certificate pinning. Se ha escogido este sistema operativo debido a que cuenta con un gran número de herramientas de seguridad y, está especialmente diseñado para labores de seguridad informática.
* **Android x86:** sistema basado en la imagen de un terminal Android [4] de forma que se puede emular dicho dispositivo. Se utilizará para realizar las correspondientes pruebas de funcionamiento del bypass de la fijación de certificados, mediante la instalación de forma remota (desde la máquina Parrot) de aplicaciones.
* **Docker:** es un software de código abierto que trata de agilizar y automatizar el lanzamiento de aplicaciones en contenedores [7], lo que provoca que se cree una capa más de virtualización. En este proyecto se utilizará ya que, al continuar con proyectos anteriores, se almacena toda la plataforma en un contenedor para facilitar su despliegue. Además de utilizar el contenedor que ya está propuesto, los cambios que se realicen serán modificados en el contendor.
* **Adb:** Android Debug Bridge,es una herramienta de línea de comandos la cual permite la comunicación con terminales [7]. Algunas de las acciones que se pueden realizar con esta herramienta son, por ejemplo, acceder al terminal, modificar aplicaciones (instalar y desinstalar APKs), eliminar ficheros, transferencia de ficheros entre la máquina conectada y el dispositivo Android.
* **Apktool:** herramienta que se utiliza para descompilar y así obtener el código de la aplicación Android [9]. El código que se obtiene a partir de la aplicación está desarrollado en lenguaje Smali, si se quiere obtener el código de la aplicación en leguaje Java, serán necesarias otras herramientas. En el caso de este trabajo, se utilizará Apktool para acceder al código de determinadas aplicaciones Android para estudiarlo y modificarlo y así, lograr realizar el bypass de certificate pinning.
* **Mitmproxy:** esta herramienta de línea de comandos es un proxy que se utiliza para interceptar tráfico de las comunicaciones entre diferentes dispositivos [5].Además, dispone de interfaz web y de API las cuales facilitan la visualización del tráfico. En este proyecto se hará uso de esta herramienta para poder interceptar las comunicaciones entre la máquina base (Parrot) y el terminal Android x86.
* **Frida:** es una herramienta dinámica de ingeniería inversa [10] mediante la cual se pueden inyectar scripts en distintos sistemas, lanzar hooks (ganchos) contra APIs para lograr acceder sin necesidad de tener las credenciales, entre otras funciones.En este proyecto se utilizará Frida para cargar programas personalizados y de esta manera conseguir evitar la fijación de certificados en las aplicaciones de Android.

## Entorno

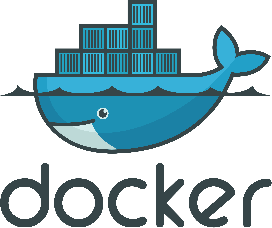
A continuación, se explica y detalla el entorno que se ha levantado para poder llevar a cabo el proyecto. Además, se comentará en qué consiste la plataforma en la que se dispone todo el sistema, cuáles son sus elementos, detallando más concretamente aquellos en los que se centra el desarrollo de este trabajo.

En la figura que se muestra a continuación, se detalla el entorno de trabajo establecido. El entorno de trabajo se desarrolla bajo el uso de la aplicación Virtual Box [3], a partir del cual se han instalado por un lado el sistema operativo Parrot (Linux, de distribución Debian GNU), que será el sistema base y, por otro lado, se ha instalado el sistema Android x86 [4], que es un emulador del sistema Android y que se utilizará como sistema de pruebas en lugar de un terminal móvil físico.

La única parte que cabe remarcar sobre la instalación de los dos sistemas en Virtual Box, son las configuraciones de red ya que como se aprecia en la figura de abajo, es necesario disponer de una red interna y de otra externa. La red interna se utilizará para la comunicación entre las máquinas de Parrot y Android x86, por tanto, es necesario establecer direcciones IP de las máquinas en la misma subred. Por otro lado, la red externa se empleará para conectar a Internet las máquinas. Tal y como se observa en la figura, la máquina Parrot es la que dispondrá de otro adaptador más a partir del cual se conectará a internet. Este adaptador tiene que ser del tipo puente, para permitir comunicaciones entre las máquinas virtuales a la vez que se permite la comunicación con internet. En la máquina Parrot se establecerá el proxy (mitmproxy [5]) a través del cual se conectará la máquina Android x86 de forma que tendrá acceso a Internet y así, desde el sistema Parrot se captura el tráfico.

Es necesario tener en cuenta que la máquina base, en este caso con sistema operativo Parrot, dispondrá de dos interfaces de red. Uno en modo puente y otro en modo red interna. La interfaz de red en modo puente tendrá asignada una dirección IP por DHCP, en cuanto a la otra interfaz de red, la interna, del sistema base, tendremos que asignarle una dirección IP estática, cuya subred sea diferente a la de la otra interfaz. En cuanto a la máquina Android x86, al arrancar la máquina hay que verificar que la conexión a Internet se produce a través de una interfaz de red en modo DHCP. Para asignar la dirección IP a esta máquina, se utilizará un fichero de configuración el cual hace uso del servicio dnsmasq [6]. Se puede encontrar más información acerca de este servicio y configuración en el Anexo de este documento.

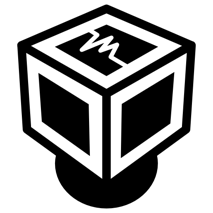
Por otro lado, para poder hacer uso de la plataforma ya desarrollada (referencia TFM y TFG), es necesario acceder a Docker desde la máquina virtual Parrot ya que es en esta donde se desarrolla e implementan los métodos para anular la fijación de certificados de las aplicaciones Android que estarán situadas en la otra máquina (Android x86) y a partir de la cual mediante el uso de la herramienta adb [7], mitmproxy [5] y Frida como principales herramientas, se desplegarán aplicaciones Android en la máquina y se analizará el tipo de certificate pinning que tienen asociado para conseguir revocarlo.



**Docker**



**Red externa**

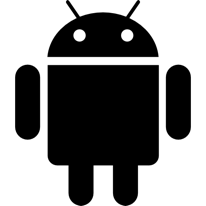


**Parrot (proxy)**

**Red interna**

**Internet**

**Virtual Box**





**Android x86**

Figura 3.1. Esquema del entorno de trabajo

## Plataforma

La plataforma sobre la que se trabaja se obtiene mediante una imagen de Docker, proporcionada por UNIVERSIDAD. Esta plataforma dispone de varios segmentos, pero para este proyecto, únicamente será necesario centrarse en la parte de análisis de tráfico.

## Método de identificación de bibliotecas de certificate pinning

Uno de los primeros problemas que surgen al comienzo de este proyecto es saber qué librerías son las que implementan algún tipo de mecanismo de fijación de certificados. Con varias búsquedas se encontraron bastantes librerías que lo implementan, como se puede consultar en [12], pero entonces, es necesario averiguar si realmente hay aplicaciones Android que hacen uso de estas librerías.

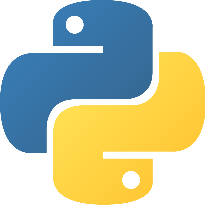
¿Tabla con las librerías encontradas? 🡪 Pasar la tabla de abajo al apartado de Estado del Arte. En la tabla indicar las librerías y los métodos más destacados o utilizados para poder realizar el bypass.

|  |  |  |
| --- | --- | --- |
| OpenSSLSocketImpl | OpenSSLEngineSocketImpl | OpenSSLSocketImpl Apache |
| II | 2 | 10 |
| III | 5 | 15 |

Para poder establecer si realmente las librerías encontradas son utilizadas en el desarrollo de las aplicaciones, se diseñan unos scripts que serán capaces de identificar si se hace uso de alguna de estas librerías en el código de cada aplicación que se quiera analizar. Para poder realizar este proceso, en primer lugar, es necesario disponer de varios APKs y no solamente esto, también hay que disponer del código de cada APK. Para ello, se utilizarán técnicas de ingeniería inversa y mediante el uso de la herramienta Apktool [8], se obtendrá de cada APK el código smali. Una vez obtenido el código, simplemente será necesario realizar una búsqueda en el código de cada una de las librerías que son conocidas por implementar certificate pinning.

El funcionamiento de los scripts es el que se relata a continuación. Por cada APK se crea un directorio el cual contiene el código de dicha APK. A partir de estos directorios se crean otro nuevos en los que se almacenarán ficheros en formato json cuyo nombre será el de la librería que se está buscando y que contendrán los resultados de las búsquedas. Es decir, habrá ficheros que si no han encontrado ningún resultado de la búsqueda estarán vacíos. Este proceso lo realizará un fichero en bash llamado *find\_libraries\_pinning.sh*. Posteriormente, mediante otro script en Python llamado *check\_pinning\_library.py* se analizarán los ficheros json anteriormente creados. Este script de Python verifica si están vacíos o no, en base a esta comprobación se generan ficheros en formato CSV que incluyen toda la información acerca de las APKs analizadas con sus respectivas librerías y si se ha encontrado algún indicio de que el código smali de cada APK tiene alguna de las librerías se indica en el CSV mediante la palabra clave “True”, si por el contrario no se ha encontrado nada entonces se indicará con “False”. Para unificar estos dos scripts, se ha implementado un script en bash, llamado *pinning\_libraries.sh*. Este script tiene como función llamar a los dos anteriores para así simplemente tener que ejecutar este y que realice todas las operaciones.

En el siguiente esquema se muestra gráficamente el funcionamiento general de los scripts explicados anteriormente para lograr identificar librerías que implementen mecanismos certificate pinning en el código smali obtenido mediante Apktool [8].



**check\_pinning\_library.py**

**find\_libraries\_pinning.sh**



**pinning\_libraries.sh**

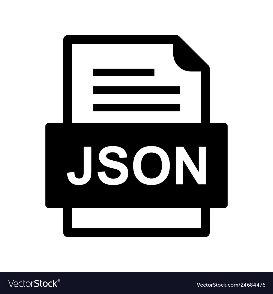
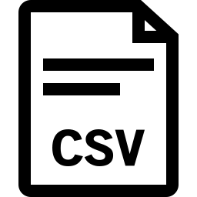


Figura 3.2. Esquema del sistema de búsqueda de bibliotecas con certificate pinning

### Requisitos

Para comenzar, es requerimiento indispensable disponer de un sistema que disponga de terminal para poder ejecutar el programa. Por otro lado, como ya se mencionó, uno de los scripts del conjunto del programa está escrito en Python 3 por lo que será necesario disponer de dicho lenguaje de programación. Además, en el script de Python se utilizan dos importaciones de módulos, uno de ellos es *os* [14] y el otro es *csv* [15]. Es necesario que estén disponibles estas librerías para poder ejecutar el programa de forma correcta.

Este programa, como ya se explicó en el anterior apartado, hace uso de la herramienta Apktool [8]. Para que funcione correctamente la herramienta es necesario pasar como argumento un APK, por lo que es necesario disponer de algún APK antes de la ejecución del programa. Además, es necesario que el APK esté guardado en un directorio con el nombre “APKs” y que se localice en la misma ruta en la que se ubica el script del programa.

Finalmente, el usuario debe saber que al ejecutar el programa se crearán varios directorios. Estos directorios se llaman, “Resources” que contendrá otros tres directorios: “Code”, que contendrá el resultado de la ejecución de Apktool, es decir, el código de todos los APKs que se encuentren en el directorio “APKs”. “Files”, en este directorio se almacenan directorios correspondientes a cada APK disponible, que a su vez tendrán los ficheros json con las librerías encontradas y un fichero CSV con la información recogida. Finalmente, habrá otro directorio llamado “Logs”, que almacenará los diferentes ficheros de log generados por la ejecución del programa.

### Ejecución

Como se ha explicado en el apartado anterior, se ha creado un script en bash que funciona de enlace entre los otros dos, de tal forma que únicamente es necesario ejecutar este script para obtener los resultados esperados (ficheros CSV) acerca de si un APK utiliza para su implementación ciertas librerías conocidas de certificate pinning.

Para ejecutar el script genérico, *pinning\_libraries.sh*, es decir, el que devuelve todos los resultados sin tener que ejecutar ningún otro script, es suficiente con escribir *sudo* *bash pinning\_libraries.sh*. Como se aprecia en el comando, es necesario tener permisos de administrador. En la imagen que se muestra debajo, se indica de forma visual cómo se ha de ejecutar el comando y cuáles son las primeras líneas que aparecen al ejecutar el script.

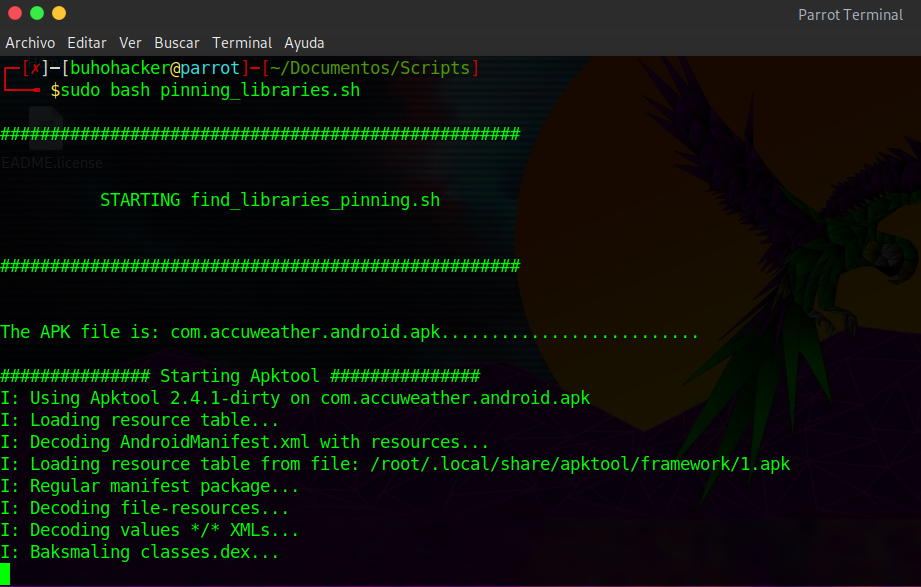


Figura 3.3. Ejecución del script pinning\_libraries.sh

Otra opción de ejecución sin necesidad de utilizar el script base *pinning\_libraries.sh*, sería ejecutar por separado los otros dos scripts, esto puede ser interesante si es necesario analizar las operaciones que realiza el código de cada uno de los scripts. Hay que tener en cuenta que para que funcione el script *check\_pinning\_library.py*, es necesario haber ejecutado previamente el otro script de bash, *find\_libraries\_pinning.sh*. Esto se debe a que el script de bash crea unos ficheros con formato json que posteriormente recogerá para analizar el script de Python. Para ejecutar *find\_libraries\_pinning.sh*, simplemente es necesario escribir en la terminal *sudo bash find\_libraries\_pinning.sh*. La ejecución del otro script requiere tener instalado Python en el sistema, para lanzar el script escribir el comando *python3* *check\_pinning\_library.py*, como observación, mencionar que en principio se puede ejecutar también con *python* en lugar de *python3,* pero se aconseja utilizar las versiones más recientes directamente.

En las siguientes imágenes se muestra en la primera, la ejecución del comando que lanza el script *find\_libraries\_pinning.sh* y a continuación, una imagen que muestra el comando y el principio de la ejecución del script *check\_pinning\_library.py.*

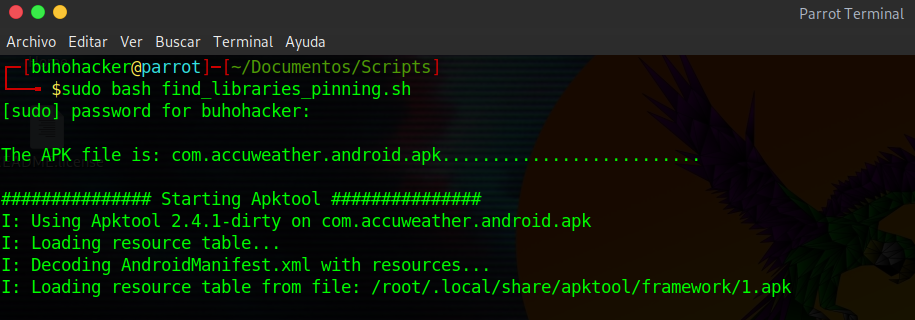


Figura 3.4. Ejecución del script find\_libraries\_pinning.sh

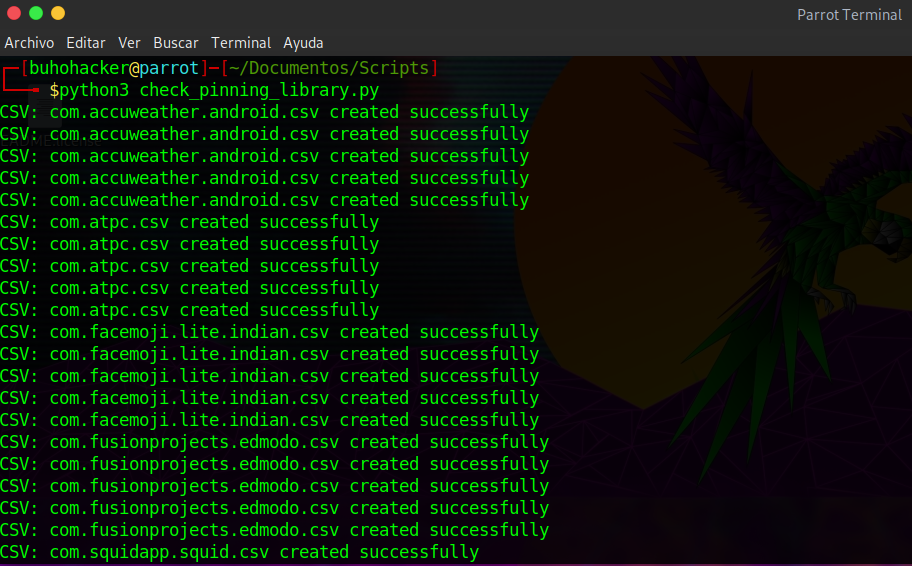


Figura 3.5. Ejecución del script check\_pinning\_library.py

Durante la ejecución del programa se van generando diferentes comentarios informativos por pantalla para que el usuario pueda saber en qué punto se encuentra del proceso y, sobre diferentes errores que puedan haber surgido durante ejecución. Además, se generan logs con posibles errores surgidos y para almacenar el resultado de las ejecuciones. Dichos ficheros se pueden encontrar en el directorio *Logs*.

### Salida

Una vez ejecutados los scripts, como se muestra en el esquema de funcionamiento del sistema en el primer apartado, se obtiene como resultado varios ficheros. Al ejecutar el script *find\_libraries\_pinning.sh*, se obtienen varios ficheros en formato json con las ocurrencias de las librerías encontradas (sí que se ha hecho uso de ellas en el desarrollo de la aplicación que se está analizando). En la siguiente imagen se muestra el contenido de uno de los ficheros. El nombre que recibirá cada uno de estos ficheros corresponde al nombre de la librería que se está analizando.

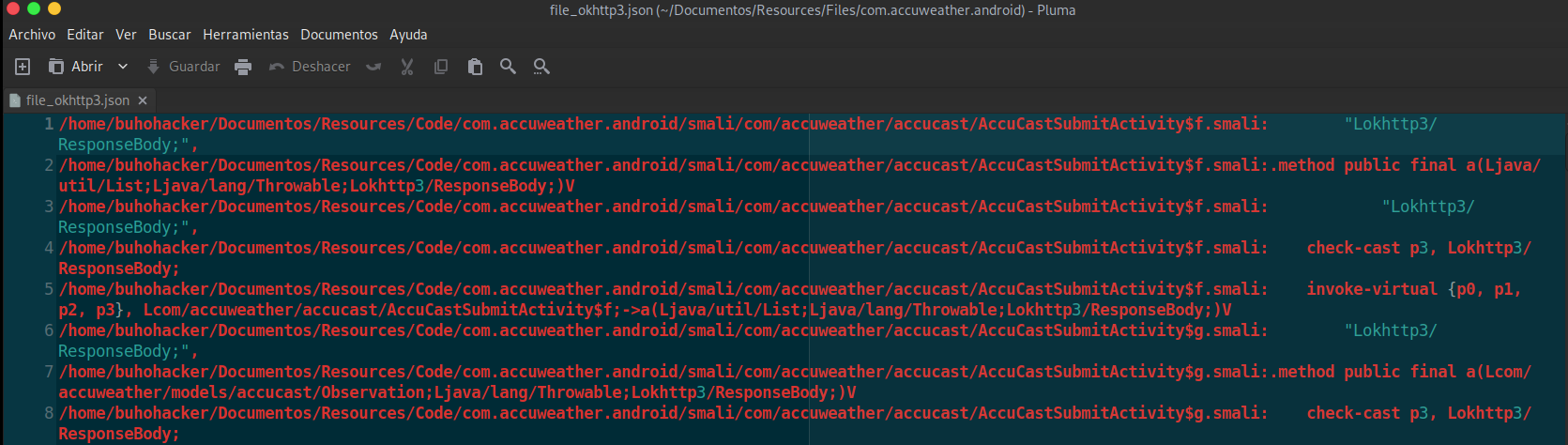


Figura 3.6. Contenido de uno de los ficheros json

Por otro lado, el otro script de Python, después de realizar comprobaciones y determinar si realmente hay librerías o no en el código del APK analizado, crea un fichero con formato CSV cuyo nombre corresponderá con el del APK analizado. Este fichero CSV tiene como columnas los siguientes valores: nombre del APK, nombre de la librería y finalmente el valor “True” si utiliza la librería o “False” si no la utiliza.

En la imagen que se muestra a continuación, se ve el contenido de uno de los ficheros CSV generados por el sistema desarrollado. Como se aprecia en la imagen, el APK con el que se ha trabajado para esta demostración es *com.facemoji.lite.indian*, en la siguiente columna aparecen los nombres de las librerías con las que se ha probado, *opensslsocketimpl*, *okhttp3*, *trustkit* y *trustmanager.* Y en la última columna se indica si ese APK en concreto tiene o no dichas librerías.

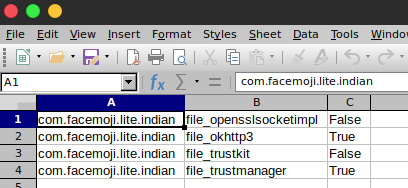


Figura 3.7. Contenido de un fichero CSV generado por el sistema

## Implementación

Una vez que ya se ha conseguido averiguar qué aplicaciones son las que incluyen mecanismos de certificate pinning y más concretamente, se ha podido averiguar qué librería en particular utilizan, el siguiente paso consiste en tratar de saltar los mecanismos de dichas librerías para así poder obtener datos acerca del tráfico de la aplicación y comprobar si se ha conseguido romper el mecanismo de implementación de fijación de certificados a cada correspondiente aplicación.

El primer apartado de esta sección explica el proceso que se ha seguido para lograr romper la fijación de certificados mediante el uso de librerías conocidas y verificadas con el sistema descrito en el apartado anterior.

Segundo apartado? 🡪 IDA

Poner los diferentes métodos de saltarse Pinning

Varios escenarios?? 🡪 Más con bibliotecas y otro método diferente

### Caso 1: certificate pinning con paquetes

Identificación de paquetes: 🡪 documentar todas las librerías que se salten

* OpenSSlSocketImpl Conscrypt
* OpenSSlEngineSocketImpl Conscrypt
* OpenSSlSocketImpl Apache Harmony
* PhoneGap sslCertificateChecker

En el fichero *pinning\_cases.js*

### Caso 2: certificate pinning X

IDA Pro?

# Resultados

TRATAMIENTO DE LOGS?

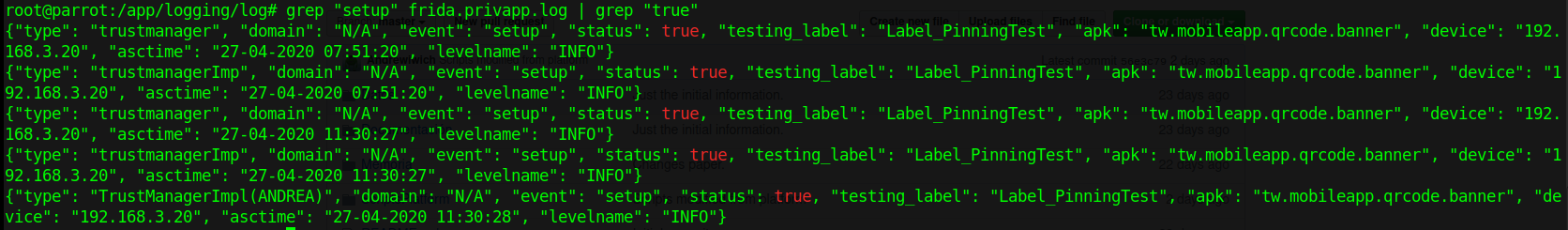
¿Gráfico con tiempo de ejecución de la búsqueda de bibliotecas ?

## Caso de nuevos métodos

**Caso 1:**

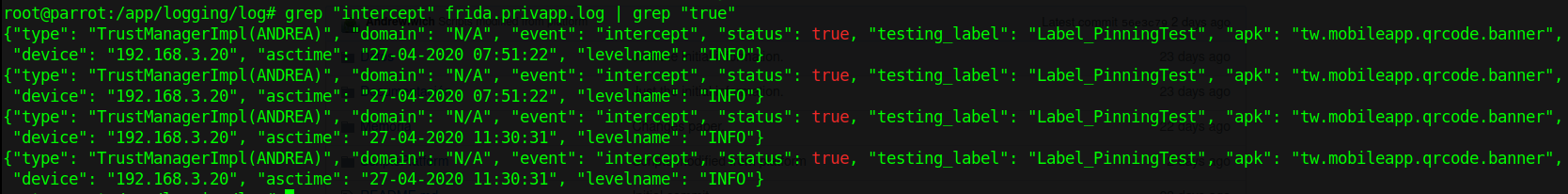
APK 🡪 tw.mobileapp.qrcode.banner

Fallo bypass 🡪 Se detecta que el APK hace uso de las librerías trustmanager y TrustaManagerImpl



Solución:

* Librería TrustManagerImpl. Se analizan las diferentes funciones de la librería y se utiliza el código que se muestra a continuación, encontrado en <https://gist.github.com/1mm0rt41PC/cd492f24dc061019fb25222ba0b96d20>. Tras realizar modificaciones e implementar el bypass para esa librería se consigue saltar tal y como se muestra en la siguiente imagen.

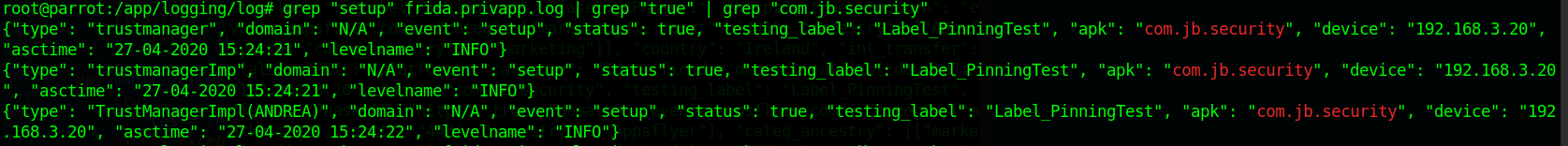


* Librería trustmanager. <https://www.natapuntes.es/trustmanager-android/>

**Caso 2:**

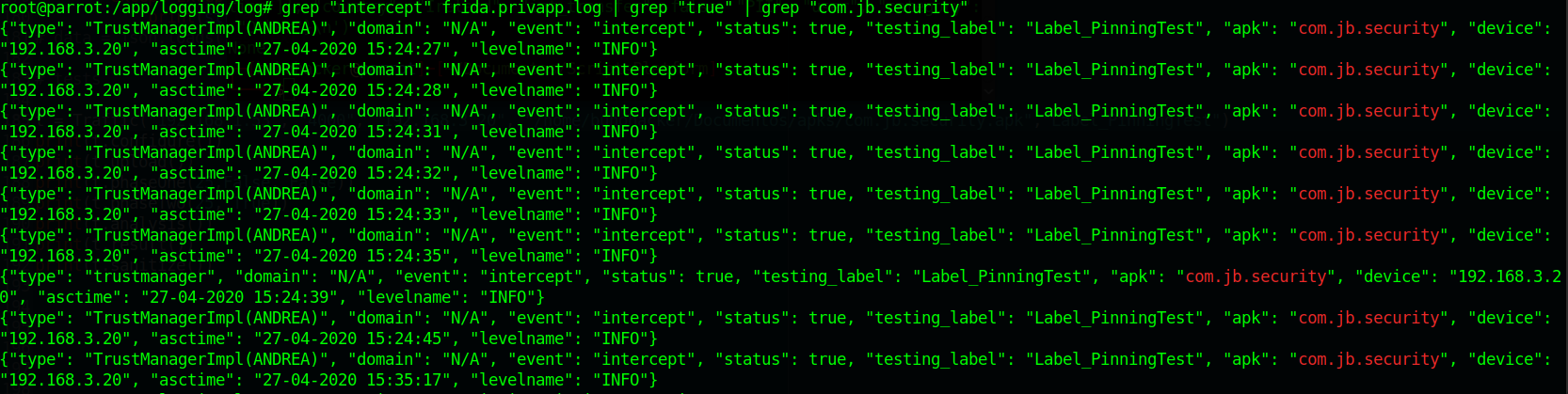
APK 🡪 com.jb.security

Fallo bypass 🡪 Se detecta que el APK hace uso de las librerías TrustaManagerImpl



Solución:

* Librería TrustManagerImpl. Se analizan las diferentes funciones de la librería y se utiliza el código que se muestra a continuación, encontrado en <https://gist.github.com/1mm0rt41PC/cd492f24dc061019fb25222ba0b96d20>. Tras realizar modificaciones e implementar el bypass para esa librería se consigue saltar tal y como se muestra en la siguiente imagen.

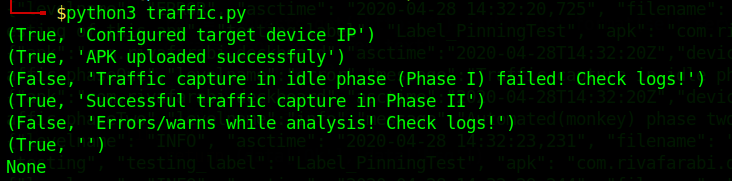


**Caso 3:**

APK 🡪com.rivarafari**.**deckboard:

Problema al tratar de instalar el apk en el terminal. Solución en <https://stackoverflow.com/questions/15014519/apk-installation-failed-install-failed-verification-failure>

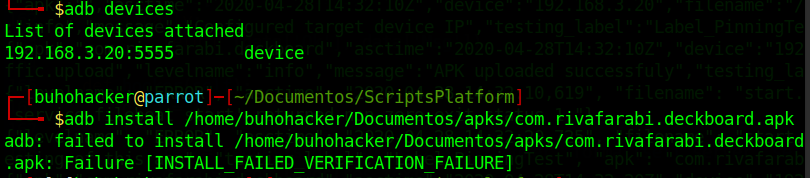
En la imagen de abajo se muestra el error que aparece al tratar de ejecutar el fichero trafic.py.



Al revisar los logs (operation) se descubre que falla la instalación del apk en el terminal, como se muestra en la siguiente captura.

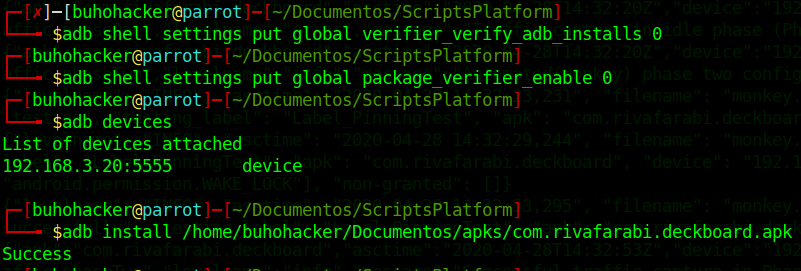


Al ver que falla el script, se trata de realizar la instalación del APK de forma manuela, tal y como se ve a continuación. Además, se aprecia que efectivamente, ocurre un error al tratar de instalar el APK desde adb incluso de forma manueal.



Se encuentra la solución al fallo y es que al parecer, en las nuevas versiones de Android puede aparecer ese error debido al sistema de protección de Play Protect Añadir cita aquí con este enlace <https://www.android.com/intl/es_es/play-protect/>

Una vez identificado el error se soluciona siguiendo los pasos que se muestran a continuación:

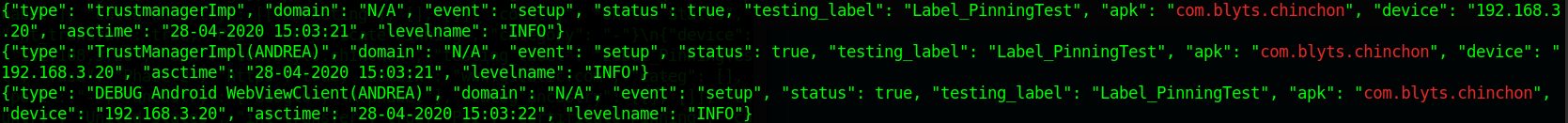


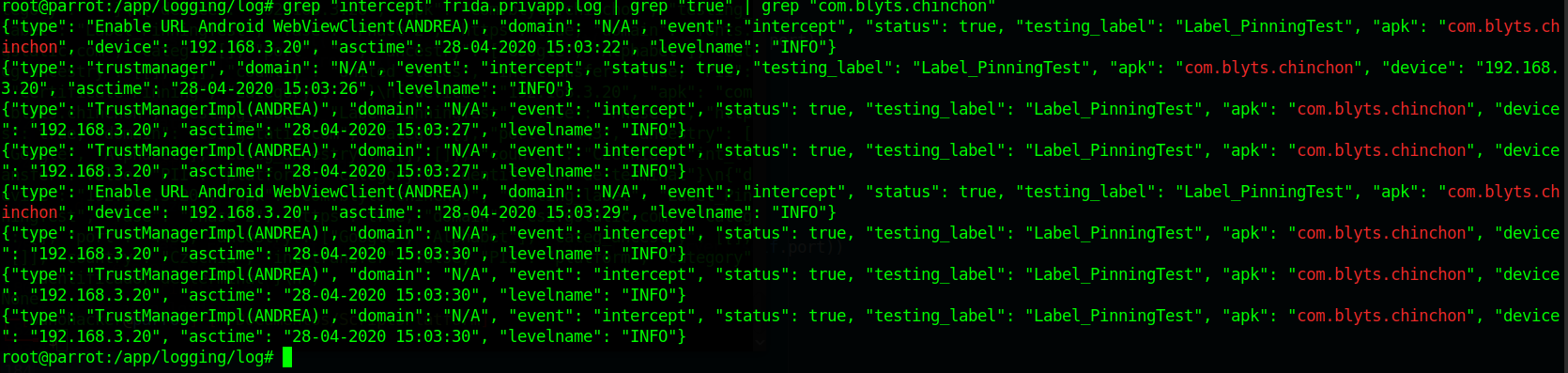
Con los pasos que se muestran en la imagen anterior, lo que se hace es conseguir deshabilitar la verificación del APK mientras se está instalando n el terminal.

**Caso 4:**

APK 🡪 com.blyts.chinchon

Lo que sucede con esta APK es que se ha detectado al igual que en el caso 1 y 2, que hace uso de las librerías trustManagerImpl para poder realizar la fijación de certificados. Con el nuevo método implementado se consigue saltar la fijación de certificado de dicha librería, tal y como se muestra en la siguiente imagen:





AÑADIR TABLA RESUMEN CON RESULTADOS DE LAS APKS PROBADAS

# Conclusiones y líneas futuras

Apéndices y/o anexos

Configuración dnsmasq

# Bibliografía

|  |  |
| --- | --- |
| [1] | Wikipedia, «Wikipedia,» 15 02 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Protocolo\_de\_configuraci%C3%B3n\_din%C3%A1mica\_de\_host. [Último acceso: 16 04 2020]. |
| [2] | «Redalia,» [En línea]. Available: https://www.redalia.es/ssl/protocolo-ssl/. [Último acceso: 08 03 2020]. |
| [3] | VirtualBox, «VirtualBox,» [En línea]. Available: https://www.virtualbox.org/wiki/Documentation. [Último acceso: 13 04 2020]. |
| [4] | A. x86, «Android x86,» [En línea]. Available: https://www.android-x86.org/. [Último acceso: 13 04 2020]. |
| [5] | mitmproxy, «mitproxy,» [En línea]. Available: https://mitmproxy.org/. [Último acceso: 13 04 2020]. |
| [6] | P. Team, «The Parrot Project,» [En línea]. Available: https://parrotlinux.org/docs/#documentation. [Último acceso: 13 04 2020]. |
| [7] | D. Docs, «Docker Documentation,» [En línea]. Available: https://docs.docker.com/. [Último acceso: 13 04 2020]. |
| [8] | A. Studio, «Android developers,» [En línea]. Available: https://developer.android.com/studio/command-line/adb. [Último acceso: 13 04 2020]. |
| [9] | Apktool, «Apktool,» [En línea]. Available: https://ibotpeaches.github.io/Apktool/. [Último acceso: 13 04 2020]. |
| [10] | «IBM Mobile Foundation,» 27 02 2020. [En línea]. Available: https://mobilefirstplatform.ibmcloud.com/tutorials/es/foundation/8.0/authentication-and-security/certificate-pinning/. [Último acceso: 08 03 2020]. |
| [11] | IconsMind, «IconsMind,» [En línea]. Available: https://iconsmind.com/. [Último acceso: 13 04 2020]. |
| [12] | IconArchive, «IconArchive,» [En línea]. Available: http://www.iconarchive.com/show/outline-icons-by-iconsmind/Smartphone-4-icon.html. [Último acceso: 13 04 2020]. |
| [13] | IconArchive, «IconArchive,» [En línea]. Available: http://www.iconarchive.com/show/papirus-devices-icons-by-papirus-team/network-server-icon.html. [Último acceso: 13 04 2020]. |
| [14] | Freepik, «Freepik,» [En línea]. Available: https://www.freepik.com/free-icon/login-password\_870212.htm#page=1&query=login&position=0. [Último acceso: 13 04 2020]. |
| [15] | Freepik, «Freepik,» [En línea]. Available: freepik.com/free-icon/login\_863353.htm#page=1&query=login&position=8. [Último acceso: 13 04 2020]. |
| [16] | Freepik, «Freepik,» [En línea]. Available: https://www.freepik.com/free-icon/database\_873831.htm#page=1&query=server&position=13. [Último acceso: 13 04 2020]. |
| [17] | Icons8, «Icons8,» [En línea]. Available: https://icons8.com/icons/set/hi-tech. [Último acceso: 13 04 2020]. |
| [18] | FreeIconsPNG, «FreeIconsPNG,» [En línea]. Available: https://www.freeiconspng.com/images/site-internet-icon. [Último acceso: 13 04 2020]. |
| [19] | L. Download, «Logos Download,» [En línea]. Available: https://logos-download.com/9688-docker-logo-download.html. [Último acceso: 13 04 2020]. |