Maestría en

CIBERSEGURIDAD

Trabajo previo a la obtención de título de
Magister en Ciberseguridad

AUTOR/ES:

Jefferson Alexander Guala Fonseca

Alan Patrick Jaramillo Pozo

Santiago Xavier Ordóñez Rosenberg

Sebastián Aldhair Tejada Mendoza

TUTOR/ES:

Alejandro Cortés

Iván Reyes Chacón

TEMA

Análisis de comportamiento de malware en dispositivos Android: Un enfoque práctico en entornos controlados con Android

Quito - Ecuador

Enero– 2026

REINVENTEMOS EL FUTURO

Certificación de autoría

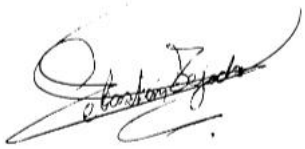
Nosotros, Jefferson Alexander Guala Fonseca, Alan Patrick Jaramillo Pozo, Sebastián Aldhair Tejada Mendoza y Santiago Xavier Ordóñez Rosenberg, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

Firma
Jefferson Alexander Guala Fonseca

Firma
Alan Patrick Jaramillo Pozo

Firma
Santiago Xavier Ordóñez Rosenberg



Firma
Sebastián Aldhair Tejada Mendoza

Autorización de Derechos de Propiedad Intelectual

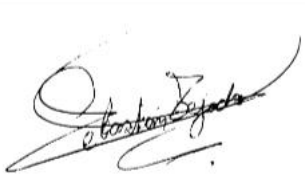
Nosotros, **Jefferson Alexander Guala Fonseca, Alan Patrick Jaramillo Pozo, Sebastián Aldhair Tejada Mendoza, Santiago Xavier Ordóñez Rosenberg**, en calidad de autores del trabajo de investigación titulado **Análisis de comportamiento de malware en dispositivos Android: Un enfoque práctico en entornos controlados con Android**, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, (diciembre 2025)

Firma
Jefferson Alexander Guala Fonseca

Firma
Alan Patrick Jaramillo Pozo

Firma
Santiago Xavier Ordóñez Rosenberg



Firma
Sebastián Aldhair Tejada Mendoza

Aprobación de dirección y coordinación del programa

Nosotros, **Alejandro Cortés e Iván Reyes,** declaramos que: **Jefferson Alexander Guala Fonseca, Alan Patrick Jaramillo Pozo, Sebastián Aldhair Tejada Mendoza, Santiago Xavier Ordóñez Rosenberg, son** los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.

Alejandro Cortés L.

Maestría en Ciberseguridad

Iván Reyes Ch.

Maestría en Ciberseguridad

DEDICATORIA

Dedico este trabajo a mis tres hijos, mi mayor motivo y mi fuerza diaria, a quienes deseo dejarles un ejemplo de perseverancia y amor por el aprendizaje; a mis padres, por inculcarme valores de esfuerzo, honestidad y constancia; y a mis compañeros, por su paciencia, apoyo y disposición para enseñarme y aprender juntos cada día. Este logro también es fruto de mi resiliencia, de la capacidad de levantarme ante cada desafío y de comprender que la vida siempre ofrece nuevas lecciones, así como de mi compromiso de poner en práctica lo aprendido en beneficio de la humanidad y aportar, desde donde me encuentre desarrollando mis actividades personales o profesionales, a la lucha contra la criminalidad y la delincuencia. Gracias a todos por ser parte esencial de este camino y de este capítulo de mi vida.

Santiago Xavier Ordóñez Rosenberg.

Dedico este trabajo de fin de maestría a mis padres, a mi hermano y a mi pareja, quienes en todo este trayecto han sido un pilar fundamental en mi vida. Su apoyo incondicional, paciencia y sabios consejos me han enseñado que no existen imposibles y que, aun frente a las dificultades, nunca debemos rendirnos, sino dar siempre lo mejor de nosotros. Con profunda gratitud y agradecimiento, les ofrezco este logro, que también les pertenece.

Sebastián Aldhair Tejada Mendoza.

Dedico este trabajo a mis padres y hermanos quienes en cada etapa del camino supieron ayudarme y entenderme, gracias por creer en mí cuando a veces yo dudaba, todos los logros tienen un pedazo de cada uno de ustedes.

Jefferson Alexander Guala Fonseca

Este trabajo de titulación está dedicado a todos aquellos que han sido mi apoyo incondicional a lo largo de este camino.

A Dios que me dio la salud y fortaleza para realizar este proyecto.

A mi esposa María Belén por su apoyo incondicional, paciencia y comprensión que ha sido mi mayor fortaleza durante este desafiante viaje académico.

A mis padres que me brindaron su incondicional apoyo para seguir adelante y a mi hermano Javier, por sus palabras de aliento y motivación.

Alan Patrick Jaramillo Pozo

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a Dios, por darme la fuerza, sabiduría y salud necesaria para poder terminar esta etapa tan importante en mi vida personal y educativa.

A los docentes de la Universidad Internacional del Ecuador (UIDE), por compartir sus conocimientos y orientación para formarme académicamente en este proceso de maestría.

Sebastián Aldhair Tejada Mendoza.

Quiero expresar mi más profundo agradecimiento a mi familia por ser el pilar fundamental, en especial a mis padres que con su enseñanza me guiaron desde niño para no rendirme en cualquier reto que se me presente.

A la UIDE por ofrecerme la oportunidad y facilidades para llevar a cabo este gran proyecto.

Jefferson Alexander Guala Fonseca

Al concluir esta etapa de mi vida académica, deseo expresar mi más sincero agradecimiento a todas las personas que han sido esenciales en este proceso.

Primero y ante todo, agradezco a Dios, por darme la fuerza, sabiduría y perseverancia para enfrentar y superar cada desafío.

A mis padres, hermanos y esposa, por estar siempre a mi lado, brindándome su compañía y palabras de aliento en cada momento.

Alan Patrick Jaramillo Pozo

Quiero expresar mi más sincero agradecimiento a todas las personas que me acompañaron en este camino. A mi tutor/a, por su paciencia, orientación y por confiar en mis capacidades incluso cuando yo mismo dudaba. A mis docentes, por compartir sus conocimientos y sembrar en mí el interés por la investigación y la ciberseguridad.

A mi familia, gracias por su apoyo silencioso pero constante, por sus palabras de ánimo en los momentos difíciles y por recordarme siempre por qué vale la pena continuar.
A mis amigos y compañeros, gracias por escucharme, aconsejarme y celebrar conmigo cada pequeño avance.
¡Este trabajo también les pertenece ¡

Santiago Xavier Ordoñez Rosenberg

RESUMEN

El sistema operativo Android, debido a su amplia adopción y a la heterogeneidad que caracteriza a su ecosistema, se ha convertido en un objetivo recurrente para amenazas cada vez más sofisticadas. Este estudio analiza el comportamiento de distintas variantes de malware dirigidas a Android—incluyendo troyanos bancarios, aplicaciones fraudulentas basadas en phishing y droppers altamente ofuscados como los asociados a la familia SpyNote—mediante un enfoque híbrido que combina análisis estático y dinámico dentro de un laboratorio virtual controlado.

La investigación se llevó a cabo en un entorno seguro configurado con VMware y Android-x86. El análisis estático, realizado con herramientas como MobSF, Jadx y APKTool, permitió examinar la estructura interna de las aplicaciones, certificados, permisos y artefactos incrustados. A su vez, el análisis dinámico, desarrollado mediante FakeNet-NG, Wireshark y ADB, facilitó la observación de interacciones en tiempo real, modificaciones del sistema y actividad genuina de red.

Los resultados revelaron tácticas de evasión avanzadas, como el abuso del Servicio de Accesibilidad para la interceptación de credenciales, el uso de cifrado AES para ocultar comunicaciones internas y la implementación de carga dinámica de código (DCL) para evadir mecanismos de detección. También se identificó el uso de dominios generados dinámicamente (DGA) para ocultar canales de comando y control.

En conjunto, los hallazgos demuestran que el análisis estático, por sí solo, resulta insuficiente para caracterizar amenazas modernas con altos niveles de ofuscación. La observación dinámica es esencial para descubrir comportamientos ocultos y patrones reales de comunicación, reforzando la necesidad de un modelo híbrido para evaluaciones de seguridad más completas y fiables.

Palabras claves: Malware en Android, análisis estático, análisis dinámico, virtualización, inspección de red, evaluación de amenazas.

ABSTRACT

The Android operating system, due to its broad adoption and the heterogeneous nature of its ecosystem, has become a persistent target for increasingly sophisticated cyber threats. This study examines the behavioral dynamics of current Android malware—specifically banking trojans, phishing-based fraudulent applications, and highly obfuscated droppers such as those associated with the SpyNote family—through a hybrid analysis approach conducted within an isolated and fully controlled virtual laboratory.

The methodological framework centered on the creation of a secure test environment using VMware and Android-x86. Static inspection was performed with tools such as MobSF, Jadx, and APKTool, allowing the examination of application structure, certificates, permissions, and embedded artifacts. Complementing this, dynamic monitoring was carried out with FakeNet-NG, Wireshark, and ADB to observe runtime interactions, system modifications, and real network activity.

Across the analyzed samples, several evasion strategies were identified. These included the misuse of the Accessibility Service for credential interception, the implementation of AES-based encryption to obscure internal communications, and the deployment of Dynamic Code Loading (DCL) to bypass signature-based detection. Additionally, the use of Dynamically Generated Domains (DGA) was observed as a mechanism for masking command-and-control channels and reducing traceability.

Taken together, the findings demonstrate that static analysis alone is insufficient to characterize modern threats with high levels of obfuscation. Dynamic observation is essential

to uncover hidden behaviors and real communication patterns, reinforcing the need for a hybrid model for more comprehensive and reliable security assessments.

Keywords: Android malware, static analysis, dynamic analysis, virtualization, network inspection, threat assessment.

TABLA DE CONTENIDOS (Índice)

Contenido

LISTA DE TABLAS (Índice de tablas)

LISTA DE FIGURAS (Índice de figuras)

CAPITULO 1:
INTRODUCCIÓN

1.1. Definición del Proyecto

El sistema operativo de Android es el más usado en el mercado global de dispositivos móviles, lo que paradójicamente lo ha convertido en el vector de ataque preferido para cualquier cibercrimen organizado. Actualmente las amenazas no se limitan a un tipo de virus simple, si no que han evolucionado hacia campañas sofisticadas de phishing, troyanos bancarios y software de espionaje (spyware). Sin embargo, a pesar de ser tendencia este tipo de amenazas, muchas instituciones carecen de procedimientos estandarizados y entornos seguros para inspeccionar estas amenazas, dependiendo excesivamente de excesivamente de antivirus comerciales que suelen fallar entre técnicas modernas de ofuscación.

El presente proyecto de titulación tiene como propósito comprender y analizar con rigor técnico y cuidado ético, cómo actúan los principales tipos de malware en el sistema operativo Android (de manera virtualizada), en el momento de ejecutarlas en condiciones controladas.

La investigación se centra en la problemática desde una perspectiva integral, examinando tanto análisis estático (permisos, manifiesto, código ofuscado) como análisis dinámico, en entornos virtualizados con el sistema operativo Android para observar permisos, artefactos en el sistema y comunicaciones de red de estos. El valor diferencial del proyecto reside a su aplicabilidad practica: validar la necesidad de metodologías de detección hibridas y convertir observaciones técnicas en recomendaciones claras y aplicables para usuarios, desarrolladores y equipos de seguridad.

1.2. Justificación e Importancia del Trabajo de Investigación

La relevancia de este proyecto surge ante la insuficiencia de los mecanismos de defensa automatizados actuales. Si bien los antivirus hacen un muy buen trabajo, el análisis realizado demuestra que técnicas modernas como la ofuscación de código, la carga dinámica (DCL) y la ingeniería social avanzada logran evadir con facilidad estas barreras estáticas. Por lo tanto, aún se siente una ausencia de seguridad en usuarios y organizaciones.

Este proyecto se justifica por la necesidad urgente de dotar a las instituciones una metodología o un procedimiento de análisis forense manual y aprobada. La implementación de un entorno de laboratorio virtualizado permite analizar estas amenazas de una forma más segura, cubriendo la brecha existente entre la alerta y la comprensión técnica del ataque.

1.3. Alcance

Este proyecto de investigación abarca el análisis técnico y forense de aplicaciones maliciosas diseñadas para el sistema Operativo Android. El estudio se limitará a la ejecución de pruebas en un entorno de laboratorio virtualizado y aislado, garantizando la contención de las amenazas y evitando cualquier riesgo de propagación a las redes de producción o en dispositivos móviles personales.

El alcance técnico considera dos dimensiones principales:

Análisis estático: Ingeniería inversa del código fuente (Java), análisis del archivo AndroidManifest.xml, revisión de permisos y validaciones de certificados digitales utilizando herramientas como JADX y MobSF.

Análisis dinámico: Observación del comportamiento del malware al momento de la ejecución para identificar conexiones a servidores de comando y control (C2), exfiltración de datos y manipulación del sistema de archivos.

Esta investigación se centra en específico tres tipos de amenazas representativas del panorama actual:

- Troyanos bancarios: Enfocados en el abuso de servicios de accesibilidad y ataques de superposición (Overlay).
- Aplicaciones fraudulentas (FakeApps): Dirigidas a la ingeniería social y campañas de phishing.
- Herramientas de acceso remoto (RATs/Droppers): Caracterizadas por el uso de ofuscación avanzada y carga dinámica de código (DCL).

Al finalizar el análisis, se elaborará un conjunto de recomendaciones defensivas, orientadas en la detección basada en los hallazgos obtenidos, excluyendo de este alcance el análisis de malware para iOS u otros sistemas operativos, así como el desarrollo de software antivirus.

1.4. Objetivos

1.4.1. Objetivo general

Analizar el comportamiento de malware en entornos Android, mediante el uso de análisis estático y dinámico en entornos controlados, para proponer estrategias de detección y prevención.

1.4.2. Objetivos específicos

- Recopilar muestras de malware recientes desde repositorios con trazabilidad.

- Configurar un laboratorio seguro en VMware (Android).
- Aplicar el análisis estático para identificar permisos, recursos y vectores.
- Realizar análisis dinámico para observar comportamiento en tiempo real (sistema y red).

CAPITULO 2:
Revisión de la Literatura

2.1. Estado del Arte

El ecosistema Android se ha consolidado como el sistema operativo móvil predominante a nivel global, lo que lo convierte en un entorno atractivo para la actividad delictiva digital. Su arquitectura abierta, la diversidad de fabricantes y la posibilidad de instalar aplicaciones desde repositorios externos incrementan de forma significativa la superficie de ataque y la probabilidad de explotación por software malicioso (INCIBE, 2023). Estas características, que impulsaron su crecimiento, han derivado también en la proliferación de amenazas que evolucionan constantemente en complejidad, impacto y mecanismos de evasión.

El malware móvil, definido como un software diseñado para comprometer la seguridad del dispositivo, sustraer información o interferir con su funcionamiento, ha experimentado un proceso de sofisticación acelerado. Sus primeras variantes estaban orientadas a actividades básicas como fraudes por SMS o despliegue de publicidad invasiva. Sin embargo, los actores maliciosos han desarrollado técnicas avanzadas que permiten desde la monitorización de actividad del usuario hasta la obtención de control total del dispositivo mediante Remote Access Trojans (RAT) (Symantec, 2023).

Trend Micro (2023) y Kaspersky Lab (2023) destacan la proliferación de troyanos bancarios capaces de suplantar interfaces legítimas, capturar credenciales en tiempo real y manipular transacciones financieras. Solo en 2023 se registraron cerca de 33.8 millones de ataques a dispositivos Android, con un incremento del 50 % respecto al año anterior, siendo el adware la categoría predominante.

La evolución del malware también se evidencia en técnicas de evasión cada vez más efectivas: ofuscación, cifrado de secciones críticas, empaquetado (packing), verificación del

entorno de análisis, cargas dinámicas y mecanismos anti-emulación. Andrango (2022) identifica estas técnicas como elementos recurrentes en malware contemporáneo, especialmente en familias como SharkBot, SpyNote y Hook, las cuales utilizan arquitecturas modulares que dificultan la aplicación de ingeniería inversa y el análisis de comportamiento.

El documento Tarea Final Ingeniería Inversa refuerza esta idea al demostrar que los malware modernos pueden incluir capas de cifrado, estructuras empaquetadas y manipulación de memoria que dificultan la extracción de payloads mediante análisis superficial. En este estudio se emplearon herramientas como Ghidra, x64dbg y PE-bear para la inspección de ejecutables, evidenciando la relevancia de aplicar técnicas de ingeniería inversa adaptables al ecosistema Android.

Asimismo, investigaciones recientes orientadas al Internet de las Cosas (IoT), como el estudio del dispositivo medidor de glucosa (Montenegro et al., 2025), muestran que las vulnerabilidades asociadas a aplicaciones móviles conectadas influyen directamente en la seguridad física y digital de los usuarios. Esta auditoría demostró que fallas en permisos, comunicaciones Bluetooth y diseño de firmware pueden ser explotadas para obtener datos sensibles. Su metodología reafirma la utilidad del análisis estático, dinámico y forense digital como pilares para entender no solo el comportamiento del malware, sino también el riesgo asociado a dispositivos complementarios (Vaca & Valle, 2024; Cervera & Goussens, 2024).

A pesar de los esfuerzos de Google por reforzar su ecosistema mediante herramientas como Play Protect, sandboxing y controles de permisos, la existencia de mercados alternativos, firmware modificable y brechas de actualización continúan representando vectores de riesgo (Google Research, 2024). Organismos como INTERPOL (2023), OEA (2023) y Europol (2023) advierten que el malware móvil ya no es un fenómeno aislado, sino parte de redes criminales organizadas que utilizan modelos como malware-as-a-service, automatización de campañas y arquitecturas escalables de Comando y Control (C2).

En conjunto, la literatura demuestra que el malware en Android representa una amenaza activa, adaptable y en expansión, lo que justifica la necesidad de entornos de análisis controlados, metodologías híbridas y evaluaciones continuas para comprender su comportamiento y mitigar sus efectos.

2.2. Marco Teórico

2.2.1. Metodología para el Análisis de Malware en un Ambiente Controlado

El análisis de malware en dispositivos Android implica un conjunto de técnicas orientadas a comprender su estructura, comportamiento y mecanismos de evasión. En este contexto, resulta fundamental establecer un marco conceptual que abarque tanto la clasificación del malware como las herramientas y normas que orientan el proceso de análisis.

Diversas fuentes clasifican al malware en categorías como troyanos, spyware, ransomware, keyloggers, rootkits y RATs, cada una con mecanismos particulares de infección, persistencia y exfiltración (Kaspersky, 2024; Regan & Belcic, 2022). En el contexto Android, la presencia de permisos sensibles y la interacción con componentes del sistema incrementan el potencial de impacto, ya que el dispositivo almacena información personal, financiera y biométrica del usuario.

La ingeniería inversa constituye un componente central en el análisis de malware. Herramientas como Ghidra, Jadx y APKTool permiten descompilar aplicaciones, visualizar clases, extraer cadenas sospechosas e identificar comportamientos ocultos. El uso de estas herramientas también se observa en investigaciones como la Tarea Final Ingeniería Inversa, donde se demuestra la importancia de comprender el flujo lógico interno del software para identificar funciones anómalas, rutas de ejecución y mecanismos de cifrado.

El análisis estático y dinámico no se conciben como técnicas independientes, sino como enfoques complementarios. El análisis estático proporciona un primer acercamiento a

la estructura interna del APK sin ejecutarlo, permitiendo identificar permisos, componentes, certificados y patrones sospechosos. Por su parte, el análisis dinámico revela el comportamiento del malware al interactuar con el sistema operativo, brindando información sobre conexiones externas, persistencia y ejecución de código en memoria (MISTIC TFM, 2021).

Para categorizar tácticas, técnicas y procedimientos (TTPs) utilizados por actores maliciosos, marcos como CERT (2021) y Rapid7 (2024) ofrecen lineamientos que permiten estructurar el análisis. Asimismo, normas internacionales como ISO/IEC 27001, 27037 y 27043 establecen buenas prácticas para preservar evidencias, gestionar riesgos y ejecutar análisis forense de forma responsable y controlada.

Finalmente, el estudio del malware debe considerarse dentro de un marco ético y legal. EC-Council (2022) señala la importancia de realizar pruebas exclusivamente en entornos aislados, evitando cualquier impacto en sistemas productivos o redes reales. Estas prácticas garantizan que el análisis no genere daños colaterales y que la seguridad del entorno de investigación se mantenga íntegra.

La metodología para analizar malware en Android se basa en un enfoque híbrido que integra análisis estático, análisis dinámico y la utilización de un entorno controlado. Esta aproximación permite observar tanto la estructura del código como su comportamiento real, reduciendo riesgos y aumentando la precisión del diagnóstico.

1. Análisis Estático: Ingeniería Inversa sin Ejecución. El análisis estático se enfoca en examinar la estructura interna del archivo APK sin ejecutarlo. Esta etapa permite identificar permisos sensibles en el AndroidManifest.xml, detectar componentes exportados que podrían ser explotados, extraer cadenas de texto, revisar certificados y evaluar mecanismos de ofuscación y empaquetado. Herramientas como Jadx, APKTool y MobSF automatizan parte del proceso, permitiendo visualizar clases Java, recursos internos y

patrones sospechosos. Tal como se evidencia en la ingeniería inversa, esta técnica resulta indispensable para comprender la lógica de un software malicioso, especialmente cuando emplea cifrado o mecanismos de empaquetado complejos que dificultan su análisis superficial.

No obstante, el análisis estático es limitado frente a malware que utiliza técnicas avanzadas como carga dinámica de código, cifrado de payloads, reflexión y verificación del entorno de ejecución. En estos casos es necesario complementarlo con análisis dinámico para obtener una visión más precisa del comportamiento real de la amenaza.

2. Análisis Dinámico: Observación del Comportamiento Real. El análisis dinámico consiste en ejecutar el malware dentro de un entorno aislado para monitorear su interacción con el sistema operativo. Esta fase permite identificar comportamientos que no pueden descubrirse mediante análisis estático, como conexiones a servidores externos, exfiltración de datos, creación de procesos persistentes, activación de sensores o ejecución de código desencriptado en memoria. Herramientas como Frida, Wireshark y FakeNet-NG permiten interceptar funciones, capturar tráfico y simular servicios receptores. Cuckoo Sandbox (2022) documenta que el malware moderno utiliza técnicas anti-análisis que exigen entornos altamente realistas para evitar falsos negativos.

3. Laboratorio Aislado: Contención y Control Total. El laboratorio debe ser completamente aislado para evitar fugas del malware. Según las prácticas recomendadas por ISO/IEC 27037 y las auditorías de Montenegro et al. (2025), un entorno adecuado incluye máquinas virtuales en VMware o VirtualBox, un sistema Android-x86 independiente, una red LAN segmentada sin acceso a Internet real, una máquina secundaria para monitoreo de tráfico, deshabilitación de carpetas compartidas y portapapeles, así como la simulación de sensores y actividad del usuario. Estas medidas permiten prevenir la propagación accidental del malware y asegurar la correcta preservación de evidencia.

4. Extracción de Indicadores de Compromiso (Iocs). El resultado del análisis híbrido es la obtención de indicadores de compromiso (IoCs) como hashes de archivos, direcciones IP, dominios sospechosos, certificados irregulares, cadenas cifradas y rutas internas de exfiltración. IBM (2023) y Gartner (2024) subrayan que estos indicadores son clave para alimentar sistemas SIEM, IDS/IPS y XDR, mejorando la capacidad de detección y la respuesta frente a incidentes de seguridad.

2.3. Perspectivas Actuales y Tendencias Emergentes

La evolución del malware en Android refleja la creciente dependencia de la sociedad en los dispositivos móviles como núcleo de la vida diaria. Estos dispositivos gestionan datos financieros, información sensible, comunicaciones personales, acceso a servicios institucionales y control de dispositivos IoT. Por ello, las amenazas actuales no solo buscan comprometer un dispositivo aislado, sino insertarse dentro de ecosistemas digitales completos.

Informes recientes resaltan que muchas familias de malware incorporan módulos de automatización que permiten adaptar su comportamiento según el contexto del dispositivo, reduciendo las probabilidades de detección. Esto incluye la activación de payloads solo cuando el dispositivo cumple condiciones específicas como la presencia de apps bancarias o señales de actividad real del usuario (Europol, 2023).

Asimismo, comienzan a observarse muestras que utilizan algoritmos básicos de inteligencia artificial para clasificar actividad del usuario o predecir el momento más oportuno para ejecutar acciones maliciosas. Aunque en fases iniciales, esta tendencia representa un desafío importante, ya que dota al malware de conductas adaptativas difíciles de anticipar.

Tendencias como el abuso de servicios de accesibilidad, el secuestro de interfaces y el control remoto mediante RATs continúan aumentando. En ambientes clínicos e IoT como el

medidor de glucosa analizado por Montenegro et al. (2025), este tipo de ataque puede comprometer datos vitales, poniendo en riesgo la integridad del paciente y la fiabilidad del dispositivo.

Finalmente, se observa un crecimiento del enfoque Zero Trust aplicado al análisis de malware. Este modelo rechaza la confianza implícita en cualquier software y fomenta ambientes de laboratorio más rígidos y segmentados, donde cada acción puede ser monitoreada y verificada. Esta tendencia se alinea con las normativas ISO/IEC 27001, 27037 y 27043 para entornos de investigación seguros y reproducibles.

CAPITULO 3:

DESARROLLO

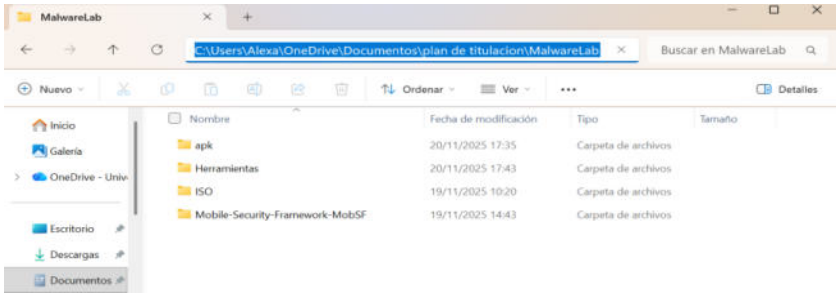
3.1 Desarrollo del Trabajo

Para el estudio de análisis de malware en dispositivos Android, es necesario utilizar un laboratorio seguro, empezamos preparando el host (Windows), en la carpeta C, se creó una carpeta llamada MalwareLab.

3.2 Preparación de ambiente

Figura 1

Carpeta aislada



Nota. En el disco C creamos la carpeta MalwareLab en donde almacenamos las herramientas que usaremos.

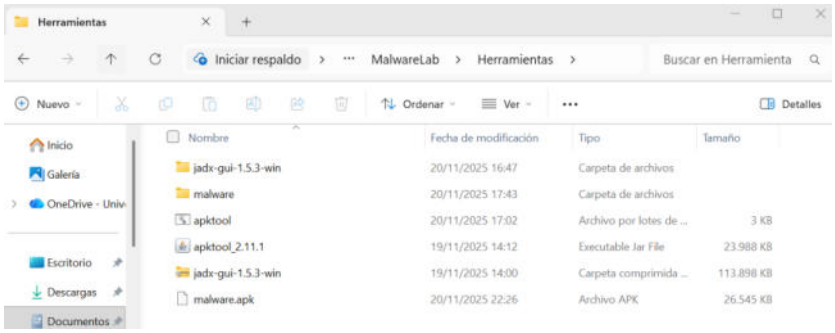
Dentro de la carpeta de MalwareLab crearemos la carpeta herramientas, es en donde almacenaremos las herramientas que usaremos:

Para análisis estático:

- Jadx: https://github.com/skylot/jadx/releases
- Apktool: https://ibotpeaches.github.io/Apktool/
- MobSF: https://github.com/MobSF/Mobile-Security-Framework-

Figura 2

Carpeta aislada



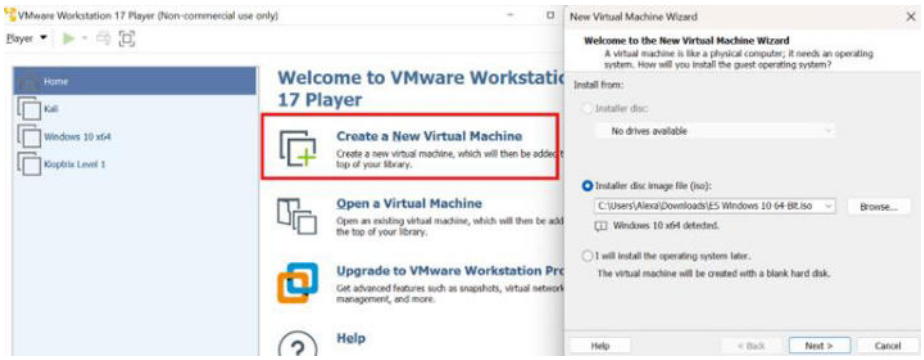
Nota. Dentro de la carpeta MalwareLab creamos la carpeta herramientas es en donde almacenamos las herramientas descargadas para el análisis estático.

Para este laboratorio se va a utilizar VMware, es más estable que VirtualBox y soporta mejor la aceleración grafica.

Creamos una nueva máquina virtual.

Figura 3

Nueva máquina virtual

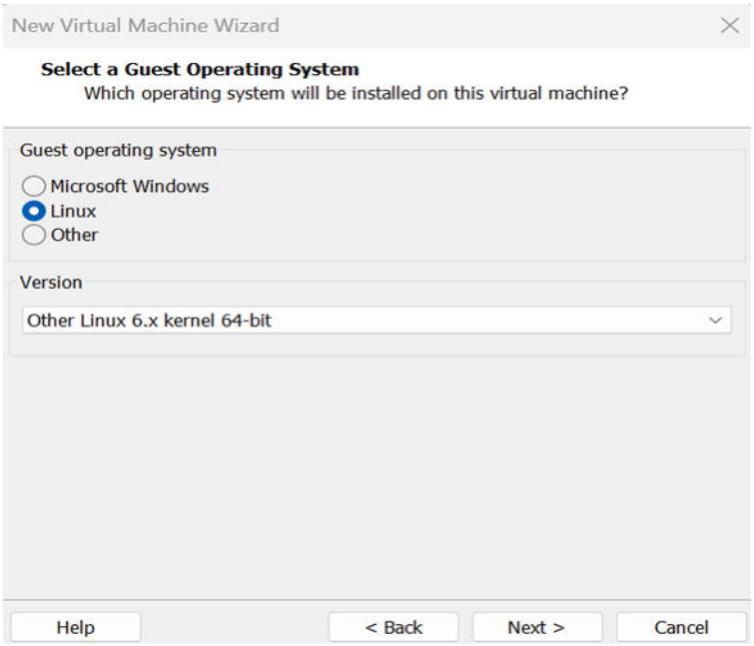


Nota. Pantalla de inicio al crear una nueva máquina virtual

Elegimos el tipo de sistema operativo: Other Linux x64-bit

Figura 4

Tipo de sistema operativo

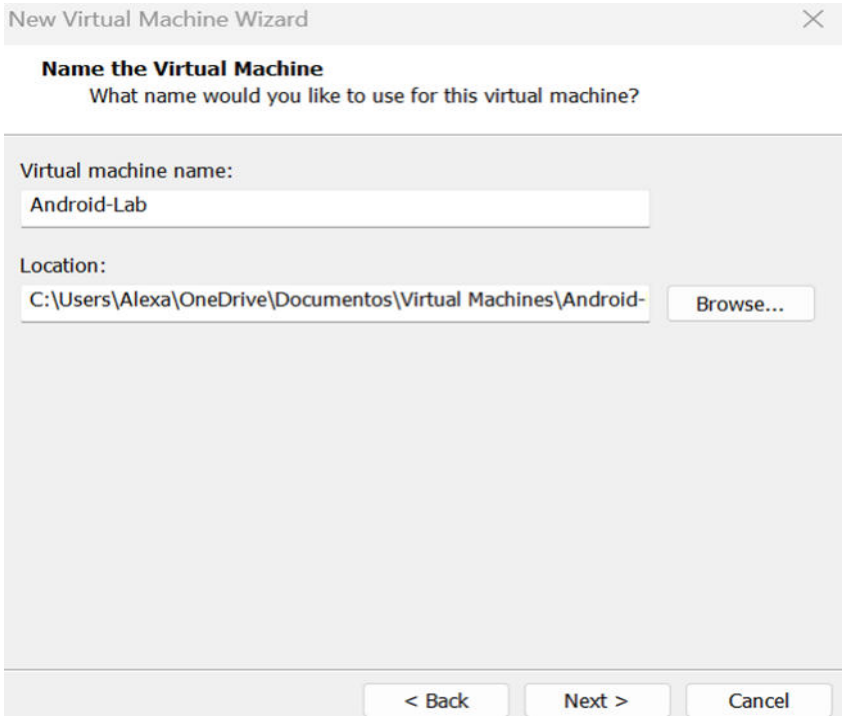


Nota. Seleccionamos other linux 64-bit

Configuramos el nombre la máquina virtual como: Android-Lab.

Figura 5

Nombre de la máquina virtual

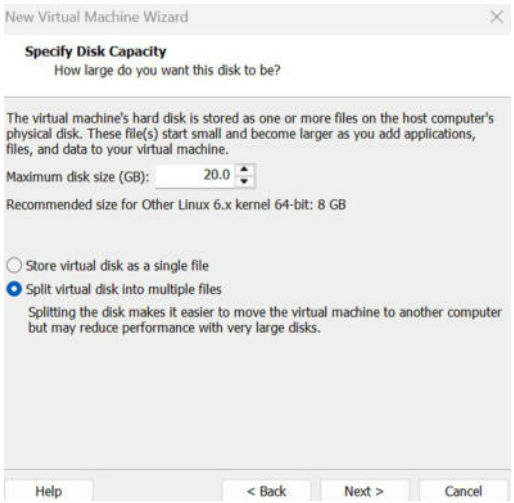


Nota. Configuramos como Android-Lab

Configuramos el tamaño máximo de disco en 20 GB

Figura 6

Tamaño de disco

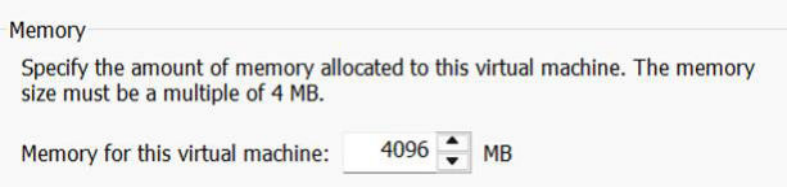


Nota. Tamaño máximo de disco 20 GB

Se configura 4096 MB para la memoria para la máquina virtual.

Figura 7

Memoria



Nota. Se establece para la memoria 4096 MB

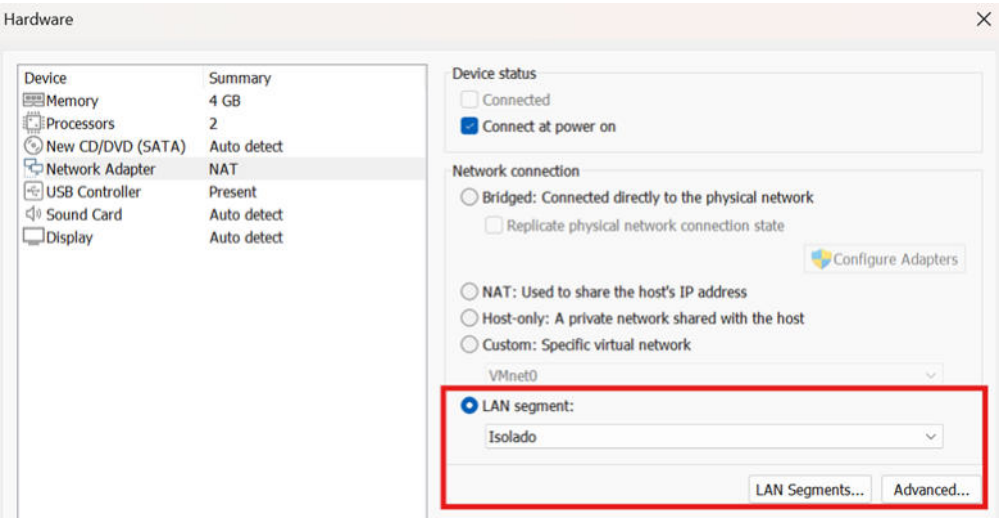
3.3 Configuración de Seguridad

Nos dirigimos a Network adapter seleccionamos LAN Segment; se creó uno llamado "isolado" esto garantiza:

- No hay internet
- No puede escapar a la LAN
- No hay contacto con Windows host

Figura 8

Network adapter

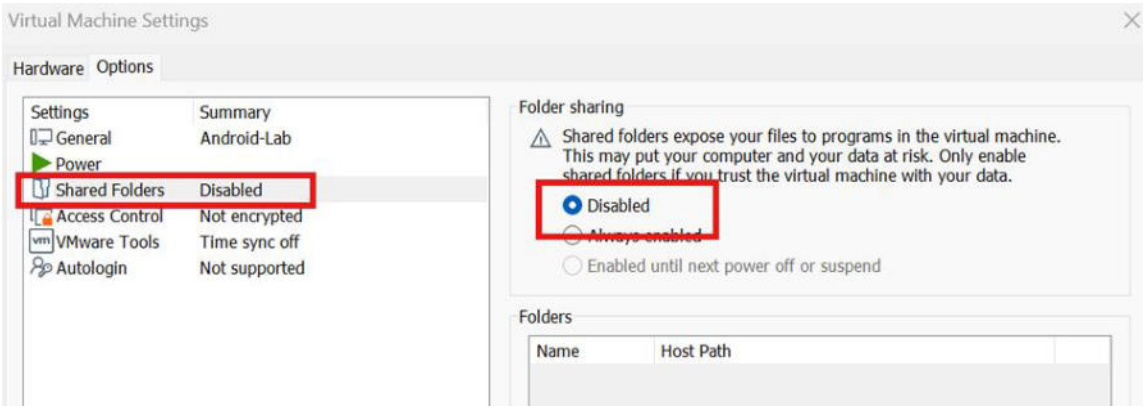


Nota. Se crea un segmento LAN llamado "Isolado"

Deshabilitamos arrastrar y soltar y quitamos carpetas compartidas.

Figura 9

Carpeta compartida

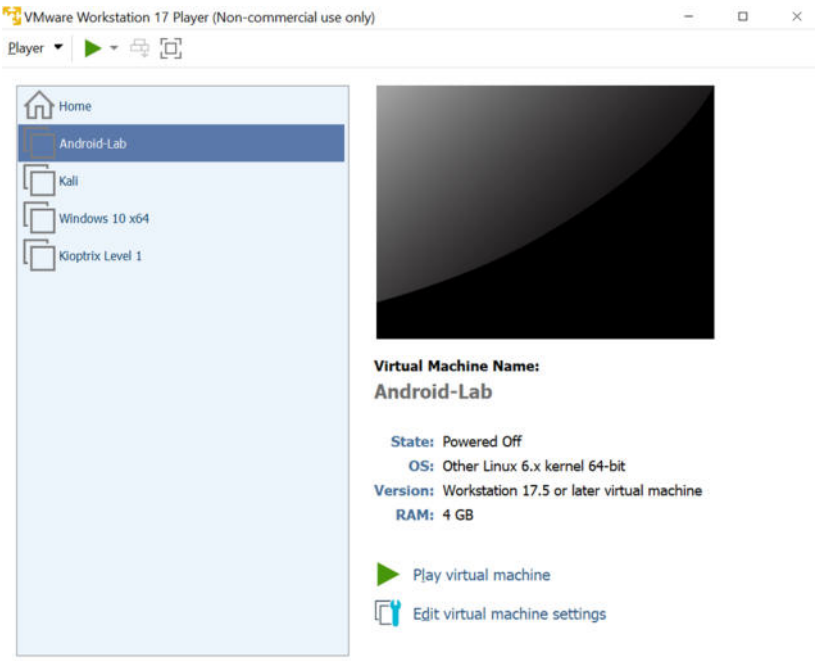


Nota. Deshabilitar carpeta compartida.

Con los pasos anteriores se crea la máquina virtual y está lista para instalar Android.

Figura 10

Estado de la máquina virtual.



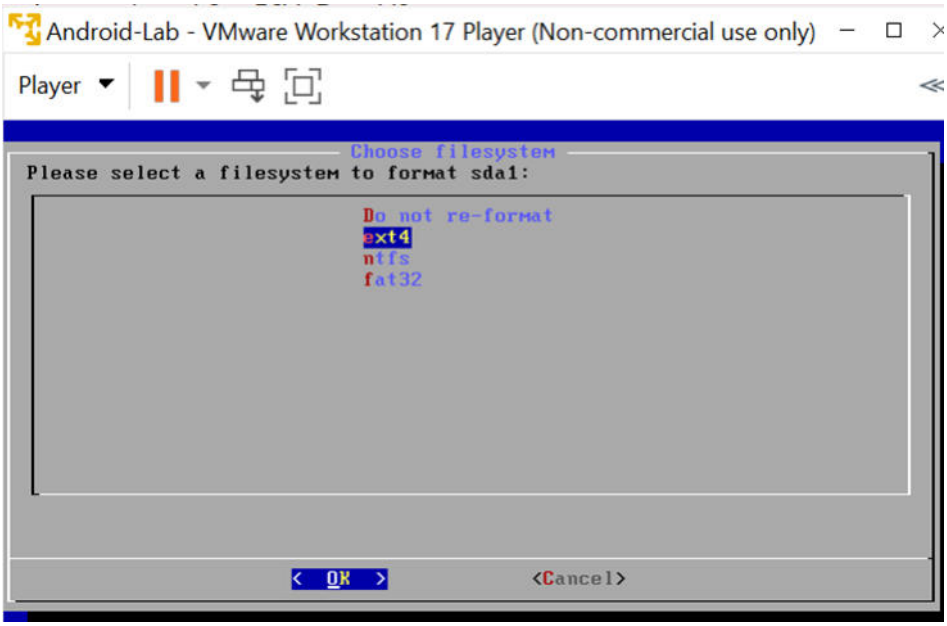
Nota. Nueva máquina virtual

3.4 Instalación de Android en VMware

Se procede a iniciar con la imagen ISO de Android x84 y creamos una partición en el disco virtual ext4.

Figura 11

Instalación de Android

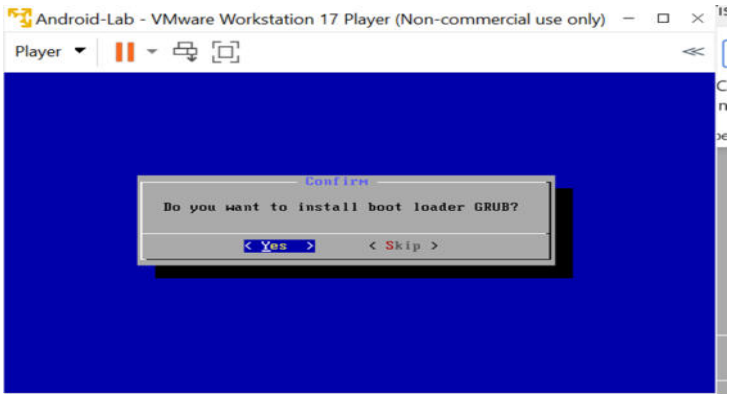


Nota. Seleccionamos la partición en el disco ext4

Se procede a instalar GRUP

Figura 12

Ventana de instalación de GRUP

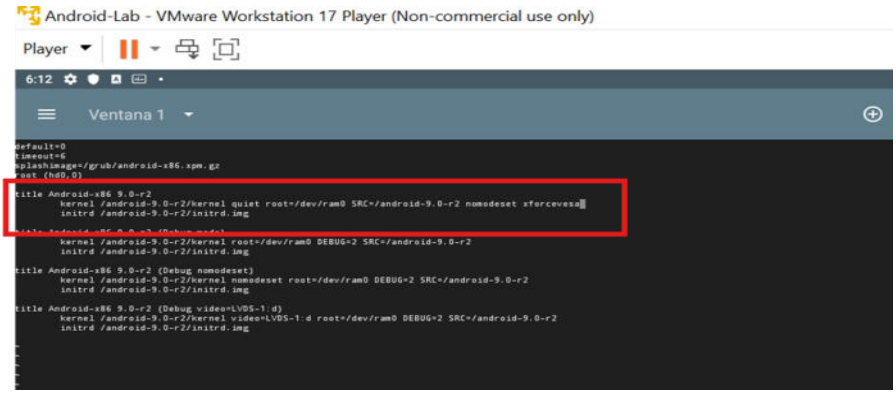


Nota. Seleccionamos YES para instalar GRUP

Al finalizar la instalación de Android y reiniciar el Android no inicio correctamente, por defecto, Android-x86 intenta usar aceleración de hardware para mostrar los gráficos. Sin embargo, VMware utiliza un adaptador de pantalla virtual que a veces no se entiende bien con los drivers nativos de Android. Por tal motivo se hizo una modificación específica en el archivo de arranque (GRUB) por la razón técnica fundamental: evitar la "Pantalla Negra" o que el sistema se congele al iniciar la interfaz gráfica.

Figura 13

Archivo de arranque GRUP

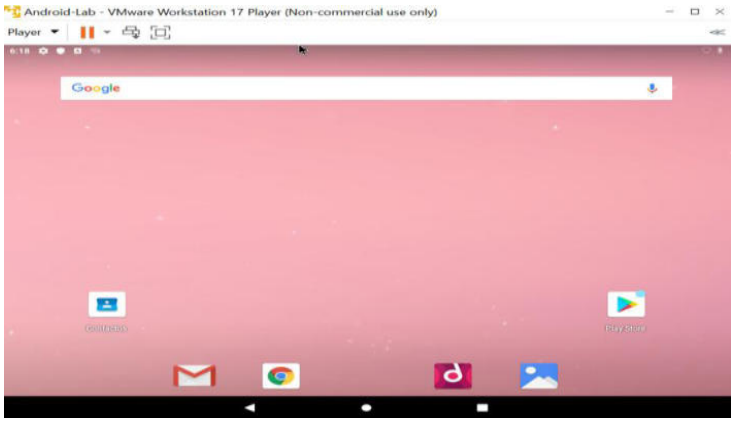


Nota. Se modifico en el archivo GRUP con "nomodeset xforcevesa"

Y finalmente podemos visualizar que la máquina de Android se encuentra corriendo correctamente.

Figura 14

Android en VMware



Nota. Estado inicial de Android en VMware.

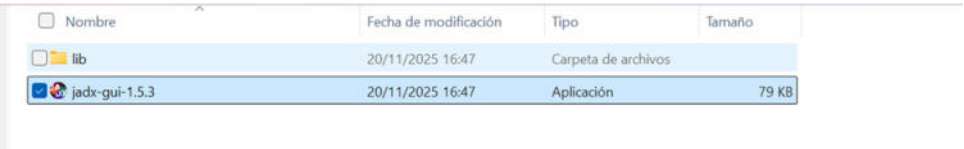
3.5 Análisis Estático

Para comenzar con el análisis estático es necesario instalar las herramientas de análisis en Windows (local host).

- JADX: Extraemos y abrimos la aplicación

Figura 15

Carpeta extraída de JADX

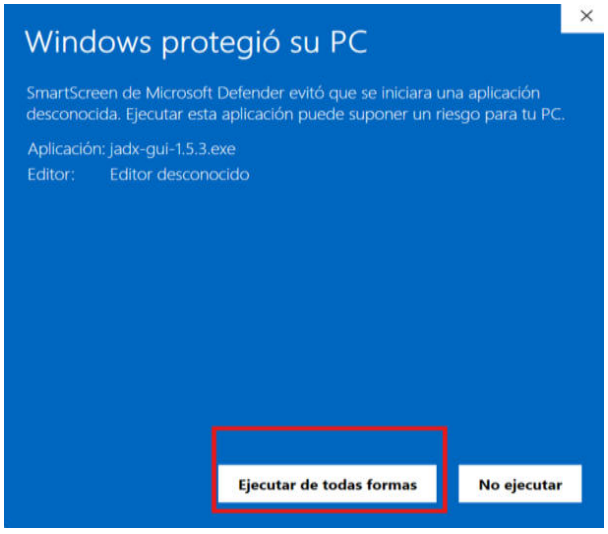


Nota. Herramienta JADX descargada en la carpeta.

Nos procede a saltar una alerta de seguridad, sin embargo, debemos ejecutar de todas formas el aplicativo.

Figura 16

Alerta de Windows al abrir JADX

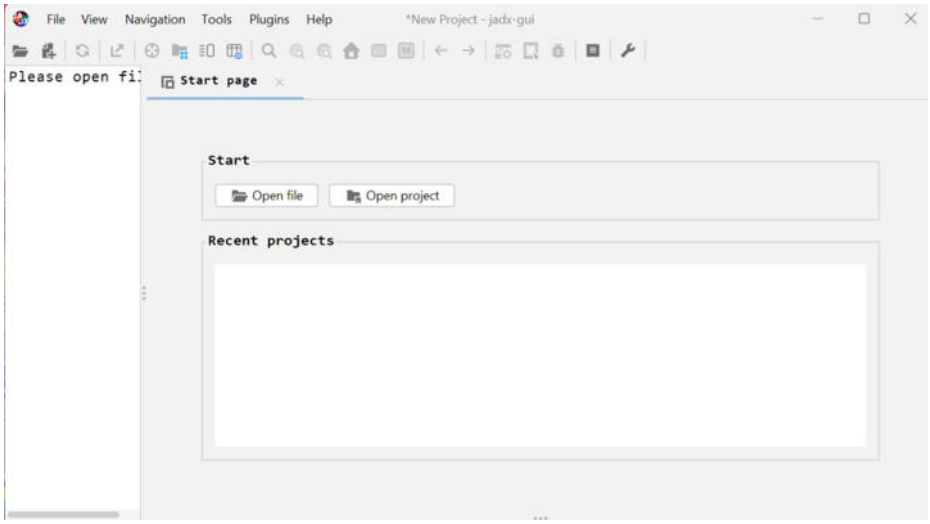


Nota. Alerta de Windows, seleccionamos ejecutar de todas formas.

Una vez realizado esto, ya se puede visualizar la página de inicio de la herramienta JADX en nuestro equipo.

Figura 17

Pantalla de inicio de JADX

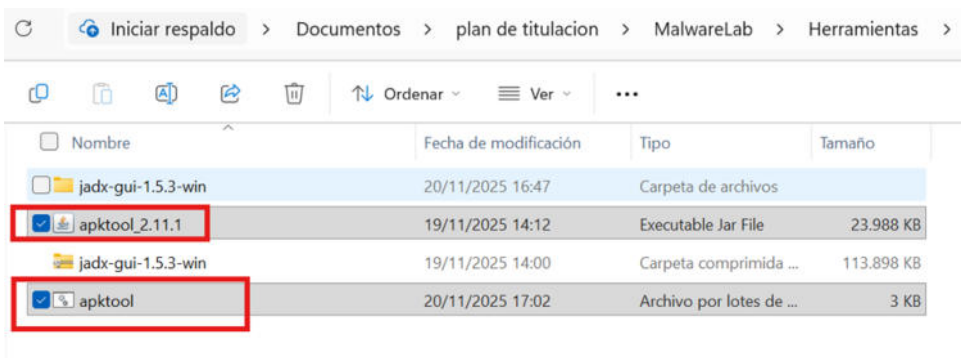


Nota: Home Page de JADX.

- ApkTool: Descargamos ApkTool.bat y ApkTool.jar y movemos a la carpeta.

Figura 18

Carpeta que contiene ApkTool



Nota. Carpeta del ApkTool descargada.

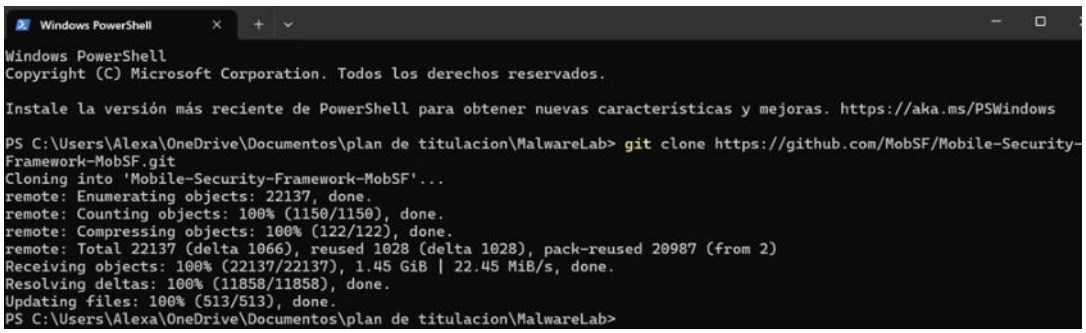
MOBSF: Para la instalación de MOBSF, debemos tener en cuenta lo siguiente: Requerimientos previos para la instalación:

- *Install Git*
- *Install Python 3.10+*
- *Install JDK 8+*
- *Install Microsoft Visual C++*
- *Install Open SSL*

En la carpeta MalwareLab clonamos el repositorio.

Figura 19

Clonar repositorio

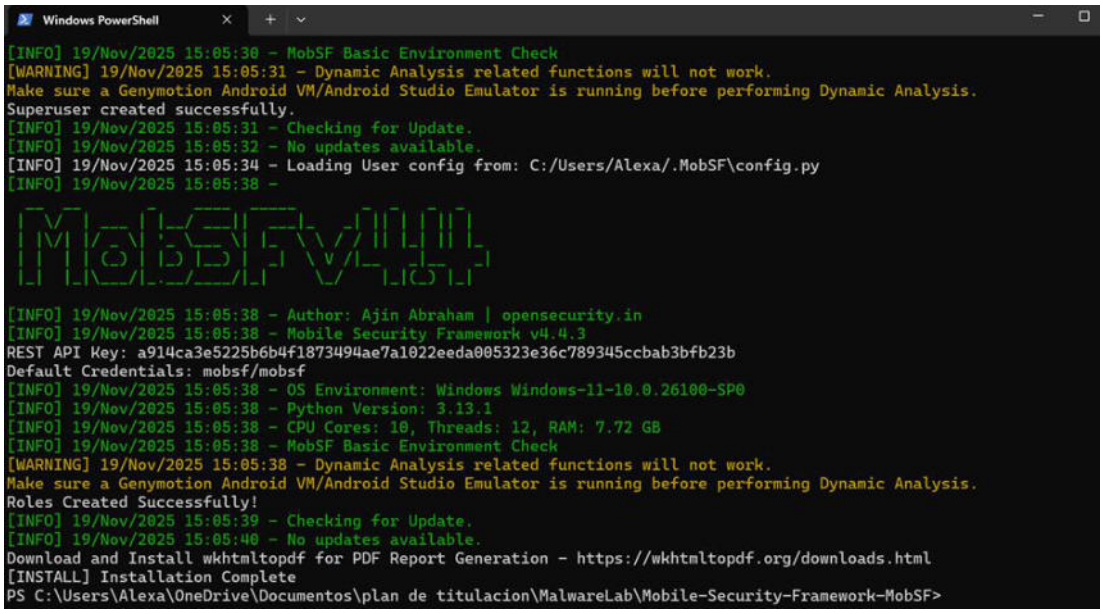


Nota. Se clonó con el comando: git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git

Nos dirigimos a cd Mobile-Security-Framework-MobSF y ejecutamos el comando “Python manage.py runserver”

Figura 20

Instalación de MobSF completa



Nota. Proceso de instalación MobSF

Ejecutamos Run.bat 127.0.0.1:8000

Figura 21

Run server

Nota. Proceso de instalación MobSF

Entramos en el navegador a http://127.0.0.1:8000 e ingresamos con user and pss: mobsf

Figura 22

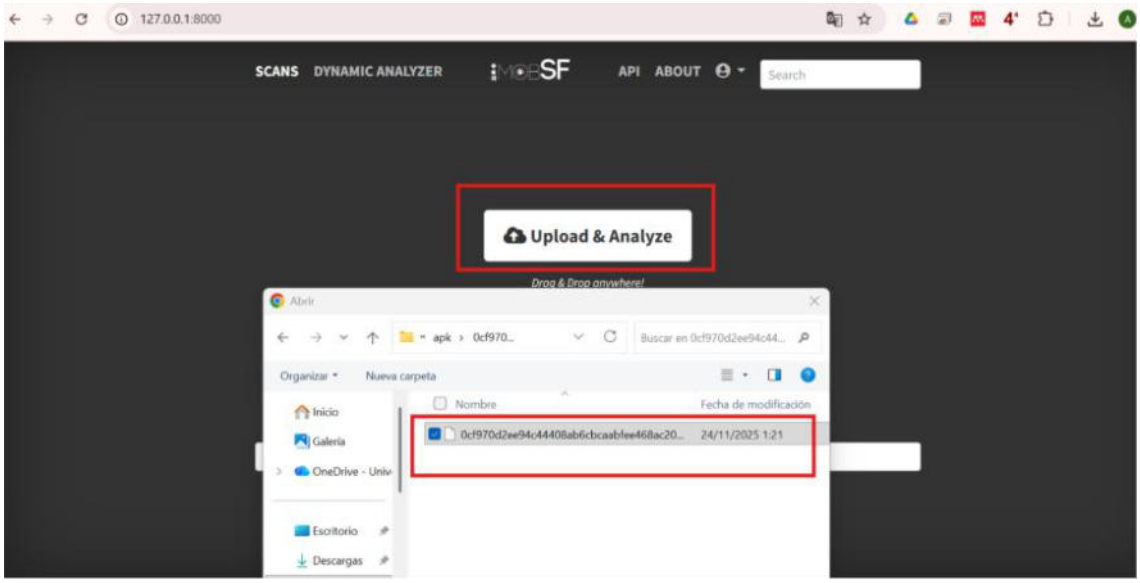
Portal de MobSF

Nota. Portal para cargar el archivo.

Para empezar el análisis estático cargamos el archivo apk muestra, seleccionamos el botón “Upload & Analyze”para subir el archivo respectivo.

Figura 23

Carga del apk en MobSF



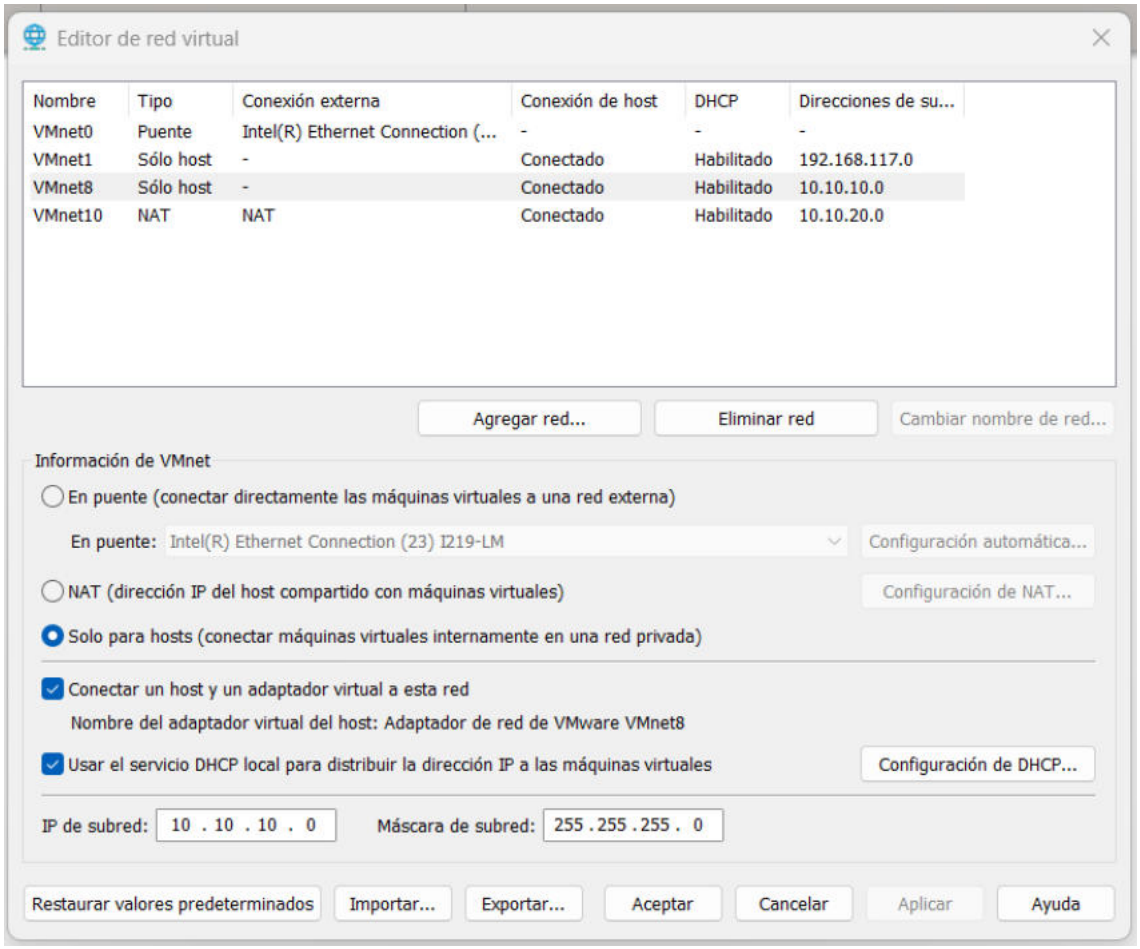
Nota. Seleccionamos la muestra apk que vamos a revisar.

3.6 Análisis Dinámico

Para iniciar con el análisis dinámico es necesario realizar la configuración, de red en el VMware, en este caso en Kali Linux se usan dos interfaces de red, la primera para monitorear la actividad y la segunda (en caso de necesitarse) se conectará a internet por cualquier instalación adicional.

Figura 24

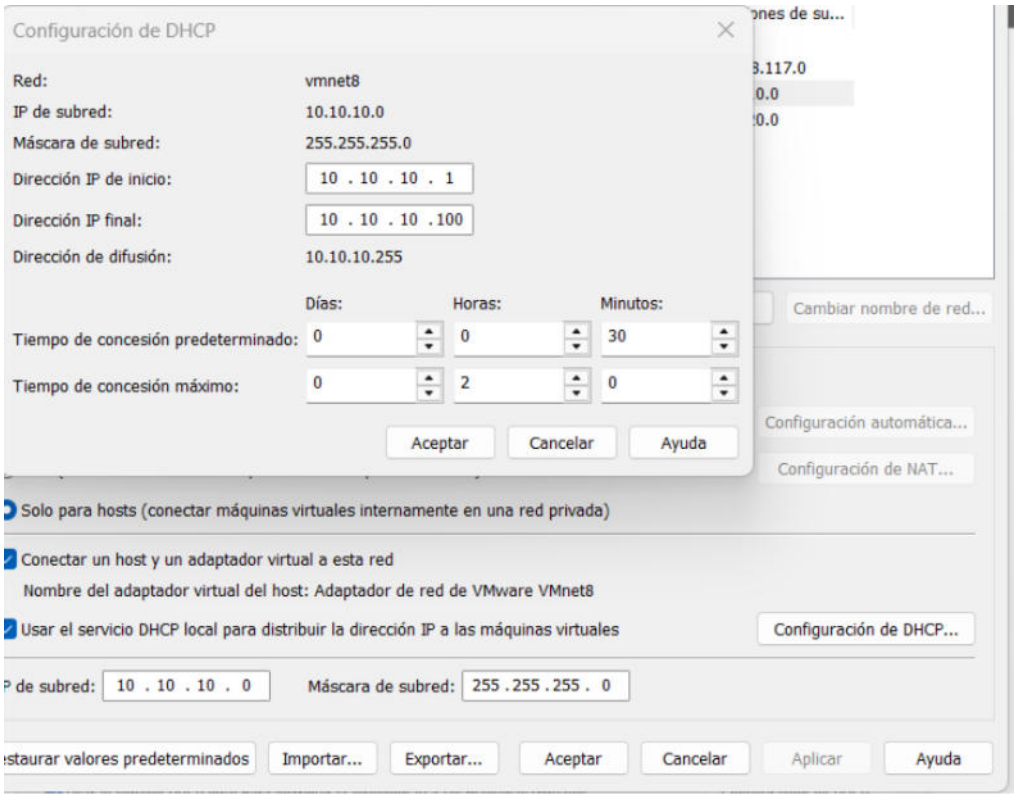
Configuración de interfaces de red en VMware



Nota. Configuración de interfaces utilizadas en el laboratorio en VMware.

Figura 25

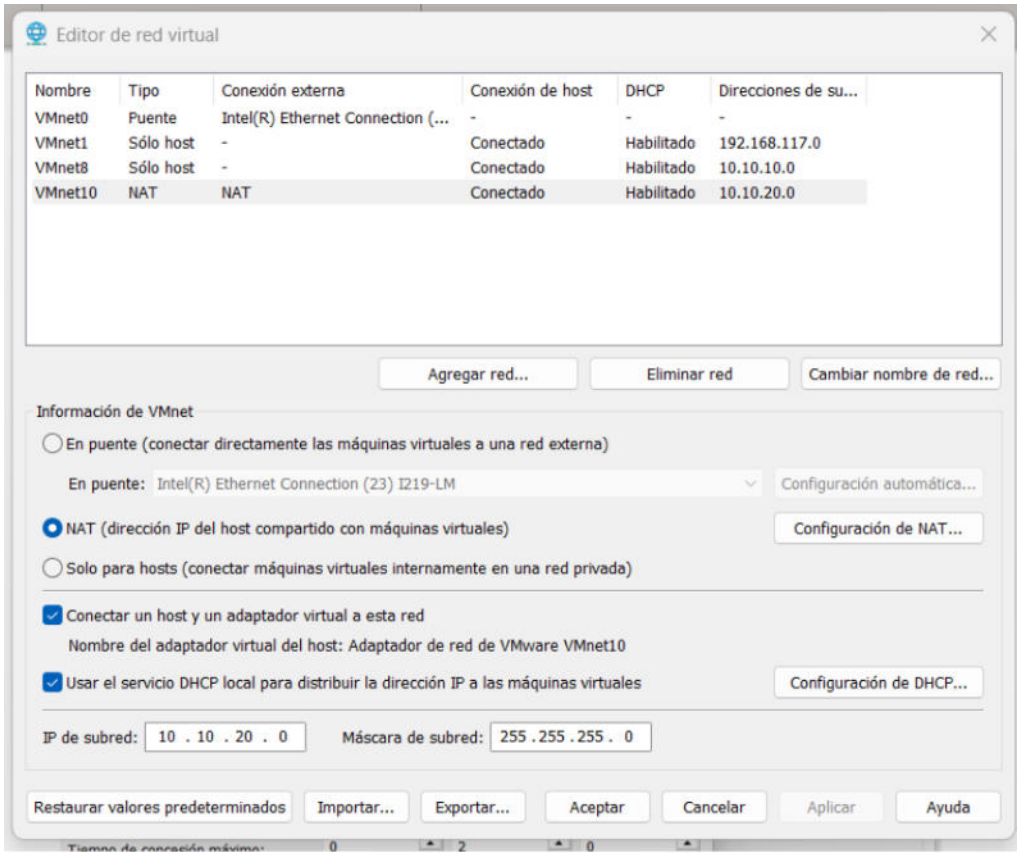
Configuración de interfaces de red en VMware



Nota. Configuración de interfaces utilizadas en el laboratorio en VMware.

Figura 26

Configuración de interfaces de red en VMware



Nota. Configuración de interfaces utilizadas en el laboratorio en VMware.

Figura 27

Configuración de interfaces de red en VMware

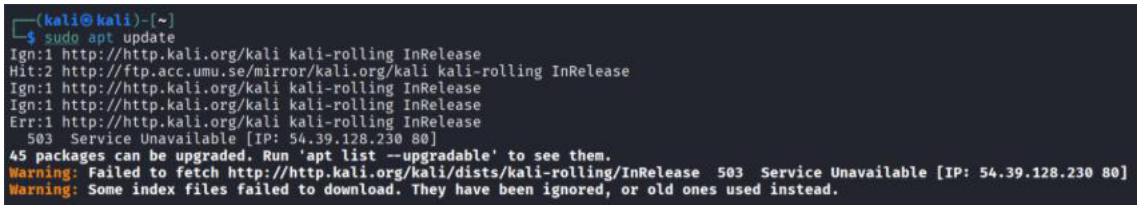
Nota. Configuración de interfaces utilizadas en el laboratorio en VMware.

Una vez configurada la red, es necesario realizar una actualización en la máquina virtual de Kali, para poder instalar los paquetes necesarios para realizar el análisis dinámico.

FIGURA 26

Figura 28

Actualización de Kali para instalar las aplicaciones necesarias



Nota. Actualización de Kali para instalar las aplicaciones del laboratorio.

Para el análisis dinámico se utilizará la herramienta Wireshark, que según Vásquez (2013), esta herramienta es un analizador de paquetes de red el cual intentará capturar paquetes de red y tratar de mostrar los paquetes de datos de manera detallada.

Usamos el siguiente comando: sudo “apt install inetsim wireshark”, para poder instalar INETSim (simulador de servicios de Internet) y Wireshark (analizador de tráfico de red).

Instalación de Wireshark

Figura 29

Instalación de Wireshark

Nota. Instalación de Wireshark.

Del mismo modo se utilizan los comandos para instalar la gestión de paquetes de Python3, descargar (clonar) el repositorio FakeNet-NG desde GitHub en la carpeta actual. Tavella (2022), en su artículo Herramientas de Linux para el análisis de malware, indica que FakeNet-NG es una herramienta freeware usada para el análisis de malware que sirve para capturar paquetes de tráfico de red.

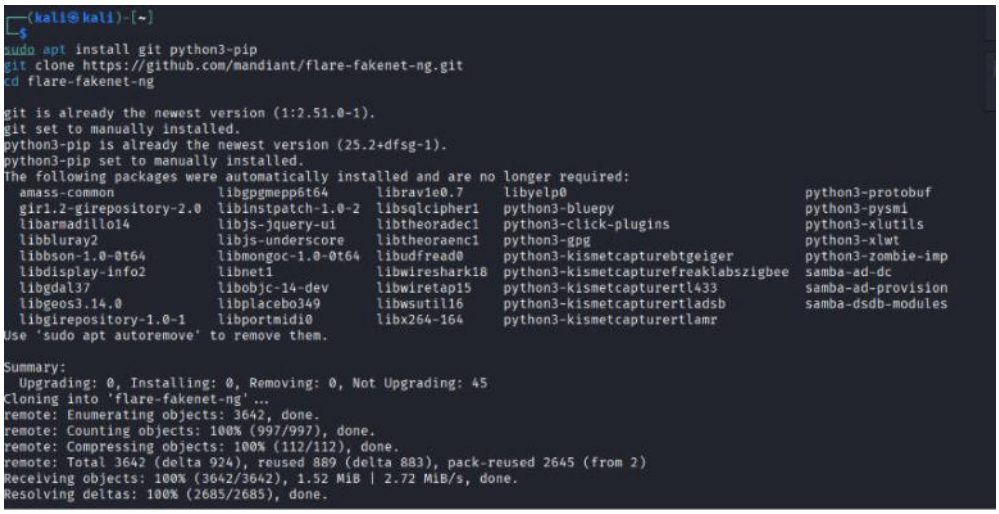
sudo apt install git python3-pip

git clone https://github.com/mandiant/flare-fakenet-ng.git

cd flare-fakenet-ng

Figura 30

Instalación de phyton y clonación de fakenet

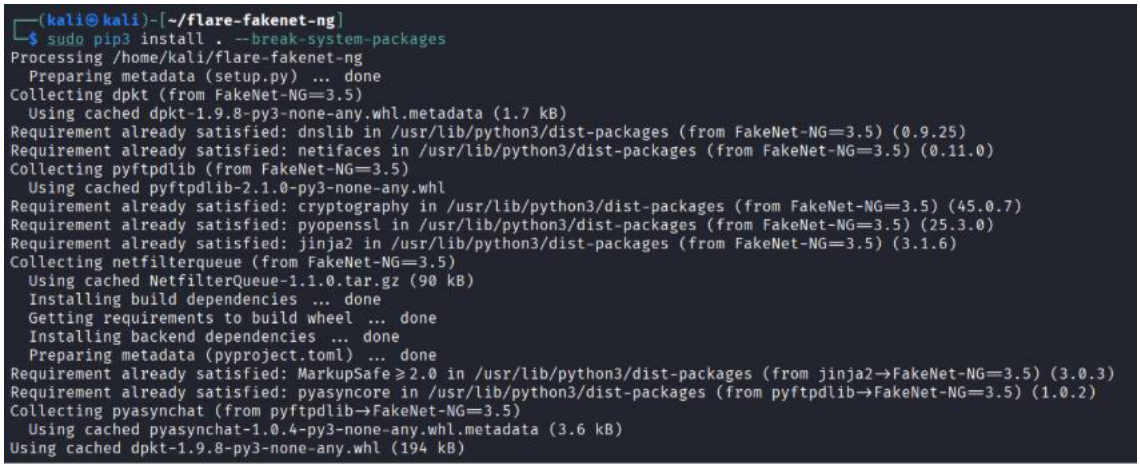


Nota. Instalación de phyton y clonación de fakenet

Una vez instalado el Fakenet, es necesario instalar sus dependencias con el comando: “sudo pip3 install . --break-system-packages “

Figura 31

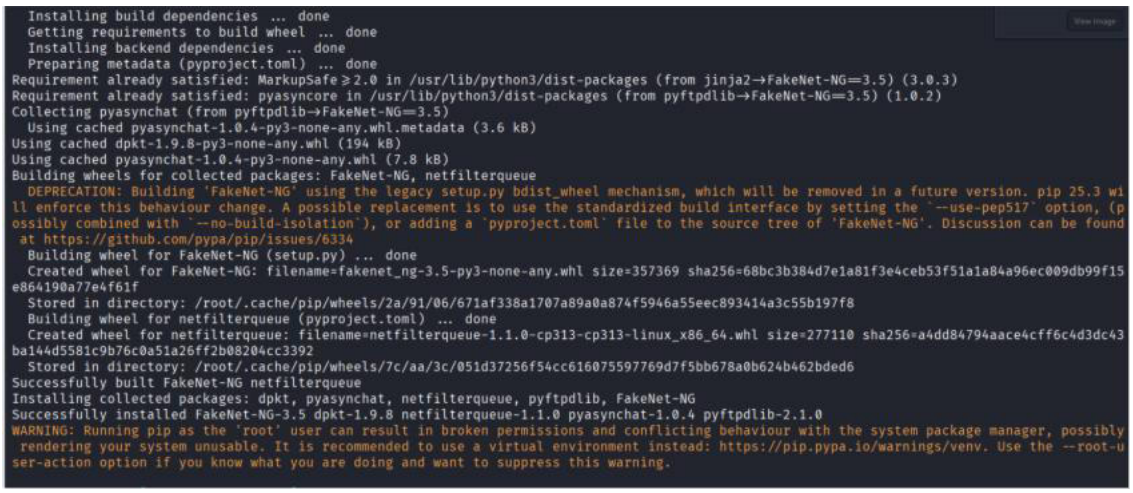
Instalación de phyton y clonación de fakenet



Nota. Instalación de phyton y clonacion de fakenet

Figura 32

Captura del progreso de la instalación y la clonación de fakenet



Nota. Progreso de la instalación y la clonación de fakenet

Figura 33

Ejecutamos el comando sudo Fakenet, para inicializar la herramienta.

Nota. Ejecutamos el comando sudo Fakenet, para inicializar la herramienta.

3.7 Configuración de red de las máquinas virtuales

A continuación, se muestra las configuraciones de red que se van a utilizar en las 3 máquinas virtuales.

Nota: No se va a configurar el gateway para mantener el laboratorio aislado de Internet. Sin embargo, existirá comunicaciones entre las máquinas mediante el DNS.

Tabla 1

Configuraciones de red de las 3 máquinas virtuales.

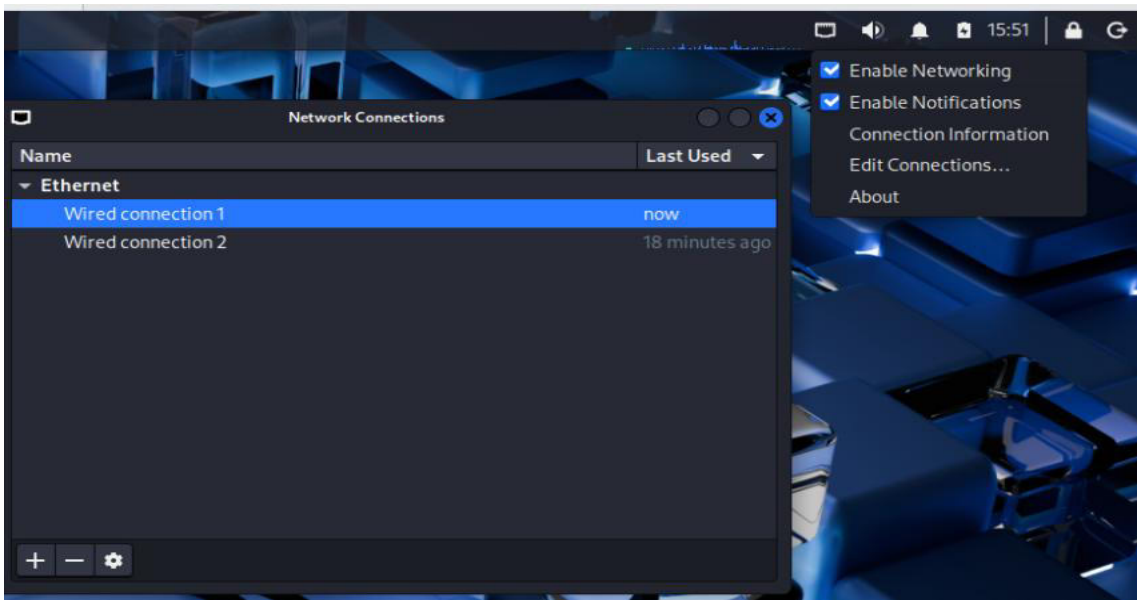
Máquina	IP	Máscara	Gateway	DNS
Kali	10.10.10.10	255.255.255.0	(vacío)	127.0.0.1 o 10.10.10.10
Windows 10	10.10.10.30	255.255.255.0	(vacío)	10.10.10.10
Android x86	10.10.10.20	255.255.255.0	(vacío)	10.10.10.10

Nota. Configuraciones de red de las 3 máquinas virtuales.

Para comenzar la configuración de la IP de la máquina de Kali, es necesario ir a la configuración de red (ícono de red) y seleccionar la opción Wired Connection.

Figura 34

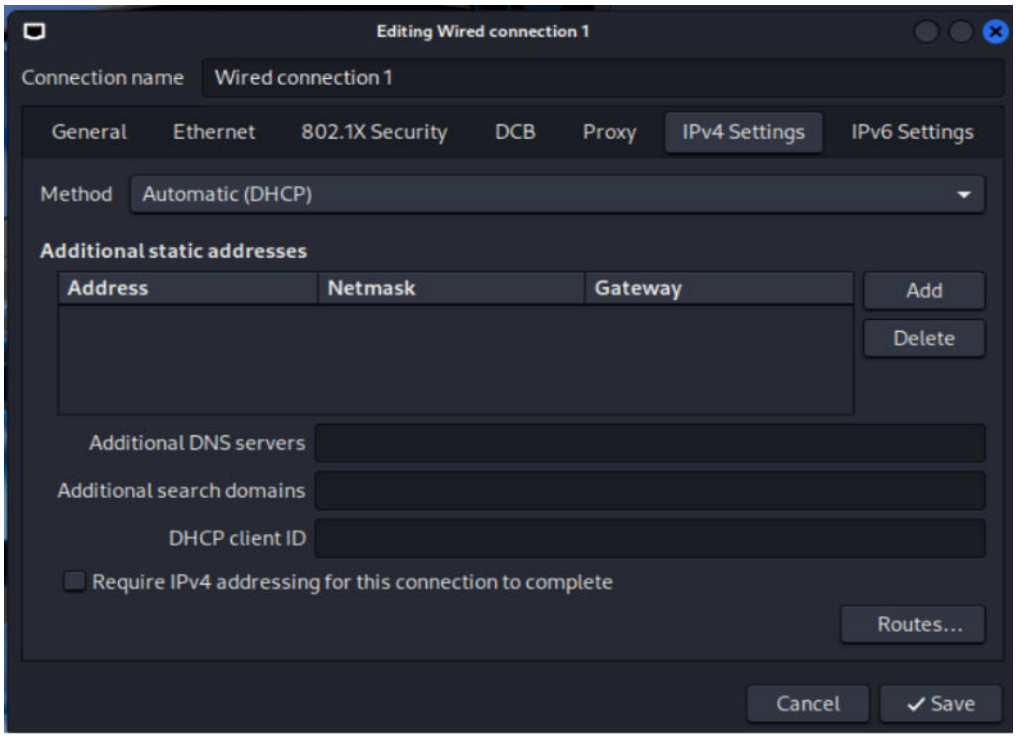
Configuración de la IP de la máquina de Kali.



Nota. Para comenzar la configuración de la IP de la máquina de Kali, es necesario ir a la configuración de red (ícono de red) y seleccionar la opción Wired Connection.

Figura 35

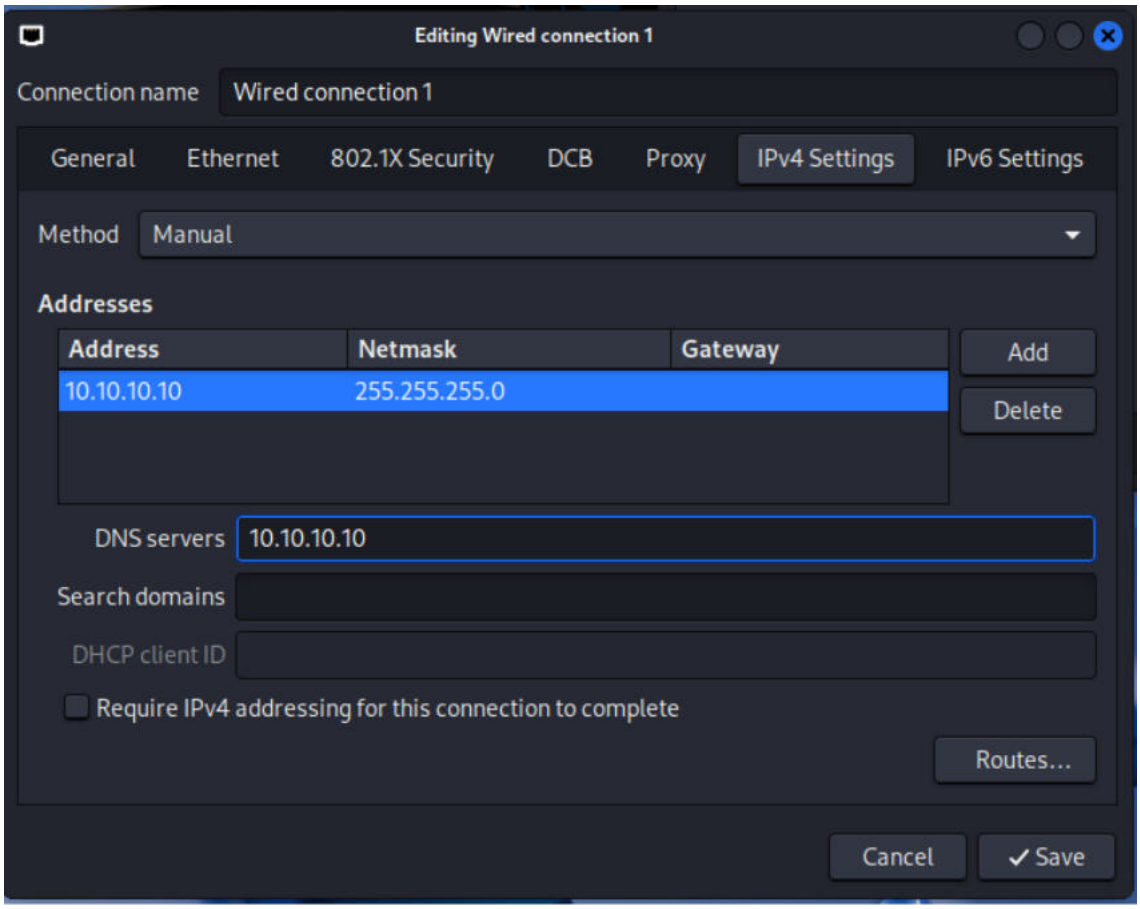
Nos dirigimos a la parte de Settings y a la configuración de IPv4.



Nota. Elegimos los parámetros requeridos para esta máquina virtual.

Figura 36

Nos dirigimos a la parte de Settings y a la configuración de IPv4.

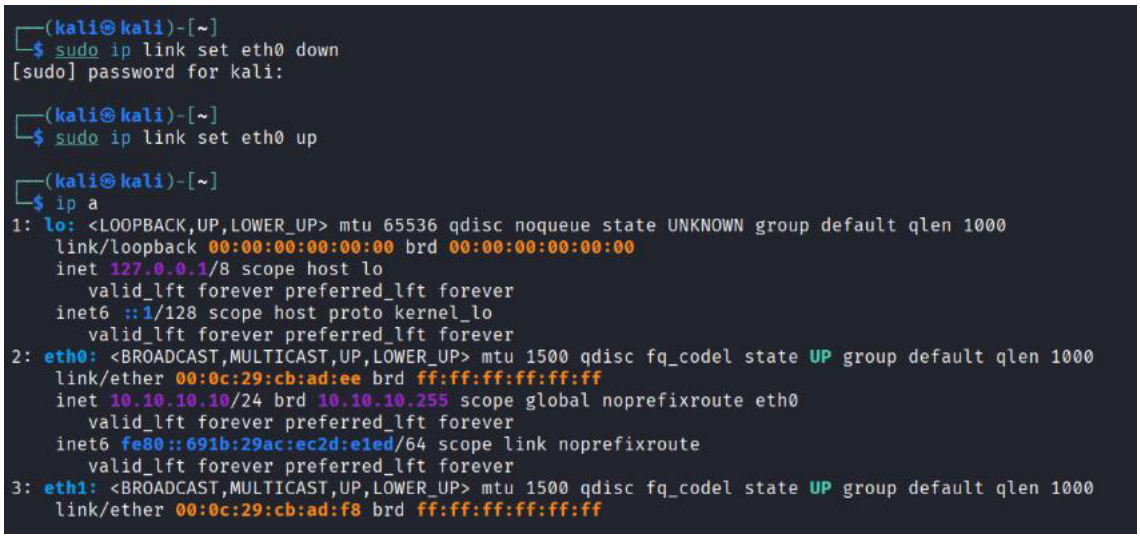


Nota. Elegimos los parámetros requeridos para esta máquina virtual.

Una vez realizado estas modificaciones, debemos abrir un terminal en el Kali para verificar que las configuraciones se encuentren activas, esto lo realizamos con el comando “ip a”.

Figura 37

Abrimos un terminal en el Kali para verificar que las configuraciones se encuentren activas

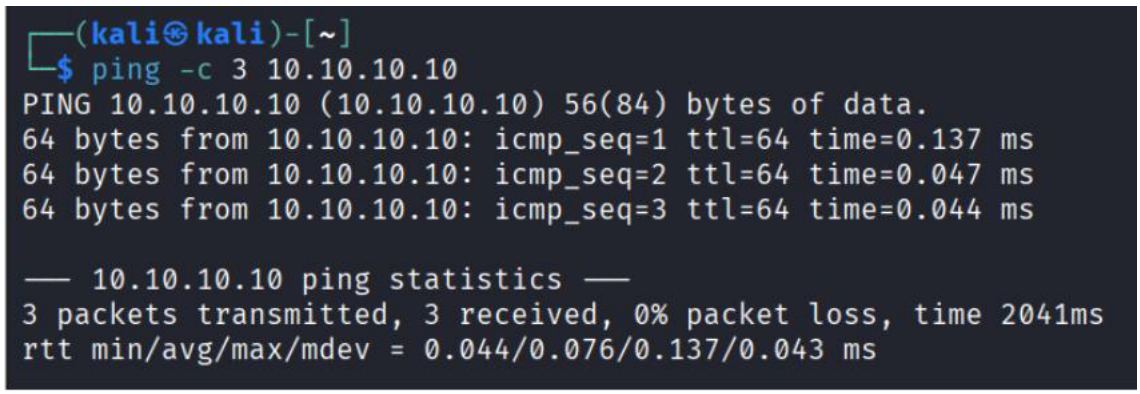


Nota. Abrimos un terminal en el Kali para verificar que las configuraciones se encuentren activas.

Del mismo modo, realizaremos un ping a la misma máquina para verificar que todo esté funcionando de manera correcta. Se puede observar que todo está correcto en la máquina Kali.

Figura 38

Abrimos un terminal en el Kali para realizar ping a los diferentes dispositivos.



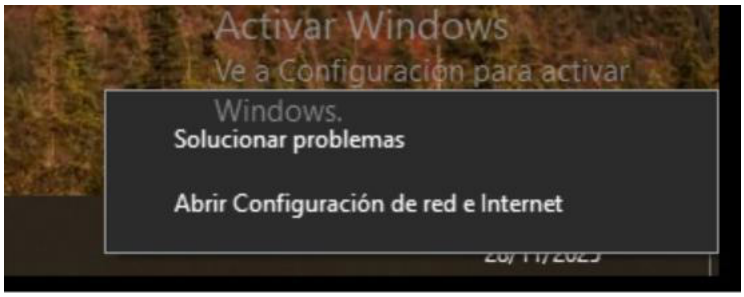
Nota. Abrimos un terminal en el Kali para realizar ping a los diferentes dispositivos

Ahora debemos configurar la IP de la máquina de Windows 10.

Para empezar, se debe ir al ícono de red y seleccionar Abrir configuraciones de red e internet.

Figura 39

Configuración de red Windows 10

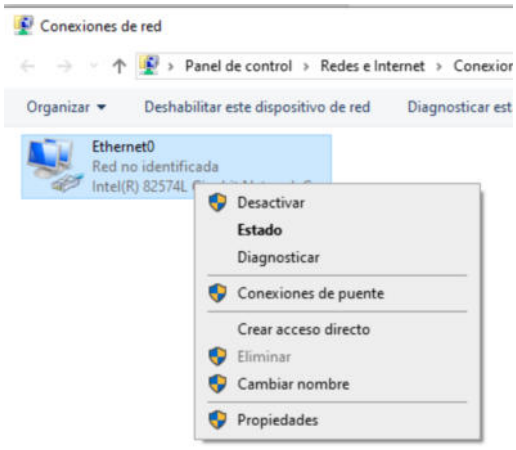


Nota. Configuración de red Windows 10

En las conexiones de red, se debe de identificar el adaptador del Vmware, que normalmente es Ethernet. Como siguiente paso abrimos las propiedades.

Figura 40

Configuración de red Windows 10

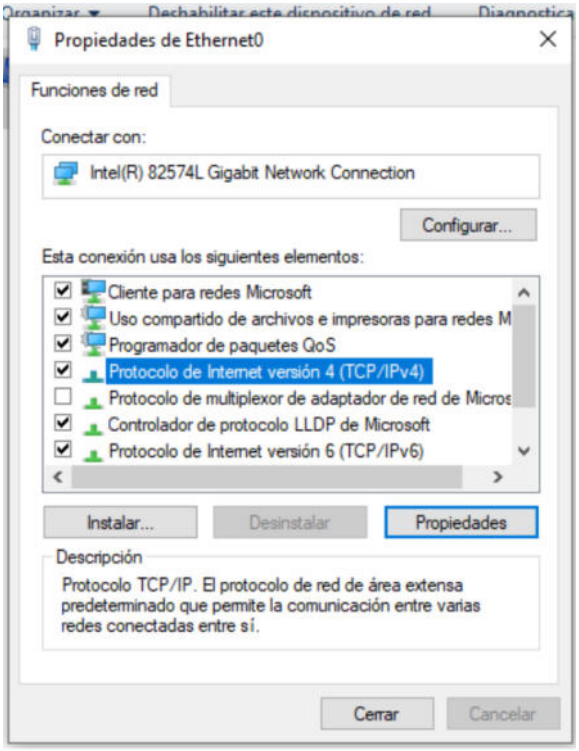


Nota. Configuración de red Windows 10

Elegimos la opción Protocolo de Internet versión 4 (TCP/IPv4) y damos clic en Propiedades

Figura 41

Configuración de red Windows 10

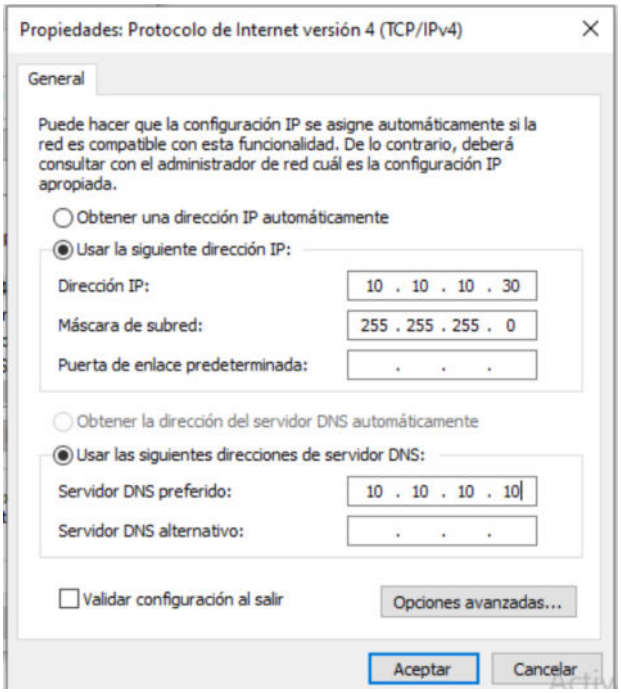


Nota. Configuración de red Windows 10

Para culminar elegimos las configuraciones previamente comentadas:

Figura 42

Configuración de red Windows 10

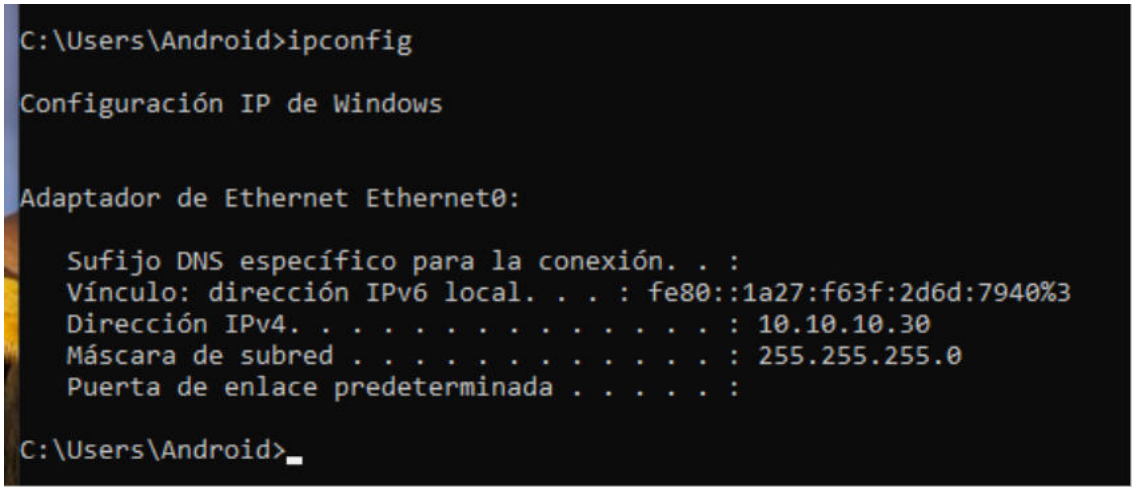


Nota. Configuración de red Windows 10

Para verificar que las configuraciones se encuentren de manera correcta, probamos el comando ipconfig en el cmd, para que nos muestre las configuraciones de red.

Figura 43

Abrimos un terminal para hacer un ipconfig

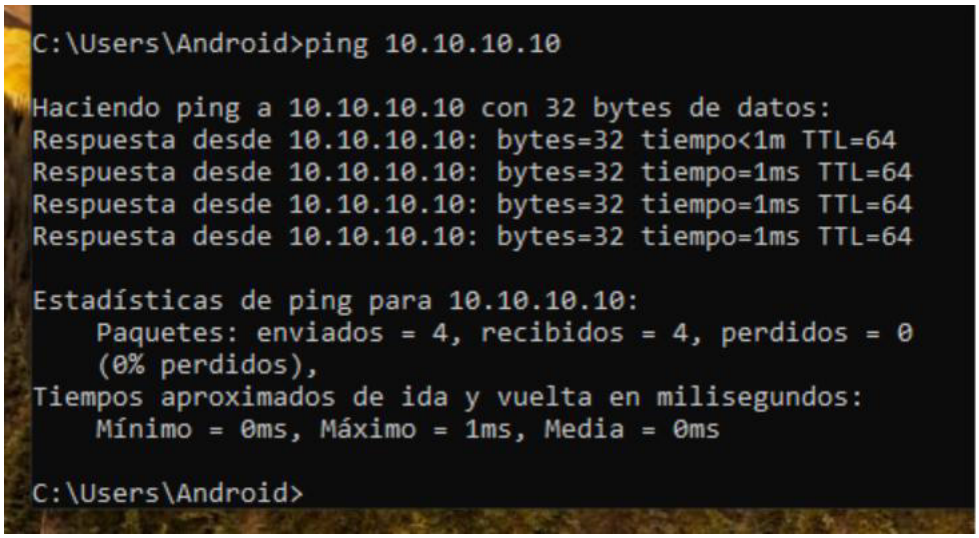


Nota. Abrimos un terminal para hacer un ipconfig

Por último, se realiza un ping desde la máquina Windows 10 hacia la máquina de Kali, y ya que el ping funcionó, eso significa que ambas máquinas ya tienen conexión.

Figura 44

Abrimos un terminal para hacer ping

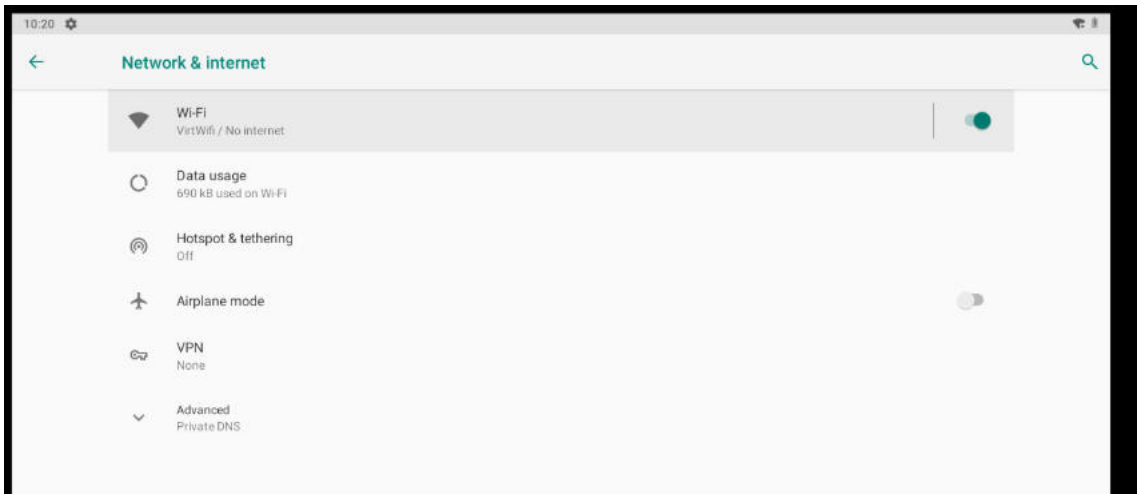


Nota. Abrimos un terminal para hacer ping

Para finalizar con las configuraciones de las IPs, se debe realizarla en la máquina Android x86, para lo cual ingresamos a la máquina virtual y nos dirigimos a las configuraciones y seleccionamos la opción de Red e Internet.

Figura 45

Configuración de la IP en Android

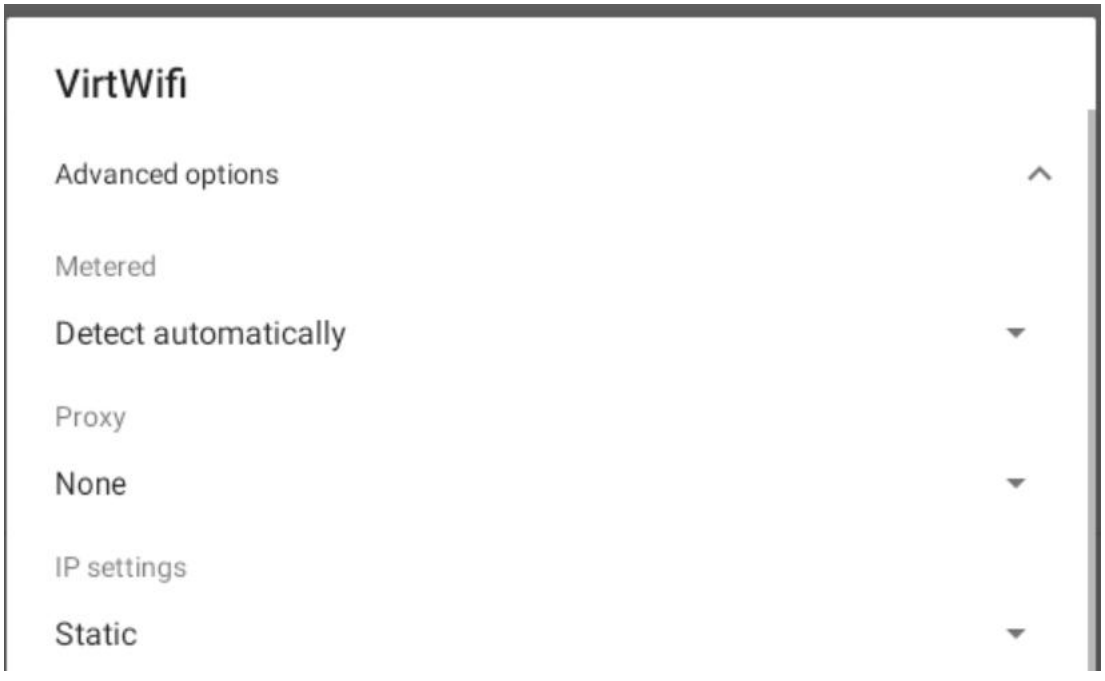


Nota. Configuración de la IP en Android

Una vez ingresado a esta configuración, nos dirigimos a la configuración IP y cambiamos esta opción de DHCP a Estática.

Figura 46

Configuración de la IP en Android

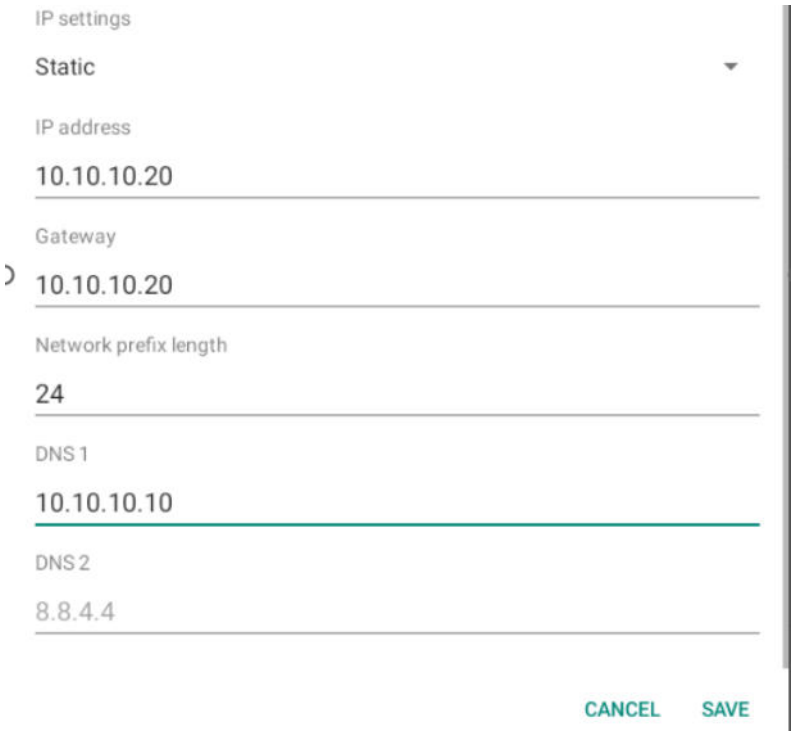


Nota. Configuración de la IP en Android

Para finalizar, llenamos la información con la configuración que requerimos para esta máquina virtual.

Figura 47

Configuración de la IP en Android

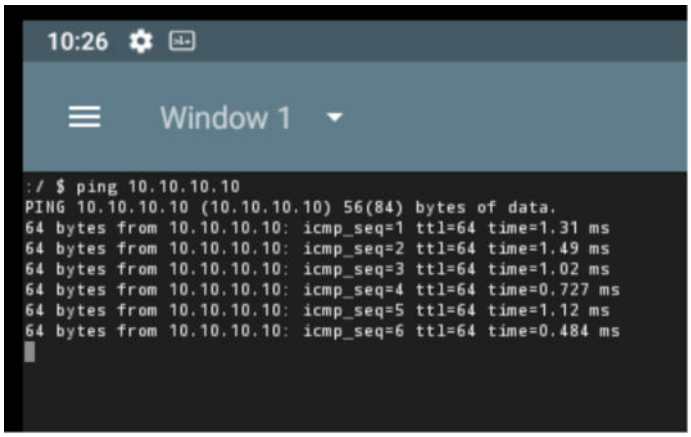


Nota. Configuración de la IP en Android

Para verificar que la configuración se realizó de manera correcta, realizamos un ping a la IP de la máquina virtual.

Figura 48

Ping en Android

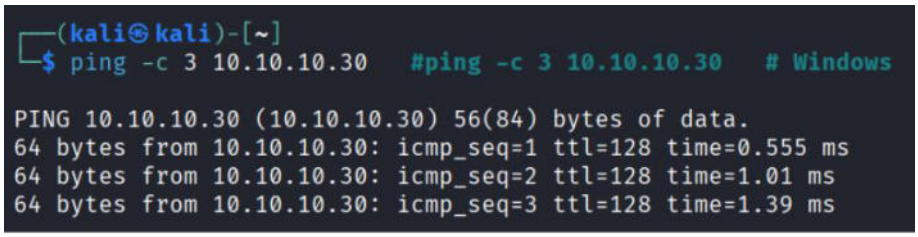


Nota. Ping en Android

Adicional a estas verificaciones, se procede a realizar la verificación de la red entre las 3 máquinas haciendo un ping desde Kali hacia las 2 otras máquinas, para verificar que todo se encuentre de manera correcta.

Figura 49

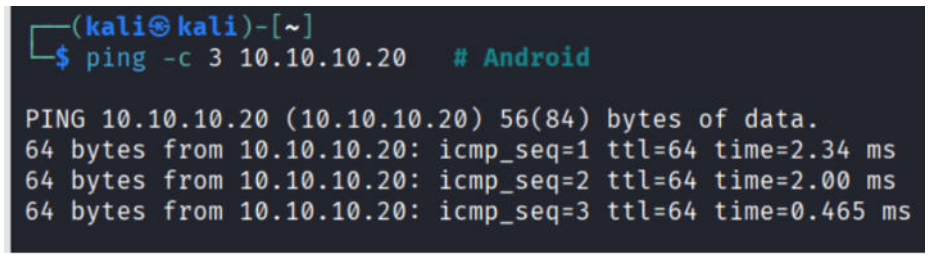
Ping Ping a windows desde Kali



Nota. Ping a windows desde Kali

Figura 50

Ping Ping a android desde Kali



Nota. Ping a Android desde Kali

Como se mencionó anteriormente, se utilizará la herramienta WireShark, para su configuración debemos abrir un terminal en Kali y ejecutar “wireshark &”.

Figura 51

Configuración de wireshark

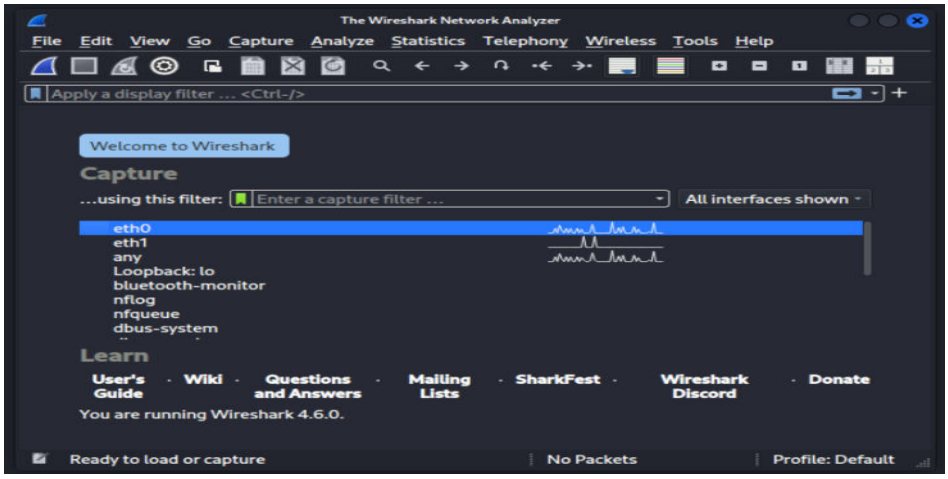
Nota. Configuración de wireshark

O también se lo puede hacer mediante el menú yendo a las opciones:

Applications → Sniffing & Spoofing → Wireshark

Figura 52

Configuración de Wireshark

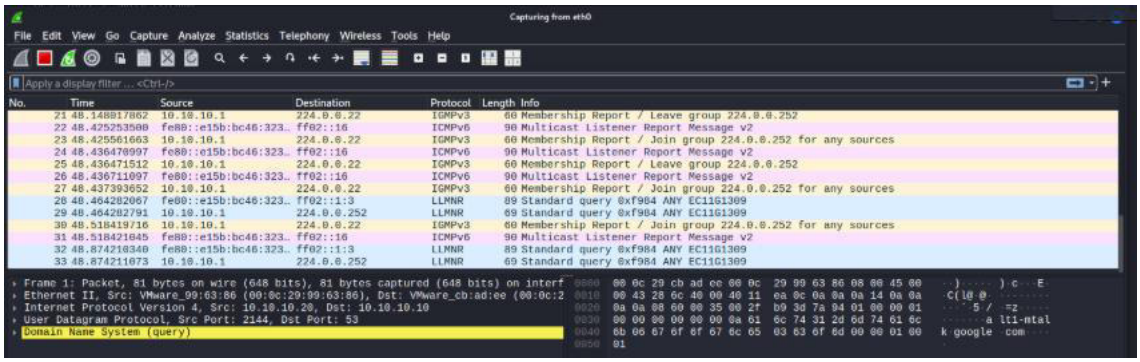


Nota. Configuración de wireshark

Para probar la captura con Wireshark es necesario dar clic en la interfaz y una vez hecho eso, pulsar el botón Start Capturing. Con esto observaremos varios paquetes.

Figura 53

En Wireshark damos en el icono azul Start Capturing

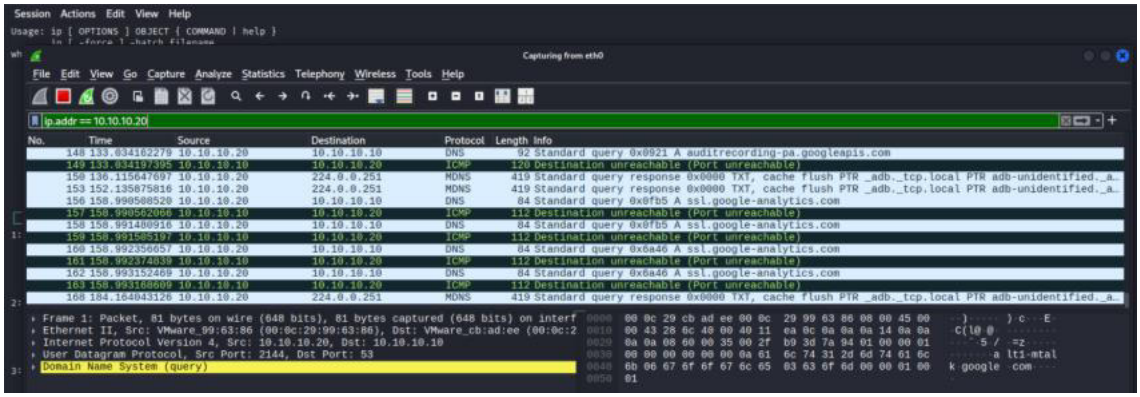


Nota. En wireshark damos en el icono azul Start Capturing

Para poder observar solo el tráfico de Android, es necesario añadir la ip de la máquina virtual (10.10.10.20) en el buscador.

Figura 54

En Wireshark vemos el tráfico de Android.

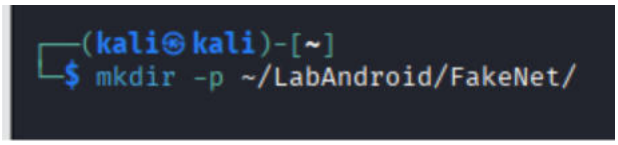


Nota. En wireshark vemos el tráfico de Android.

Volviendo a la herramienta FakeNet, debemos de configurarlo en la máquina Kali, para lo cual debemos crear un directorio en la ruta del proyecto. En este directorio vamos a guardar los logs, configuraciones, evidencias, etc.

Figura 55

Configuración de Fakenet

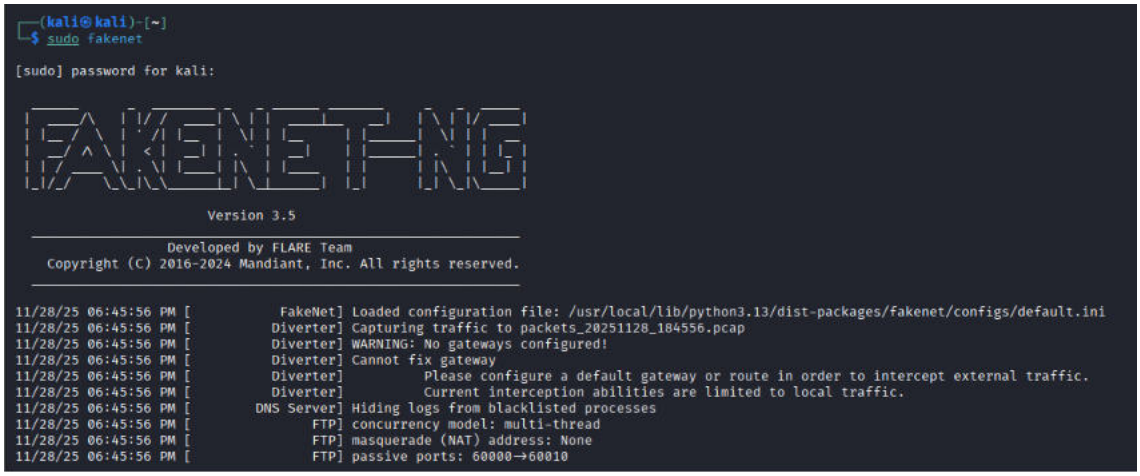


Nota. Configuración de Fakenet

Ejecutamos Fakenet en el terminal de la máquina de Kali y podemos observar que se encuentra listo para ser utilizado.

Figura 56

Ejecución de fakenet



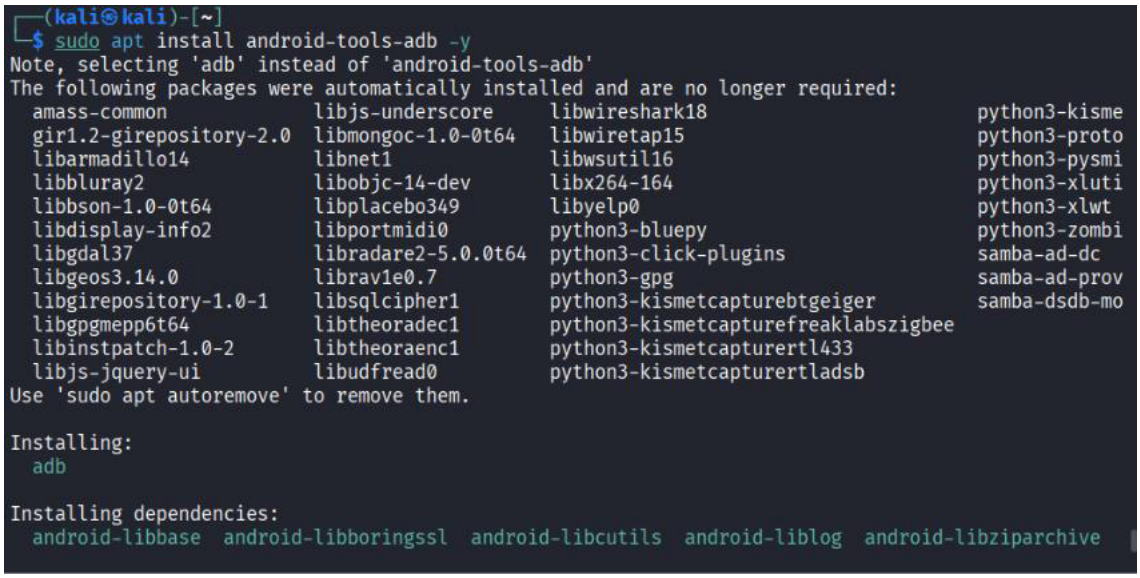
Nota. Ejecución de fakenet

Para finalizar con las herramientas a utilizar en el análisis dinámico, en la máquina de Kali, vamos a instalar abd (android debug bridge), que según lo mencionado por Kili (2023), abd es una herramienta de línea de comandos, la cual permite la comunicación entre una computadora personal y un dispositivo o instancia de emulador con Android conectado a través de un cable USB o TCP/I.

Para instalar abd en la máquina de Kali es necesario correr el siguiente comando: “sudo apt install android-tools-adb -y “

Figura 57

Instalación de Android Tools adb.



Nota. Instalación de Android Tools adb.

Una vez instalado, podemos verificar la versión que se instaló, mediante el comando “abd version” y se puede identificar que se instaló la versión 1.0.41.

Figura 58

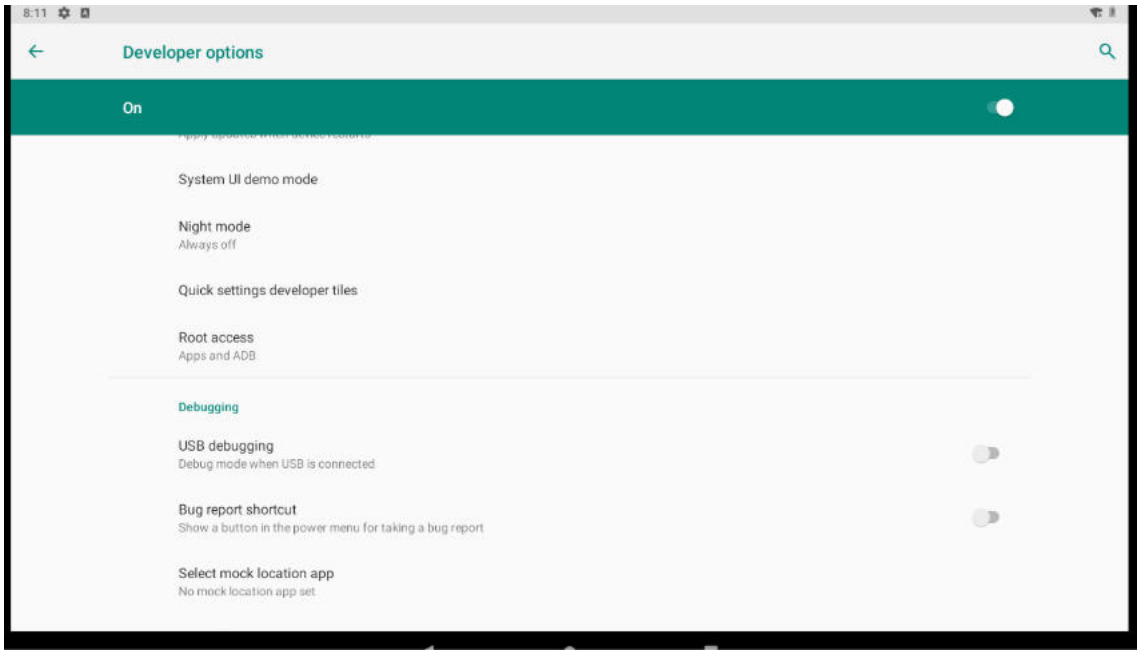
Versión de adb

Nota. Versión de adb.

En la máquina de Android, es necesario realizar algunas configuraciones, para lo cual nos dirigimos a la configuración y activamos la opción de desarrollador.

Figura 59

Configuración de depuración USB en Android

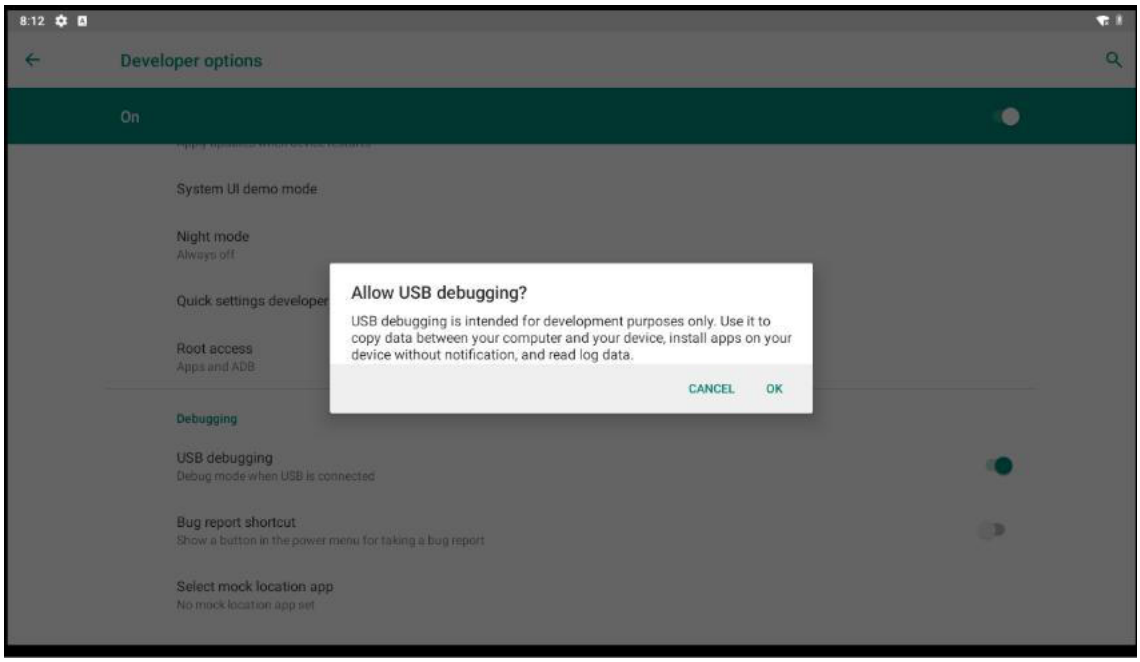


Nota. Configuración de depuración USB en Android.

Una vez realizado esto, debemos habilitar la Depuración USB.

Figura 60

Configuración de depuración USB en Android

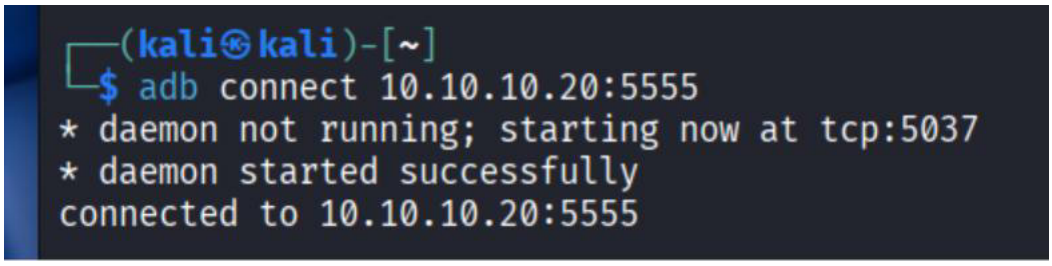


Nota. Configuración de depuración USB en Android.

Se procederá a realizar la conexión desde la máquina Kali hacia la máquina Android mediante ABD, mediante el comando “adb connect 10.10.10.20:5555 “ (ip de la máquina Android). Se comprueba que hay conexión desde Kali hasta Android.

Figura 61

Conexión adb desde Kali hacia Android



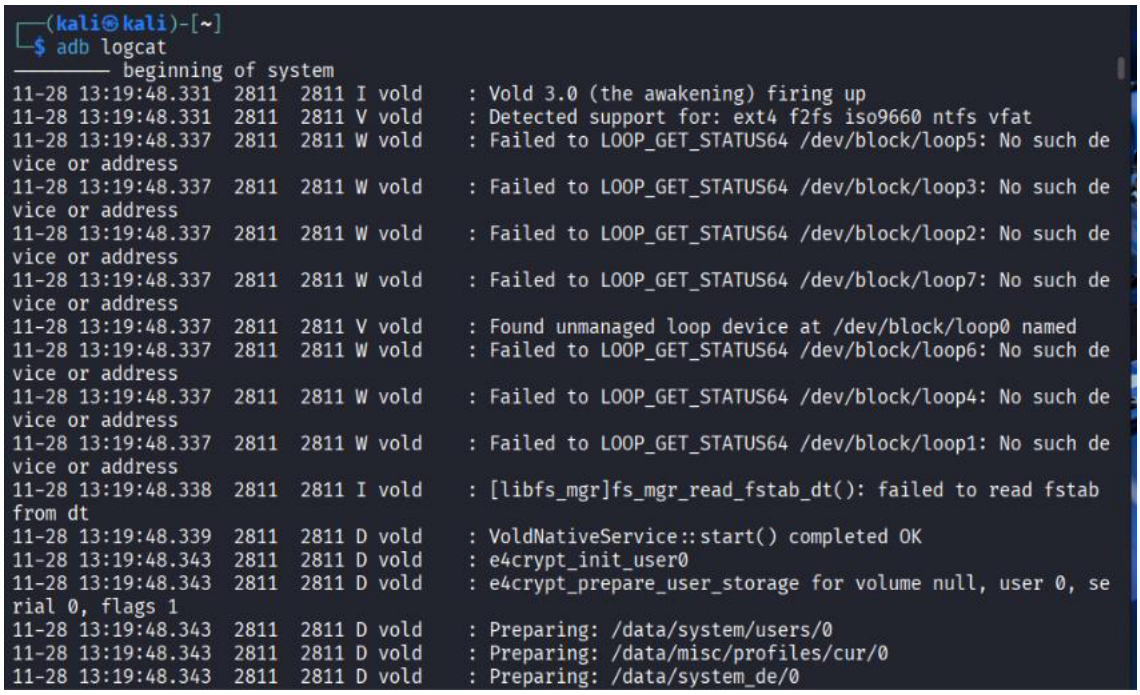
Nota. Conexión Adb desde Kali a Android.

Por último, tenemos que preparar el logcat para realizar el análisis dinámico, según Ramírez (2020), el logcat es una herramienta cuya finalidad es la de acceder al registro de

mensajes del sistema de Android, por lo cual es fundamental para el análisis. Par verificar que el logcat esté funcionando de manera correcta procedemos a usar el comando “adb logcat”.

Figura 62

Ejecución de Logcat en Kali



Nota. Ejecución de Logcat en Kali

Con esto confirmamos que el logcat está funcionando de manera correcta.

CAPITULO 4:

ANÁLISIS DE RESULTADOS

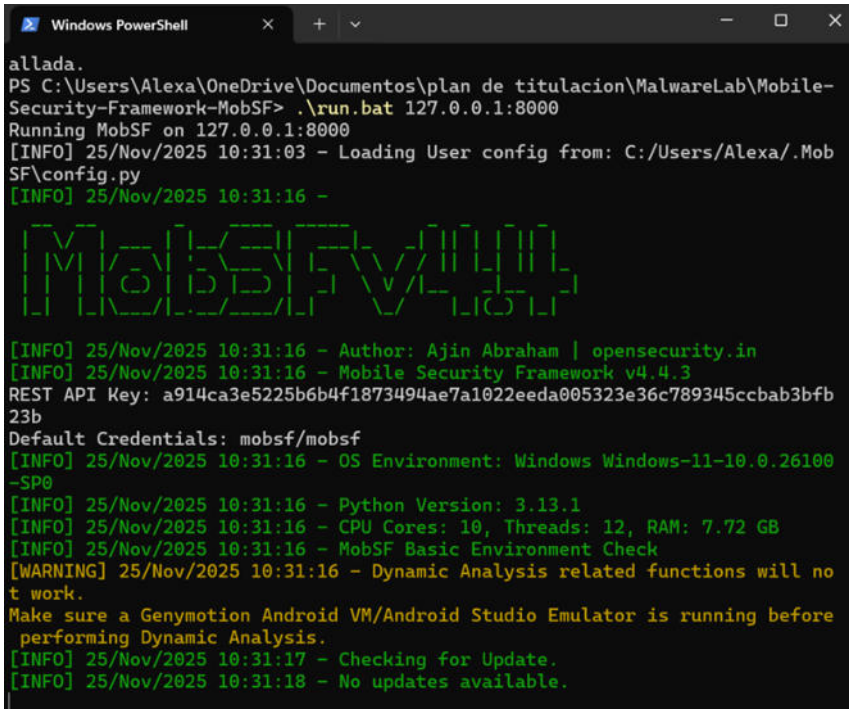
4.1. Análisis Estático

Para el análisis estático de las pruebas de concepto, se desplegó un entorno controlado de laboratorio utilizando la herramienta de Mobile Security Framework (MobSF).

La instancia se inicializo en la maquina local Windows mediante el script de ejecución en el puerto 8000, asegurándonos que el motor de análisis estático estuviera operativo para la ingesta de binarios.

Figura 63

Despliegue del entorno MobSF.



Nota. Inicialización de MobSF

El procedimiento estandarizado para cada muestra consistió en la carga del archivo APK mediante la interfaz web de la herramienta MobSF, lo que desplegó de forma automática los procesos de descompilación, análisis de manifiesto y extracción de recursos. Adicionalmente, se completó el estudio utilizando JADX-GUI para la ingeniería inversa con una revisión manual del código fuente Java y Virus Total para la validación de reputación mediante firmas.

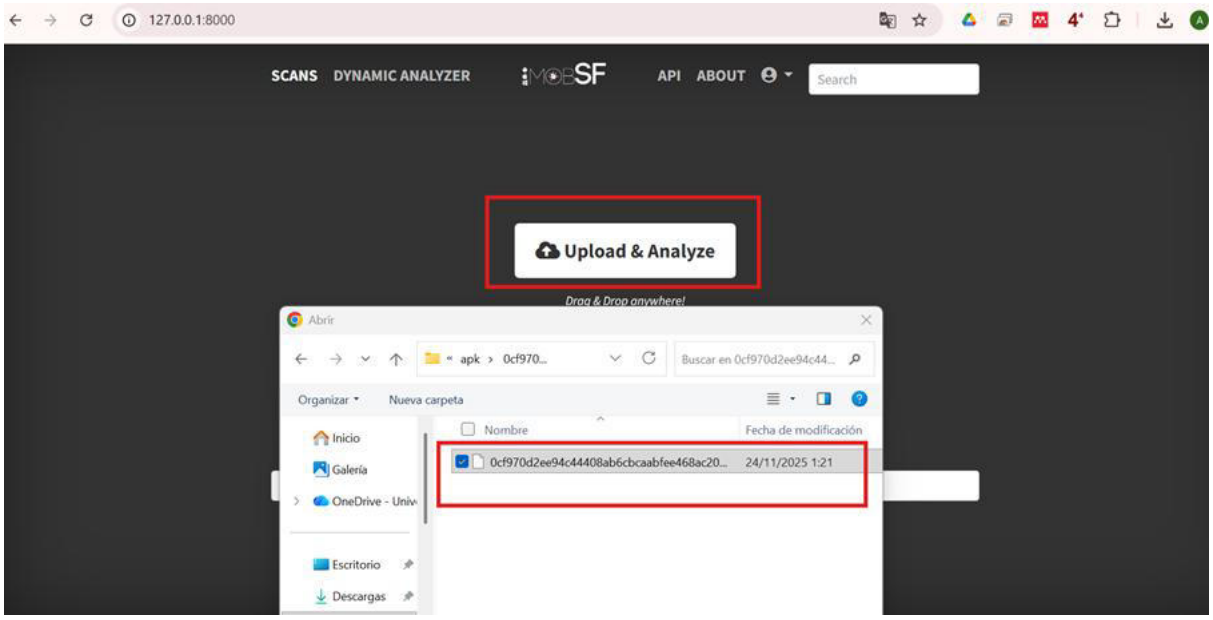
4.1.1. Análisis de Resultados - Caso de estudio 1

Aplicación fraudulenta obtenida de malwareBazaar Malware Bancario (Troyano) (0cf970d2ee94c44408ab6cbcaabfee468ac202346b9980f240c2feb9f6eb246d)

Para empezar el análisis cargamos el archivo apk muestra, seleccionamos el archivo y le damos en abrir.

Figura 64

Carga de la muestra de malware en MobSF

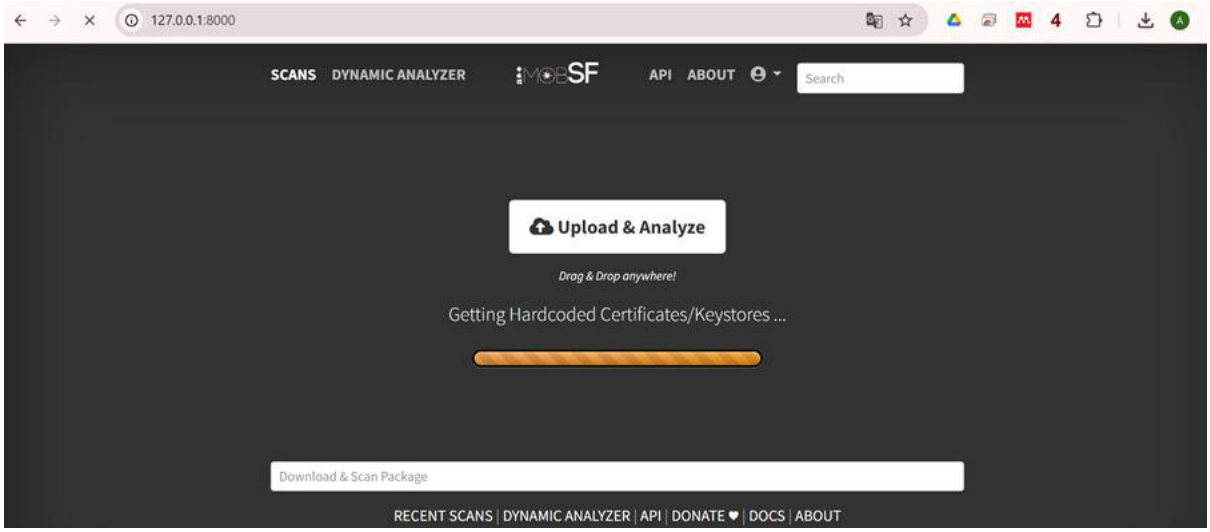


Nota. La imagen muestra el proceso de seleccionar la muestra apk para cargarlo en MobSF.

Una vez seleccionado el archivo empezara el proceso de análisis estático.

Figura 65

Carga de la muestra de malware en MobSF

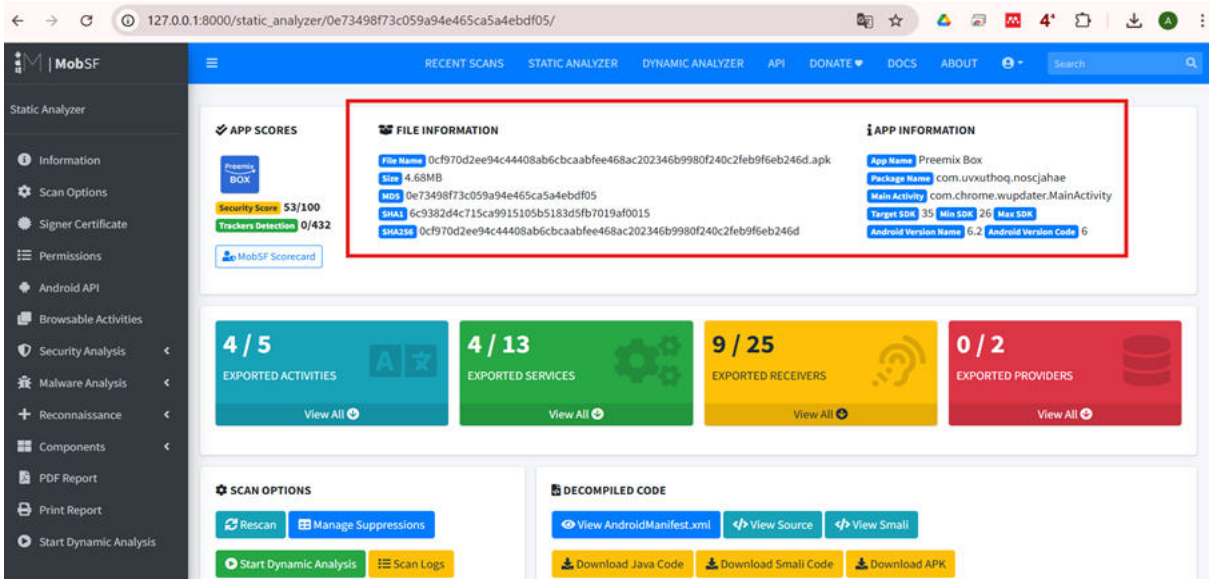


Nota. Proceso de análisis en mobSF.

Al finalizar el proceso de análisis del apk1, se visualiza información general como nombre, encriptación, tamaño, entre otra información importante.

Figura 66

Panel de resultados

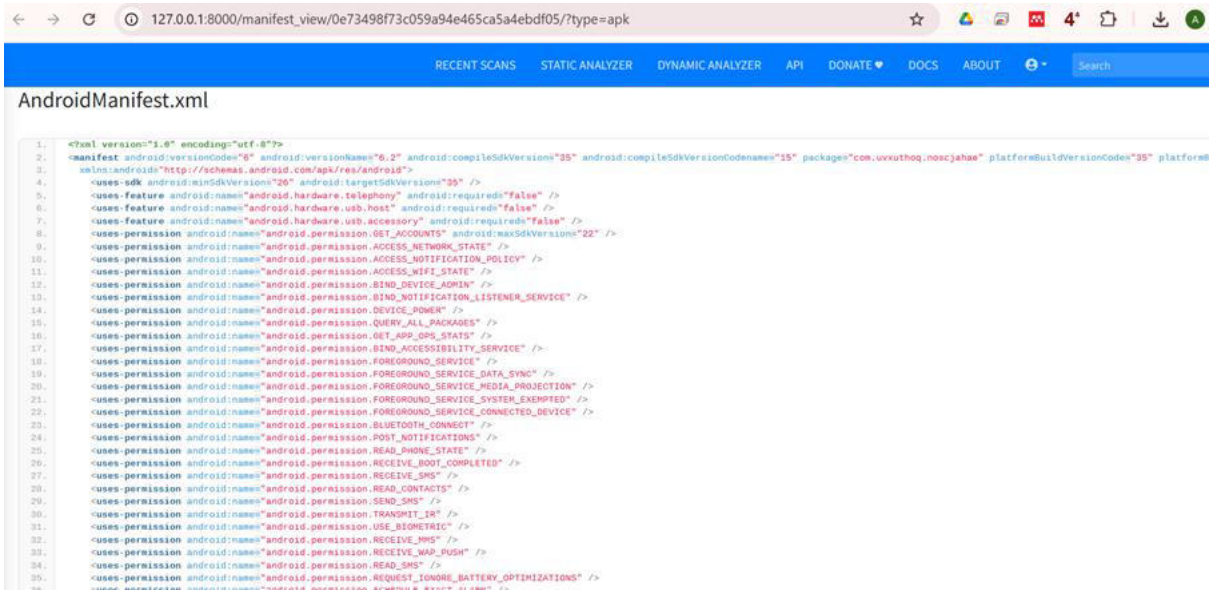


Nota. Información general tras el análisis estático.

En el archivo AndroidManifest.xml podemos ver información importante que define como la aplicación interactúa con el sistema, se trata del documento de identidad de la app.

Figura 67

Visualización del código fuente del archivo AndroidManifest.xml en MobSF



Nota. Extracto del código XML recuperado desde MobSF.

En este análisis estático del archivo AndroidManifest.xml , perteneciente al paquete com.uvxuthoq.noscjahae se evidencia que la app analizada no es una aplicación legitima, si no un Software malicioso (Malware) con características de Troyano bancario y herramienta de acceso remoto (RAT).

Esta app emplea técnicas de ingeniería social al utilizar espacios de nombres internos (com.chrome.wupdater) que imitan componentes de actualización de Google Chrome para confundir al usuario y evadir la detección superficial. El ventor principal de ataque es el abuso de los servicios de accesibilidad (Accessibility Services), lo que le otorga control total sobre la interfaz de usuario para realizar keylogging, robo de credenciales en pantalla y otorgamiento automático de permisos sin interacción del usuario. Aparte, implementa

mecanismos avanzados de persistencia (Administrador de Dispositivos) y evasión (iconos transparentes y ocultación del cajón de aplicaciones), junto con la capacidad de intervenir comunicaciones SMS para la evasión del doble factor de autenticación (2FA).

A continuación, se muestra una tabla con los hallazgos técnicos y vulnerabilidades.

Tabla 2

Matriz de hallazgos del análisis estático

Categoría	**Evidencia en AndroidManifest.xml**	**Análisis de impacto y riesgo**
Suplantación de Identidad	package="com.uvxuthoq.noscjahae" vs componentes nombrados como com.chrome.wupdater...	RIESGO ALTO. La aplicación intenta hacerse pasar por un servicio legítimo de "Google Chrome Update" para evitar sospechas por parte del usuario y dificultar el análisis forense rápido.
Control Total del Dispositivo	<service ... permission="android.permission.BIND_ACCESSIBILITY_SERVICE">	CRÍTICO. El abuso de accesibilidad permite al malware leer todo lo que aparece en pantalla (robo de contraseñas bancarias), realizar clics automáticos y superponer ventanas (Overlay Attacks). Es el vector principal de los troyanos modernos.
Robo de 2FA (SMS)	READ_SMS, RECEIVE_SMS, SEND_SMS con android:priority="999"	CRÍTICO. La prioridad máxima (999) en el receptor de SMS le permite al malware interceptar y ocultar mensajes entrantes (como códigos OTP bancarios) antes de que el usuario los vea.

Persistencia y Bloqueo	<receiver ... permission="android.permission.BIND_DEVICE_ADMIN">	ALTO. Al solicitar permisos de Administrador de Dispositivos, el malware dificulta extremadamente su desinstalación, pudiendo bloquear la pantalla o cambiar el PIN si el usuario intenta borrarla.
Técnicas de Evasión	<activity-alias ... android:label=" " ... icon="@drawable/transparent_icon">	MEDIO/ALTO. Uso de caracteres invisibles en la etiqueta y un icono transparente para que, una vez instalada, la app "desaparezca" de la vista del usuario, aunque siga ejecutándose en segundo plano.
Ocultación de Actividad	android:excludeFromRecents="true", android:noHistory="true"	MEDIO. Evita que la aplicación aparezca en la lista de "Aplicaciones Recientes" o en el historial de tareas, ocultando su ejecución activa al usuario.
Ejecución Automática	RECEIVE_BOOT_COMPLETED, WAKE_LOCK, REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	ALTO. Garantiza que el malware se inicie automáticamente al encender el teléfono y previene que el sistema operativo lo cierre para ahorrar batería, asegurando una conexión constante con el servidor C&C.
Sofisticación del Desarrollo	targetSdkVersion="35" (Android 15)	INFORMATIVO. Indica que los desarrolladores del malware mantienen el

	código actualizado para intentar evadir las restricciones de seguridad más recientes de las últimas versiones de Android.

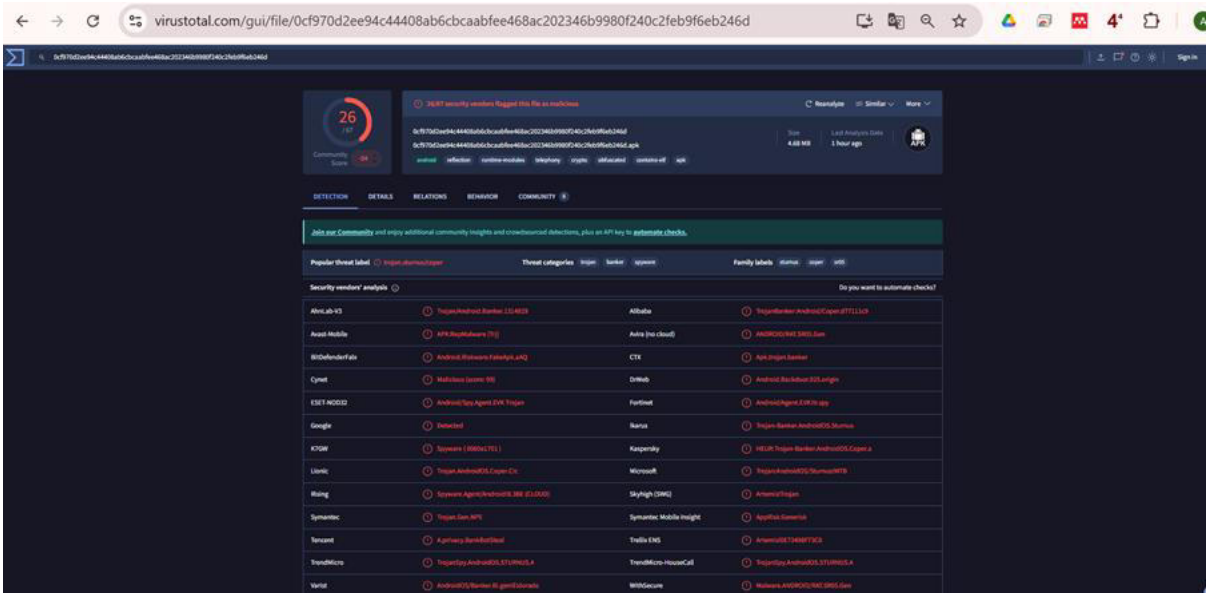
Nota. Se detalla los componentes y permisos críticos hallados en el manifiesto de la aplicación maliciosa.

4.1.2. Análisis de reputación (hashes) - Caso de estudio 1

Se busco con el SHA256: (0cf970d2ee94c44408ab6cbcaabfee468ac202346b9980f240c2feb9f6eb246d) en virus total en el cual se identificó positivamente la amenaza como Android/Trojan.Banker.Coper (Starnus), un troyano bancario avanzado, esta clasificación valida el análisis estático, confirmando que los permisos abusivos se utilizan para el robo de crendenciales y keylogging.

Figura 68

Análisis de reputación de la muestra de malware mediante VirusTotal



Nota. Clasificación en virus total.

4.1.3. Análisis de código fuente (ingeniería inversa) - Caso de estudio 1.

En el código fuente de java, en la carpeta llamada a y en el archivo “ t21.java” podemos ver la parte de Ofuscación de Cadenas y Configuración Dinámica: "El análisis del código fuente (clase t21.java) revela que las direcciones del servidor de Comando y Control (C2) no están almacenadas en texto plano. El malware implementa un mecanismo de protección que recupera cadenas cifradas desde la clase BuildConfig (variables v0 y v1) y las descifra dinámicamente en tiempo de ejecución utilizando la clase y21. Esta técnica tiene como objetivo evadir la detección basada en firmas de red estáticas."

Figura 69

Análisis de la clase t21

Nota. Fragmento del código Java obtenido mediante el descompilador de MobSF.

4.1.4. Certificado de firma APK 1- Caso de estudio 1.

En este análisis podemos evidenciar el intento deliberado de suplantación de identidad corporativa.

Subject: CN=Google Chrome, 0=Google Inc, L=Mountain View.

Issuer: CN=Google Chrome, 0=Google Inc.

Certificado Autofirmado (Self-Signed): El certificado carece de una cadena de confianza válida. El emisor es el mismo que el sujeto.

Suplantación de Marca: Los campos de "Organización" y "Nombre Común" han sido manipulados con datos falsos de "Google Inc".

Cronología del Ataque: La fecha de validez (Valid From: 2025-10-14) establece que esta campaña específica o "build" del malware fue generada a mediados de octubre de 2025, indicando una amenaza activa y reciente.

Figura 70

Detalle del certificado de firma digital

Nota. Sigener certificate, muestra que el malware utiliza un certificado falsificado.

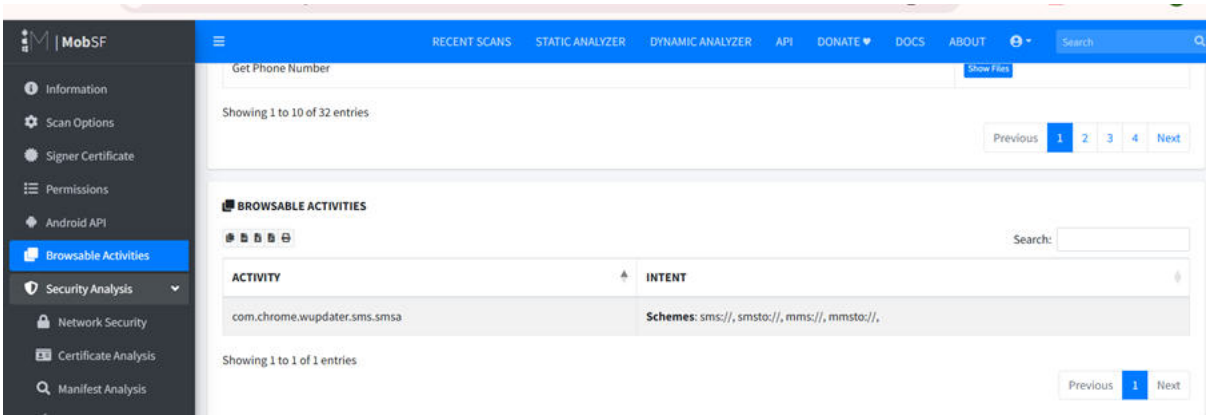
4.1.5. BROWSABLE ACTIVITIES - Caso de estudio 1

Se evidencio que la actividad com.chrome.wupdater.sms.smsa está configurada como "Navegable" (Browsable) y registra filtros de intención (Intent Filters) para los protocolos de mensajería estándar: sms://, smsto://, mms:// y mmsto://.

Con esto podemos reforzar la capacidad del troyano (identificado como Coper) para controlar el flujo de entrada y salida de mensajería SMS, vital para la intercepción de tokens bancarios (OTP) y la propagación del malware vía SMS a la lista de contactos (Smishing).

Figura 71

Actividades exportadas y configuracion de intercepcion de mansajeria SMS



Nota. Browsable activities, se visualiza los detalles de las actividades navegables detectadas por MobSF

4.1.6. ANALISIS CON JADX - Caso de estudio 1

Con open file cargamos el archivo apk y esperamos a que descompile.

Figura 72

Exploración de la estructura de directorios y recursos de la muestra de malware en Jadx

Nota. Pantalla de inicio de JADX.

A diferencia de aplicaciones legítimas que utilizan "Certificate Pinning" o almacenamiento seguro en Keystore, el malware almacena las claves en el código fuente para facilitar su descifrado rápido en cualquier dispositivo infectado, ocultando la infraestructura de red a los sistemas de detección automatizada (IDS/IPS).

De acuerdo al análisis de MobSF Se analizo la clase y21, en esta nos ha permitido determinar el algoritmo criptográfico utilizado para proteger las comunicaciones del malware.

Algoritmo: AES (Advanced Encryption Standard).

Modo de Operación: ECB (Electronic Codebook).

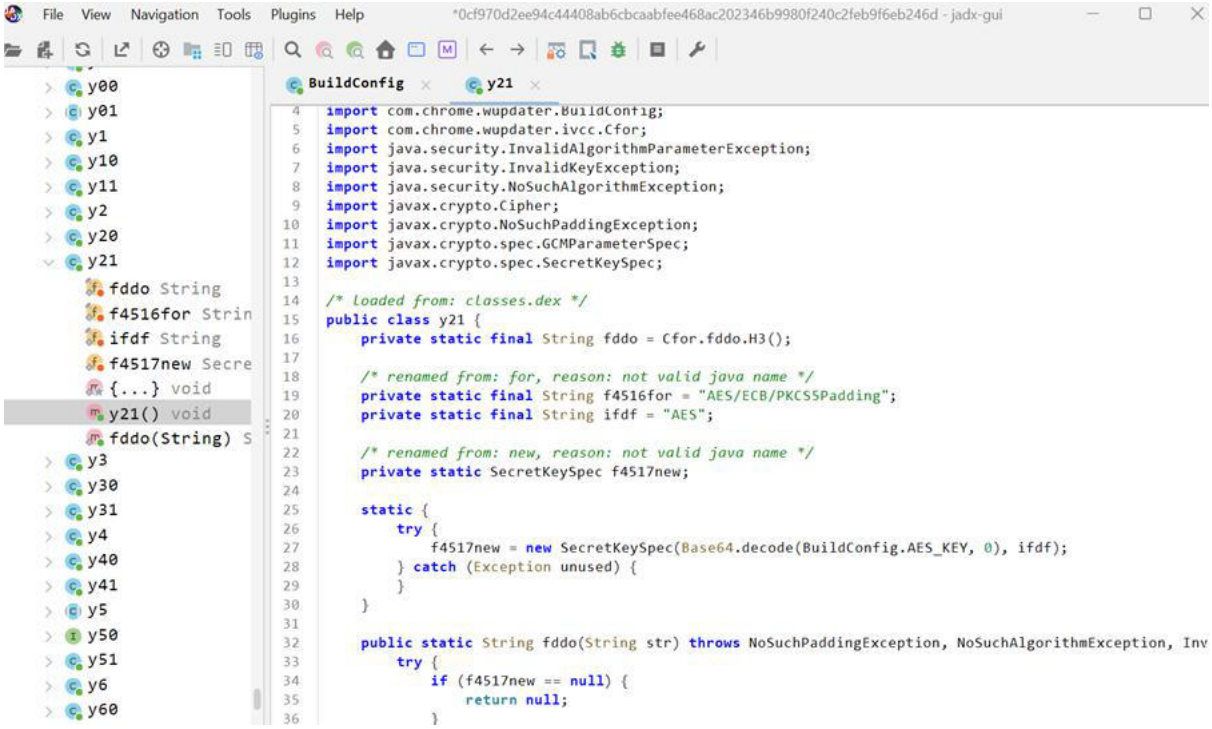
Padding: PKCS5Padding.

Gestión de Claves: La clave de sesión se recupera dinámicamente decodificando en Base64 la variable estática BuildConfig.AES_KEY.

El uso del modo ECB (inseguro por diseño debido a la falta de difusión y vector de inicialización) permitió la recuperación del texto plano (URL del C2) mediante un ataque de texto cifrado conocido, utilizando las claves estáticas extraídas del propio binario.

Figura 73

Identificación de rutinas criptográficas y recuperación de claves de cifrado AES en el código fuente



Nota. Contenido de la clase y21

Nota. Contenido de la clase y21

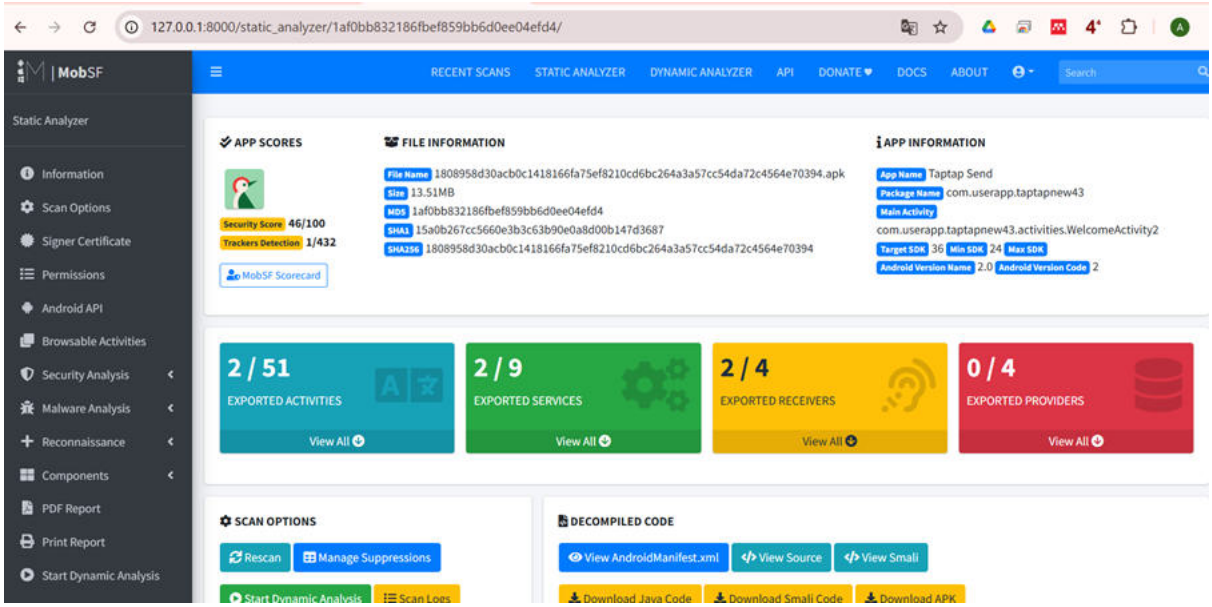
La correlación de estos hallazgos con Virus Total se identificó la amenaza como Android/Trojan.Banker.Coper (Starnus), un se trata de un troyano bancario avanzado caracterizado por su persistencia y capacidad de control remoto (RAT).

4.1.7. Análisis de Resultados - Caso de estudio 2

Aplicación fraudulenta obtenida de malwareBazaar (Fake App/Phishing) (1808958d30acb0c1418166fa75ef8210cd6bc264a3a57cc54da72c4564e70394)

Figura 74

Información general del apk 2

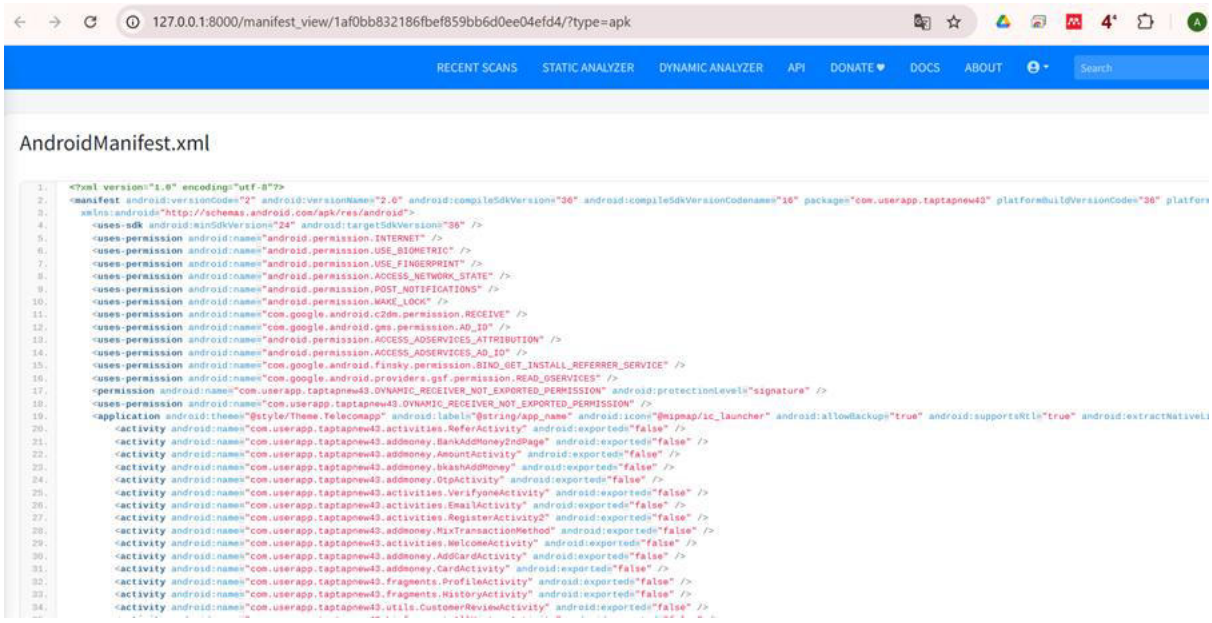


Nota. Información general del apk 2

De igual forma se empieza analizando el archivo AndroidManifest.xml.

Figura 75

Visualización del código fuente del archivo AndroidManifest.xml en MobSF



Nota. Extracto del código XML recuperado desde MobSF.

En contraste al primer caso, en este análisis podemos evidenciar que presenta un perfil totalmente diferente. No hay indicadores de un troyano bancario (RAT).

Podemos observar nombres como blashAddMoney, BkashNagadActivity, BankAddMoney, AddCardActivity y esto sugiere que se trata de una aplicación de transacciones financieras (bkash y Nagad son servicios financieros móviles muy populares en Bangladesh).

El nombre del paquete com.userapp.taptapnew43 es muy genérico y poco profesional. Las apps bancarias legítimas suelen tener dominios inversos corporativos (ej. com.bkash.wallet). Esto sugiere que podría ser una app "Fake" o de Phishing creada con un constructor de apps (App Builder) para robar dinero mediante engaño, no mediante hacking técnico.

Tabla 3

Matriz de hallazgos del análisis estático

Categoría	Evidencia en el Manifiesto	Análisis de Impacto y Riesgo
Clasificación Preliminar	package="com.userapp.taptapnew43"	**RIESGO DE FRAUDE (SCAM).** El nombre del paquete es genérico y no corporativo, lo que sugiere una app no oficial, clonada o creada con plantillas (App Builders). Posible *Phishing*.
Permisos	Ausencia de READ_SMS, ACCESSIBILITY, Overlay.	**BAJO RIESGO TÉCNICO.** A nivel de sistema operativo, la app no tiene capacidades para robar control del dispositivo ni espiar otras apps. Sus permisos son coherentes con su función.
Seguridad de Datos	android:allowBackup="true"	**VULNERABILIDAD MEDIA.** Permite realizar copias de seguridad de los datos privados de la app vía USB (ADB). En una app financiera, esto es una mala práctica de seguridad (debería ser false).

Superficie de Ataque	activities críticas con exported="false"	**BUENA PRÁCTICA.** Las pantallas de ingreso de dinero, OTP y registro están protegidas y no pueden ser invocadas por terceros.
Integraciones	firebase, google.gms.ads	**ESTÁNDAR.** Utiliza librerías legítimas de Google para notificaciones push, analíticas y publicidad. No se ven librerías ofuscadas o extrañas en el manifiesto.
Funcionalidad	Referencias a bkash, Nagad, BankAddMoney	**CONTEXTO.** La app simula ser una billetera móvil. Si no es la app oficial de bKash/Nagad, es una estafa diseñada para que el usuario "recargue saldo" voluntariamente a la cuenta del atacante.

Nota. Se detalla los componentes y permisos críticos hallados en el manifiesto de la aplicación maliciosa

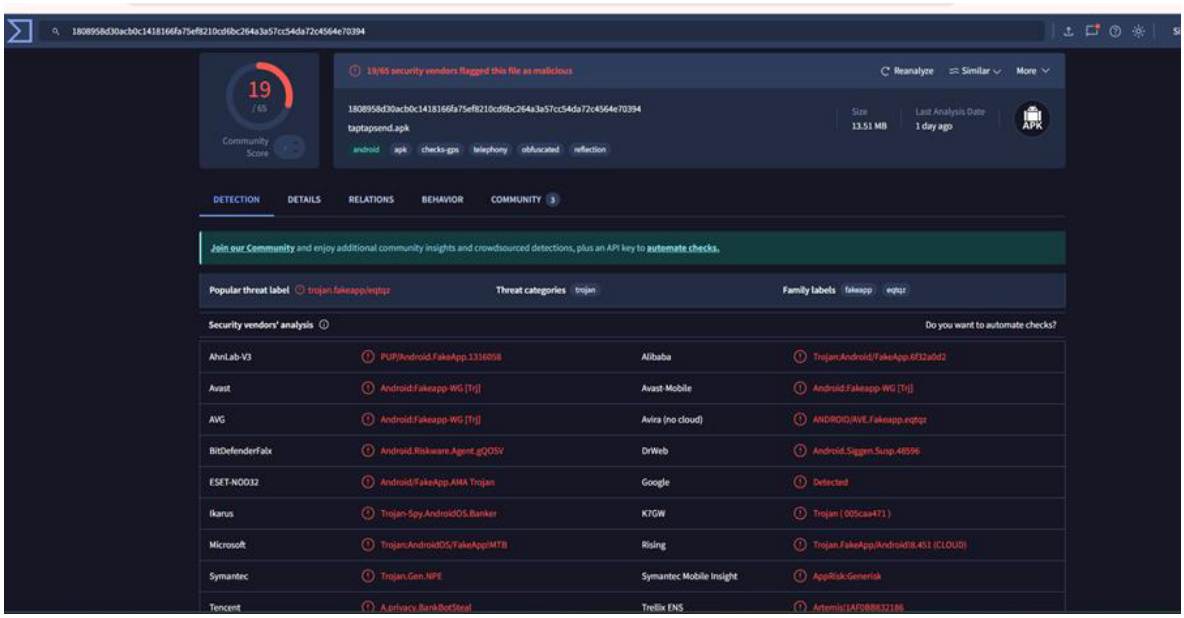
4.1.8. Análisis de reputación (hashes) - Caso de estudio 2

Podemos visualizar que arroja una tasa de detección de 19/65, con un consenso en clasificar la muestra como Trojan.FakeApp (Aplicación Falsificada).

Podemos concluir que es un software fraudulento diseñado para el robo de identidad y credenciales financieras, confirmando la naturaleza sospechosa del nombre del paquete observada previamente.

Figura 76

Análisis de reputación de la muestra de malware mediante VirusTotal



Nota. Clasificación en virus total.

4.1.9. Análisis de código fuente (ingeniería inversa) - Caso de estudio 2.

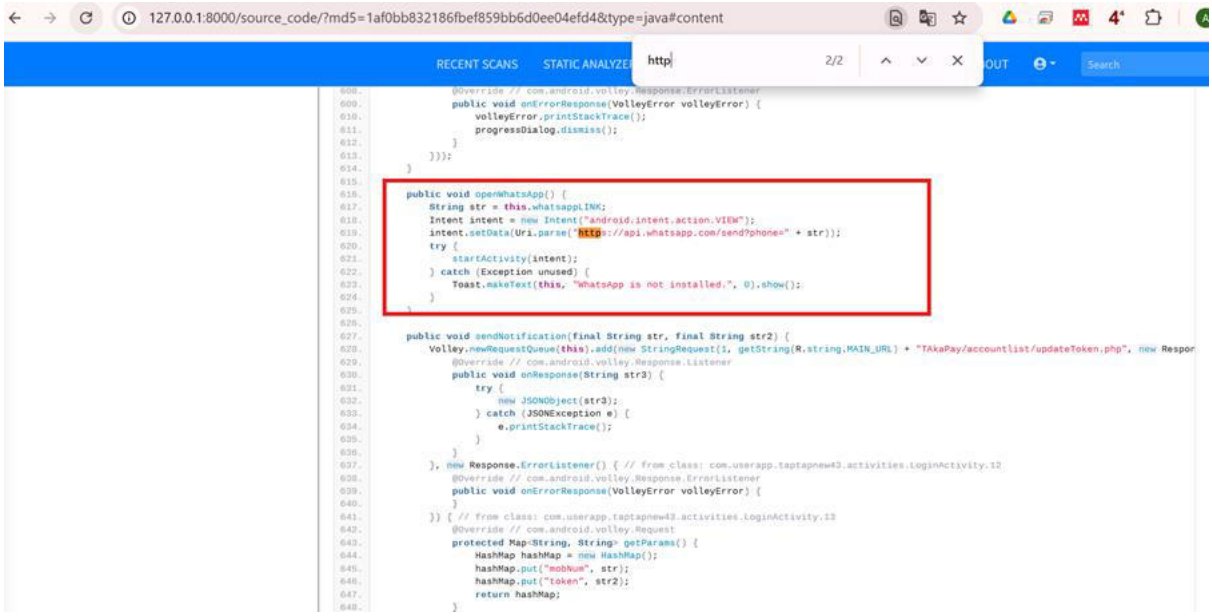
Aquí podemos evidenciar la línea getString(R.string.MAIN_URL) + "TAkaPay/accountlist/updateToken.php" , indica que la aplicación está enviando datos a una carpeta llamada TAkaPay. Esto es un indicador de que están usando un Kit de Phishing conocido (un paquete de código que venden a estafadores).

Adicional los datos van a updateToken.php. Un servidor legítimo usaría APIs modernas (REST/JSON), no scripts PHP sueltos en carpetas públicas.

hashMap.put("mobNum", str); hashMap.put("token", str2); Aquí es donde podemos evidenciar que la app roba la información. Está empaquetando el número de móvil (mobNum) y el token/contraseña (token) para enviarlos al servidor del atacante.

Figura 77

Análisis de la clase t21



Nota. Fragmento del código Java obtenido mediante el descompilador de MobSF.

4.1.10. Certificado de firma APK 2 - Caso de estudio 2

En este análisis se evidenció que el certificado digital utilizado para firmar el archivo APK (taptapsend.apk) confirma la ilegitimidad de la aplicación y su falta de vinculación con la entidad financiera oficial que pretende suplantar.

Detalles:

Sujeto y Emisor: CN=telecom.

Número de Serie: 0x1.

Fecha de Creación: 24 de octubre de 2025.

El certificado carece de los campos estándar de Organización (O), Unidad Organizativa (OU) y Ubicación (L/C) que son obligatorios en aplicaciones bancarias reguladas. El uso del nombre genérico "telecom" denota anonimato deliberado.

La identidad del firmante no está validada por ninguna Autoridad de Certificación (CA) externa. Esto rompe la cadena de confianza y confirma que la aplicación no proviene de un desarrollador verificado ni de la tienda oficial Google Play Store.

El número de serie 0x1 y la falta de metadatos son característicos de certificados generados automáticamente por herramientas de construcción de software o kits de phishing, en lugar de entornos de desarrollo profesional.

Figura 78

Detalle del certificado de firma digital

Nota. Sigener certificate, muestra que el malware utiliza un certificado falsificado.

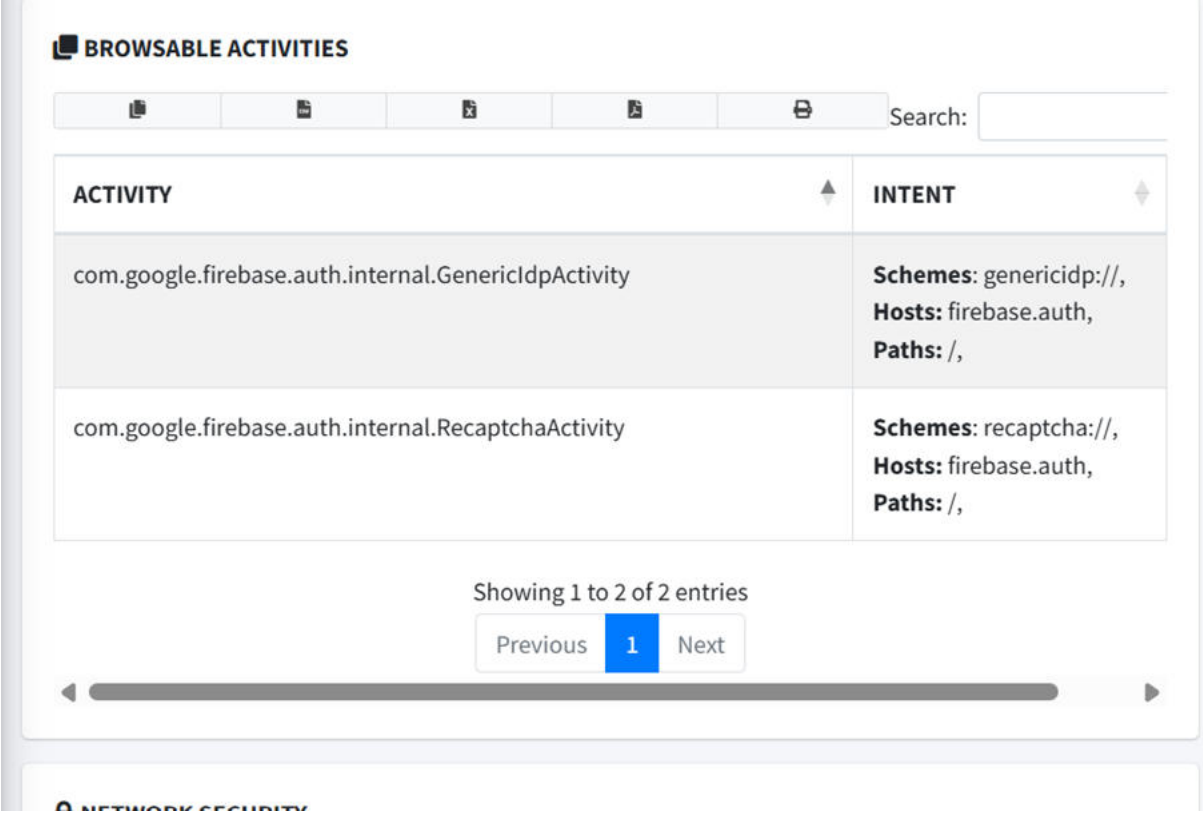
4.1.11. Browsable activities - Caso de estudio 2.

La sección de "Actividades Navegables" identifica componentes exportados pertenecientes al espacio de nombres com.google.firebase.auth.internal. Específicamente GenericIdpActivity y RecaptchaActivity con esquemas de URI genericidp:// y recaptcha://. Estos componentes pertenecen a la implementación estándar del SDK de Google Firebase Authentication. Su presencia confirma que la aplicación integra librerías de Google para gestión de infraestructura, notificaciones o análisis.

Aquí podemos evidenciar que no muestra maldad, muestra la intención del atacante. Usaron librerías estándar.

Figura 79

Actividades exportadas y configuración de intercepción de mensajería SMS



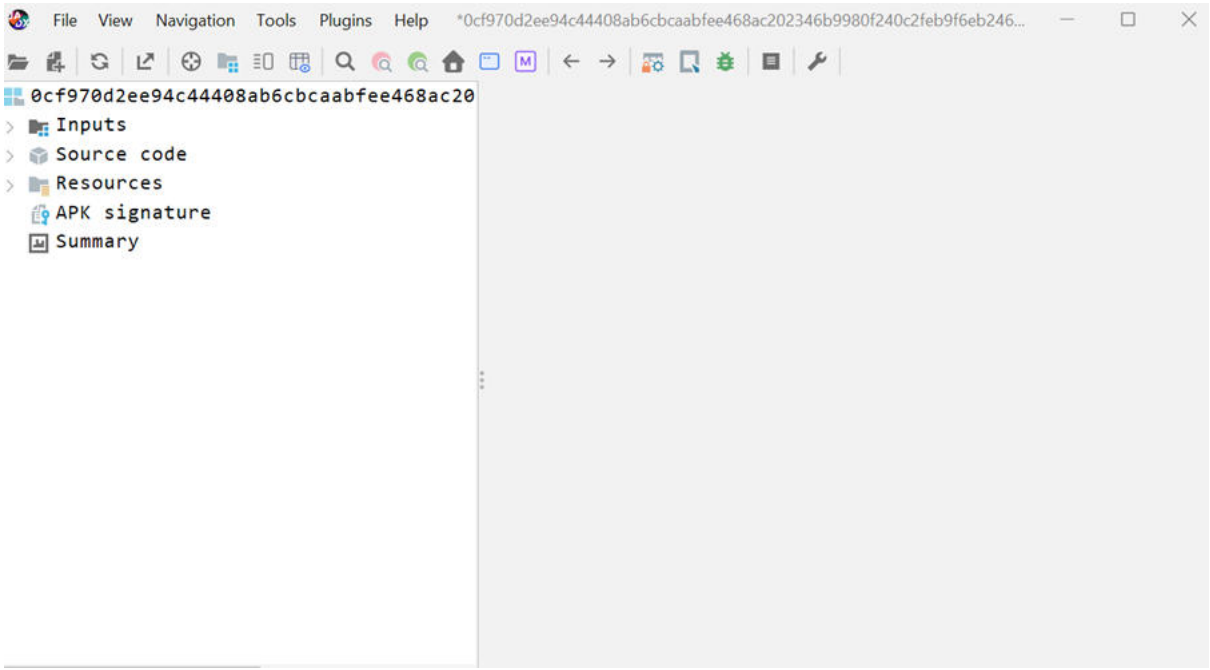
Nota. Browsable activities, se visualiza los detalles de las actividades navegables detectadas por MobSF

4.1.12. ANALISIS CON JADX - Caso de estudio 2.

Con open file cargamos el archivo apk y esperamos a que descompile.

Figura 80

Exploración de la estructura de directorios y recursos de la muestra de malware en Jadx



Nota. Pantalla de inicio de JADX.

De acuerdo al análisis en MobSF en JADX buscamos el texto MAIN_URL

Lo que hallamos: Identificación del Servidor de Comando y Control (C2)

Mediante análisis estático de recursos (strings.xml), se logró desofuscar la variable MAIN_URL, revelando la dirección IP/Dominio de destino de los datos exfiltrados.

Variable: R.string.MAIN_URL

Valor Desencriptado: https://clientproject.online/taptapnew43/

Ruta Completa de Exfiltración:

https://clientproject.online/taptapnew43/TAkaPay/accountlist/updateToken.php

El dominio clientproject.online es ajeno a la infraestructura legítima de la entidad financiera suplantada ("Taptap Send"). Se trata de un servicio de hosting de terceros utilizado para alojar el panel de administración del kit de phishing "TAkaPay". La coincidencia del

subdirectorio /taptapnew43/ con el nombre del paquete de la aplicación nos dice que este servidor fue configurado específicamente para esta campaña de fraude.

Conclusiones:

Con MobSF y Jadx pudimos reclasificar las amenazas del malware técnico a ingeniería social, con el análisis de LoginActivity.java pudimos confirmar el robo de credenciales. La aplicación se trata de una interfaz fraudulenta (“FakeApp”) diseñada para exfiltrar datos sensibles hacia un servidor externo controlado por el atacante, esto lo evidenciamos con la identificación de las rutas PHP y estructuras del kit (TAkaPay).

El análisis manual de recursos en JADX (Strings.xml) revelo la infraestructura criminal oculta, se identificó el dominio (clientproject.online) como el destino final de los datos robados, junto al certificado auto firmado y genérico, podemos confirmar que se trata de una campaña de suplantación y robo de datos.

Figura 81

Identificación de rutinas criptográficas y recuperación de claves de cifrado AES en el código fuente

Nota. Análisis en JADX “MAIN_URL”.

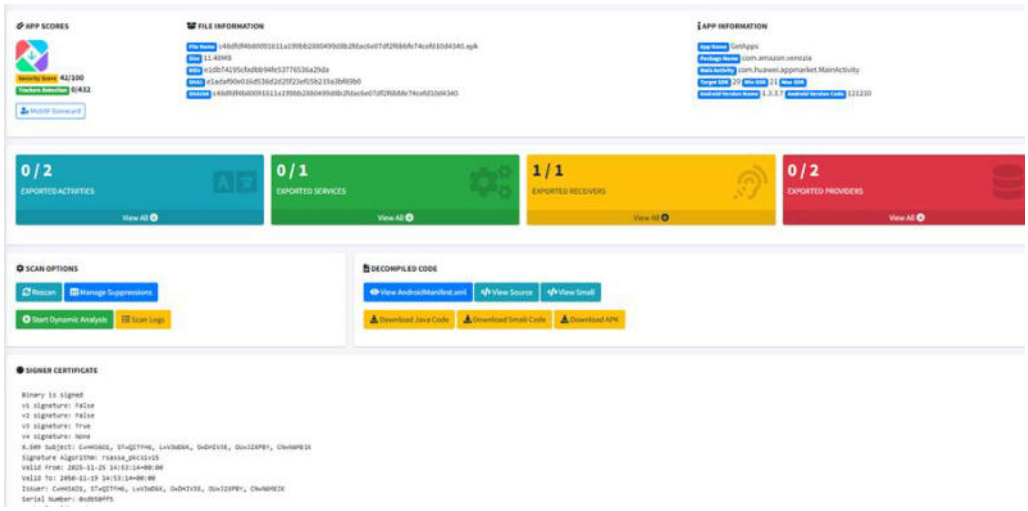
La plataforma de reputación clasifico la muestra como trojanFakeApp, validando que, aunque técnicamente menos compleja que en la muestra anterior (caso de estudio 1) anterior, representa un riesgo de fraude financiero directo.

4.1.13. Análisis de Resultados - Caso de estudio3

Aplicación fraudulenta obtenida de malwareBazaar : Malware de Espionaje y Persistencia (RAT/Dropper) (c48dfdf4b80091611a199bb2880499d8b2fdac6e07df2f6bbfe74cefd10d4340).

Figura 82

Descripción del apk3



Nota. Información general apk 3.

De la misma manera se revisará el archivo AndroidManifest.xml, en este se puede visualizar cómo la aplicación interactúa con el sistema, se trata del documento de identidad de la app.

Figura 83

Archivo AndroidManifest

Nota. Visualización del archivo AndroidManifest.xml

En el análisis del archivo AndroidManisfest podemos observar la línea 17 “com.TH3HARRY.Encryption” podemos etender que se trata de una app que oculta el virus real y se asegura que nadie lo pueda borrar (Loader/Dropper altamente ofuscado).

El objetivo es evadir la detección inicial del antivirus para esto usa la encriptación “TH3HARRY” y posterior inyecta el malware (SpyNote/SpyMax).

“REQUEST_INSTALL_PACKAGES”: Permite a la app instalar otros archivos APK sin pasar por la Play Store. Probablemente descargará o desencriptará la versión completa del RAT y te pedirá instalarla (quizás disfrazada de actualización).

“QUERY_ALL_PACKAGES”: Quiere ver todo lo que tienes instalado. Esto le sirve para saber si tienes apps de bancos (para atacarlas luego) o si tienes herramientas de análisis de seguridad.

Por tanto, se trata de una app maliciosa que usa técnicas de empaquetado (packing) y carga dinámica de código (DCL),

Tabla 4

Matriz de hallazgos del análisis estático

Categoría	Hallazgo Principal	Evidencia Técnica (Artefactos)	Impacto / Riesgo	Nivel de Severidad
Clasificación	Identificado como RAT / Dropper (Familia SpyNote/SpyMax).	Detecciones de VirusTotal: Trojan.SpyNote, Android.Riskware.Packer, TrojanDropper.Agent.	Capacidad de acceso remoto completo y exfiltración de datos sensibles sin consentimiento.	CRÍTICO
Ofuscación y Evasión	Uso de Empaquetado (Packing) y Cifrado personalizado para ocultar el payload.	Clase: com.TH3HARRY.Encryption. Meta-data con basura/cifrado: android.support.v7.dp.Secure (contenido ilegible).	Evasión de análisis estático y antivirus convencionales. El malware se "reconstruye" en memoria.	ALTO
Persistencia	Ejecución automática y resistencia al cierre.	Permisos: RECEIVE_BOOT_COMPLETED (inicio con sistema) y FOREGROUND_SERVIC	Garantiza que el malware siga activo tras reiniciar el	ALTO

		E (servicio en primer plano).	dispositivo o cambiar de aplicación.	
Comportamiento Dropper	Capacidad para instalar/eliminar software silenciosamente.	Permisos: REQUEST_INSTALL_PACKAGES, REQUEST_DELETE_PACKAGES, QUERY_ALL_PACKAGES.	El APK analizado es un "cargador"; su función es descargar e instalar el módulo espía final (payload) con más permisos.	ALTO
Ingeniería Social	Suplantación de Identidad (Spoofing) de proveedores legítimos.	Incoherencia de paquetes: com.amazon.venezia (Amazon) mezclado con componentes com.huawei.appmarket.	Engaña al usuario y a sistemas de seguridad haciéndose pasar por aplicaciones de sistema o tiendas oficiales.	MEDIO

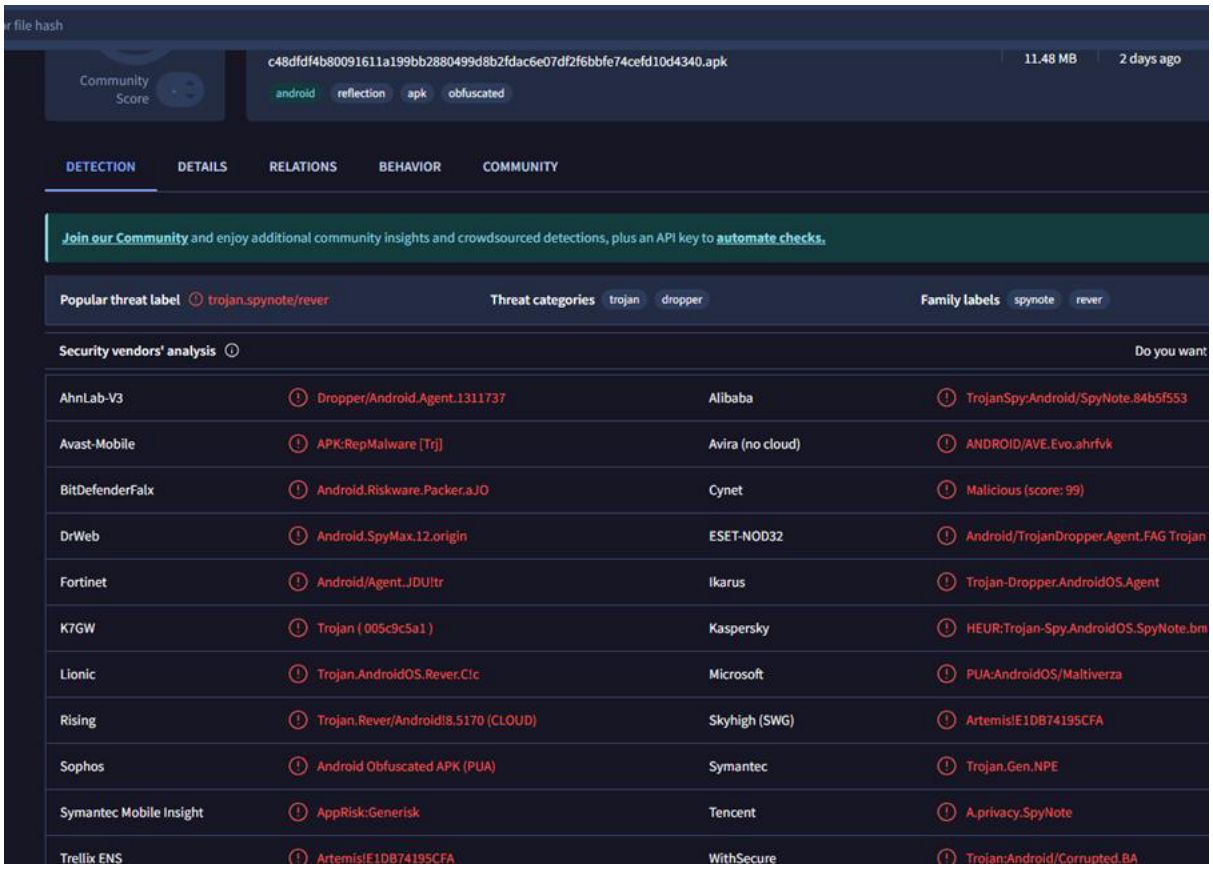
Nota. Se detalla los componentes y permisos críticos hallados en el manifiesto de la aplicación maliciosa.

4.1.14. Análisis de reputación (hashes)

Podemos observar que tiene un puntaje de 22/67 , con un nivel de riesgo crítico, con el análisis antes visto podemos validar que se trata de una app de espionaje, robo de identidad, acceso a cuentas bancarias y vigilancia de datos personales.

Figura 84

Calificación del apk en virus total.



Nota. Reputación en virus total.

4.1.15. Análisis de código fuente (ingeniería inversa).

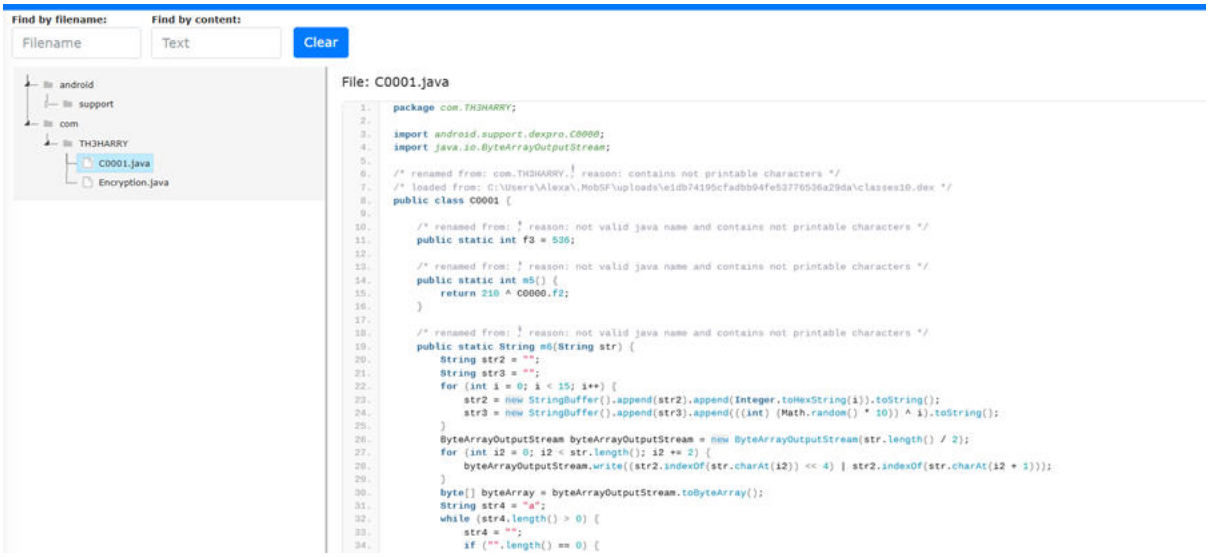
En la línea de código 6 que contiene "/* renamed from: ... reason: contains not printable characters */" podemos darnos cuenta de que el creador del malware usó caracteres ilegibles o especiales para nombrar sus funciones y variables, podemos intuir con el objetivo de romper las herramientas de análisis automático.

En resumen, el análisis del código fuente (C0001.java) revela técnicas avanzadas de anti-análisis. Se identificó el uso de nombres de métodos con caracteres no imprimibles para dificultar la ingeniería inversa. Además, se hallaron rutinas de manipulación de bits (XOR y

Bit Shifting) diseñadas para descifrar dinámicamente la carga útil (payload) oculta en los metadatos del Manifiesto. Esto confirma que la aplicación actúa como un 'Loader' cifrado.

Figura 85

Código fuente



Nota. Análisis de código fuente.

4.1.16. Certificado de firma APK 2

En este análisis de detecto que existe una aleatoriedad en los campos de identificación (Organización, País, Nombre Común) es una técnica de ofuscación estándar utilizada por herramientas de creación de RATs (como SpyNote) para evadir detecciones basadas en firmas de desarrolladores conocidos. No existe relación criptográfica alguna con Amazon o Huawei (como vemos en el archivo AndroidManifest.xml).

Tabla 5

Información general

Parámetro Analizado	Valor Encontrado	Interpretación Forense
Sujeto (Distinguished Name)	C=H4S6O1, O=DHIV3E, CN=N6MEIK	**INDICADOR DE COMPROMISO.** Los campos contienen cadenas alfanuméricas aleatorias sin significado semántico. Esto evidencia el uso de un generador automatizado de malware (Builder) y contradice la supuesta identidad de la aplicación (Amazon/Huawei).
Autoridad Emisora (Issuer)	Idéntico al Sujeto (C=H4S6O1…)	El certificado es **Autofirmado**. No existe una cadena de confianza (CA) que valide la identidad del desarrollador.

Huella Digital (SHA-1)	485a39226b5b2d6c7e8d6ada36893eccdc29f8c2	**IoC (Indicador de Compromiso).** Este hash identifica unívocamente a la clave privada del atacante usada en esta campaña específica.
Algoritmo de Firma	SHA512withRSA (2048 bits)	Uso de criptografía estándar para cumplir los requisitos mínimos de instalación de Android, pero sin validez de identidad real.
Temporalidad	Creado: 2025-11-25	La amenaza es **reciente**. El artefacto fue compilado y firmado apenas días antes de su detección.

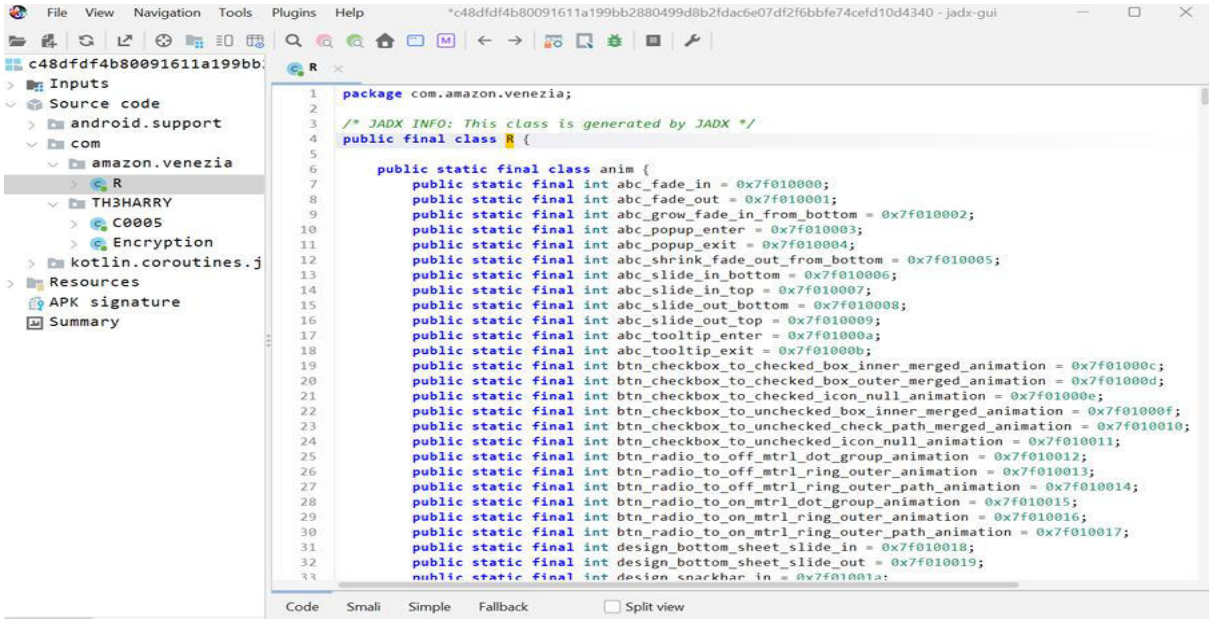
Nota. Información general

4.1.17. ANALISIS CON JADX

Seleccionando la opción de “open file”, cargamos el archivo apk y esperamos a que el mismo descompile.

Figura 86

Análisis con JADX



Nota. Vista principal del análisis con JADX.

Se va a analizar la ruta com/TH3HARRY /Enryption , podemos evidenciar lo siguiente:

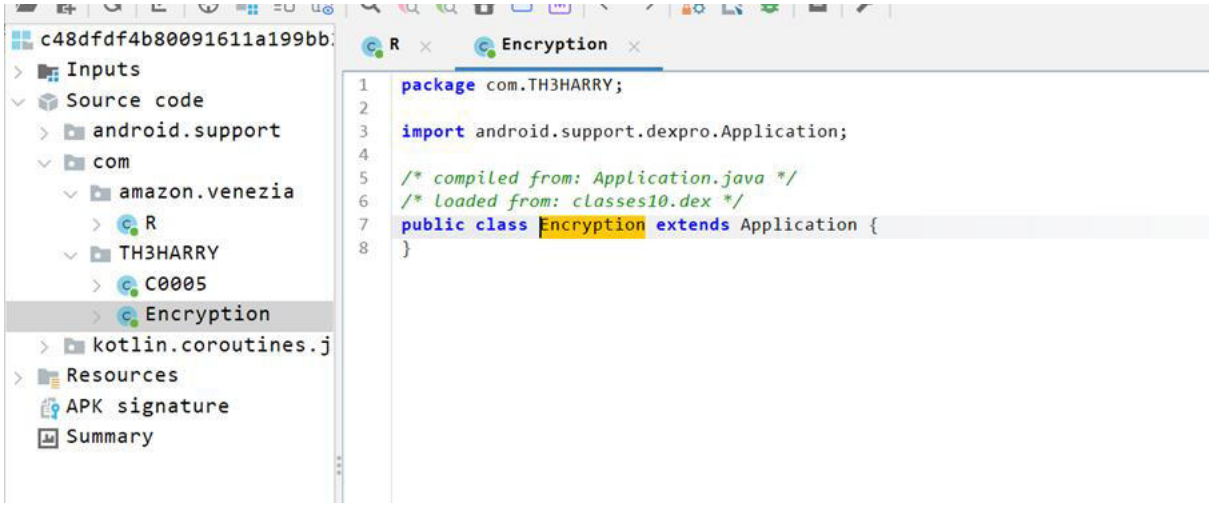
El malware está utilizando Herencia para ocultar su código de inicio.

La clase Encryption no tiene código propio, PERO hereda todo de su "padre": android.support.dexpro.Application.

Esto significa que el código malicioso real que se ejecuta al inicio no está aquí, sino en esa clase padre (dexpro).

Figura 87

Clase Encryption



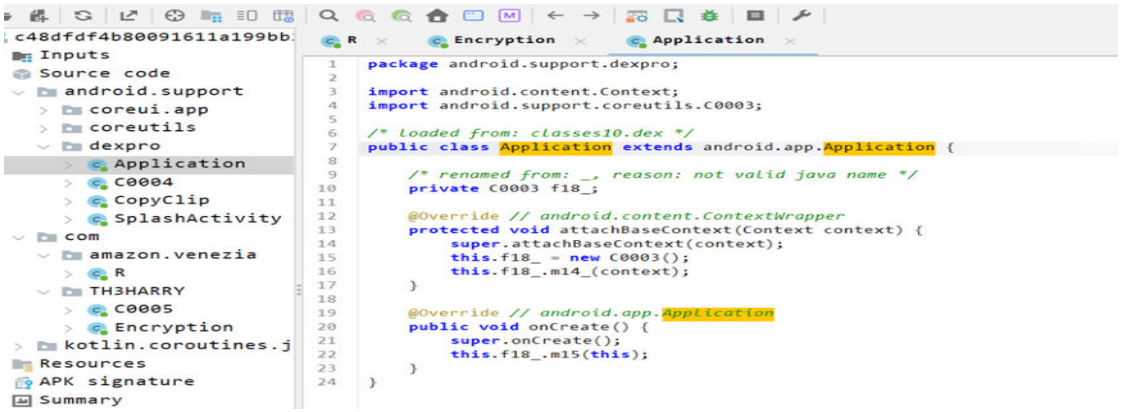
Nota. Visualización de la clase Encryption,

En la clase “Aplication” podemos observar en la línea 13 “attachBaseContext” , en Android, el método “attachBaseContext” se ejecuta antes que “onCreate” y mucho antes de que se cargue cualquier interfaz gráfica o actividad.

Los creadores de malware (y Packers) usan este método para "preparar el campo". Aquí es donde inyectan su propio código en la memoria del sistema para que, cuando la app arranque "normalmente", el virus ya esté activo.

Figura 88

Clase Aplication



Nota. Visualización de la clase Aplication.

La correlación de la manipulación en Virus Total (22/67 detecciones) con el análisis estático confirma que se trata de una Android/Trojan.SpyNote. Esta amenaza compromete la privacidad del usuario mediante espionaje.

4.2. Análisis Dinámico

Para el análisis dinámico, la muestra fue detectada como peligrosa por el antivirus al momento de descomprimirlo, el objetivo principal de este análisis es evaluar el comportamiento dinámico de la Muestra 3 en un entorno controlado. El laboratorio para el análisis dinámico es compuesto por una máquina virtual Kali Linux, una máquina Android x86 y las herramientas de análisis correspondientes. Se logró establecer la conexión ADB desde la máquina Kali hacia la máquina Android para la instalación y pruebas, validando la conectividad mediante comandos ADB (adb connect, adb devices) y pruebas de red (ICMP).

Se creó un snapshot previo a la instalación de la Muestra 3, el mismo fue instalado desde la máquina de Kali mediante el siguiente comando: “adb install /home/kali/LabAndroid/APK/Muestra3.apk”.

Figura 89

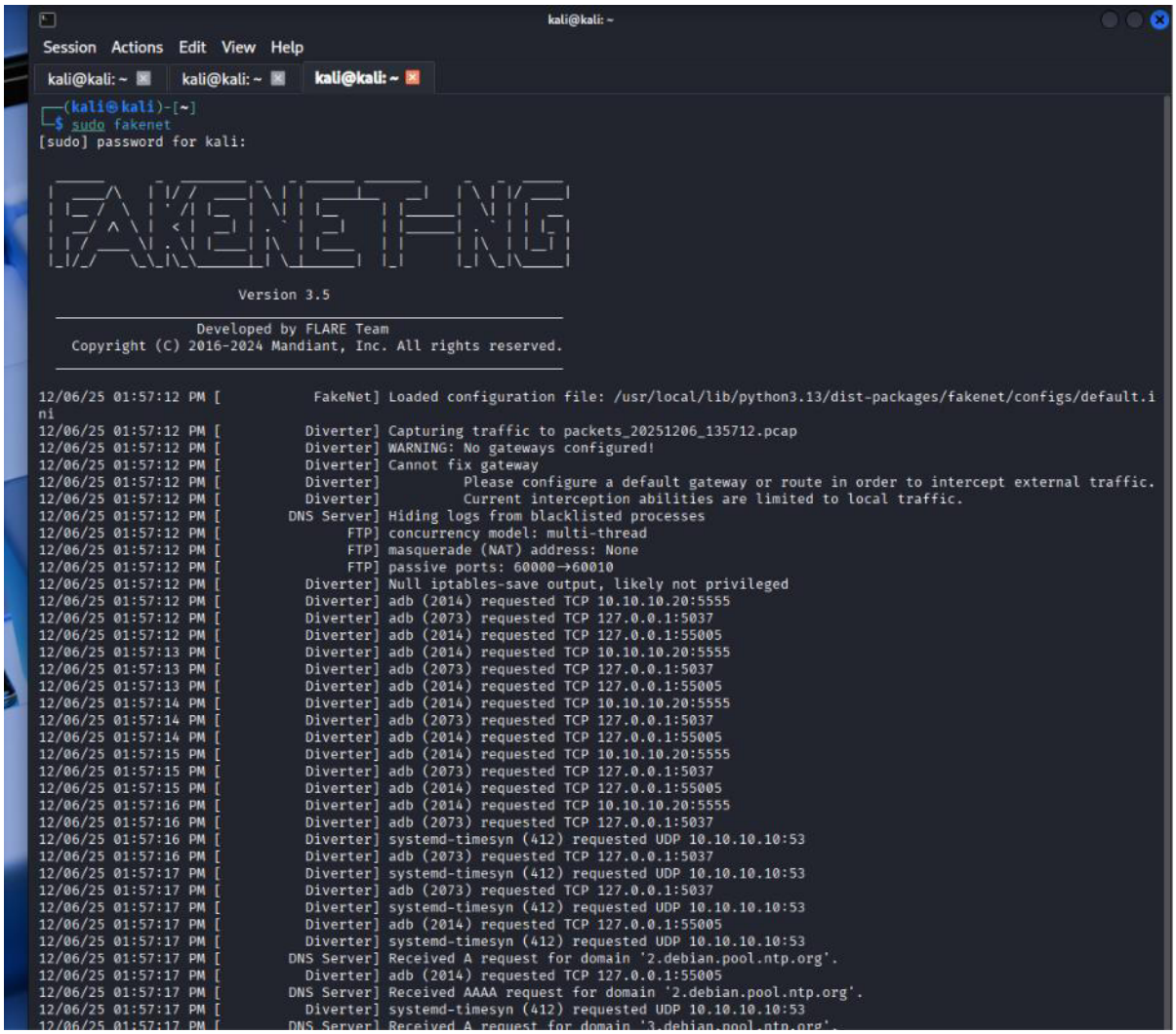
Instalación del apk malicioso.

Nota. Instalación de la muestra en Android mediante adb.

Para el monitoreo y análisis dinámico se tienen preparados 3 terminales en Kali para el monitoreo: (1) FakeNet-NG para poder interceptar y redirigir el tráfico, (2) Wireshark para captura de paquetes en la interfaz con IP 10.10.10.10 realizando el filtro en el buscador con ip.addr==10.10.10.20, y (3) adb logcat para registrar los eventos del sistema Android.

Figura 90

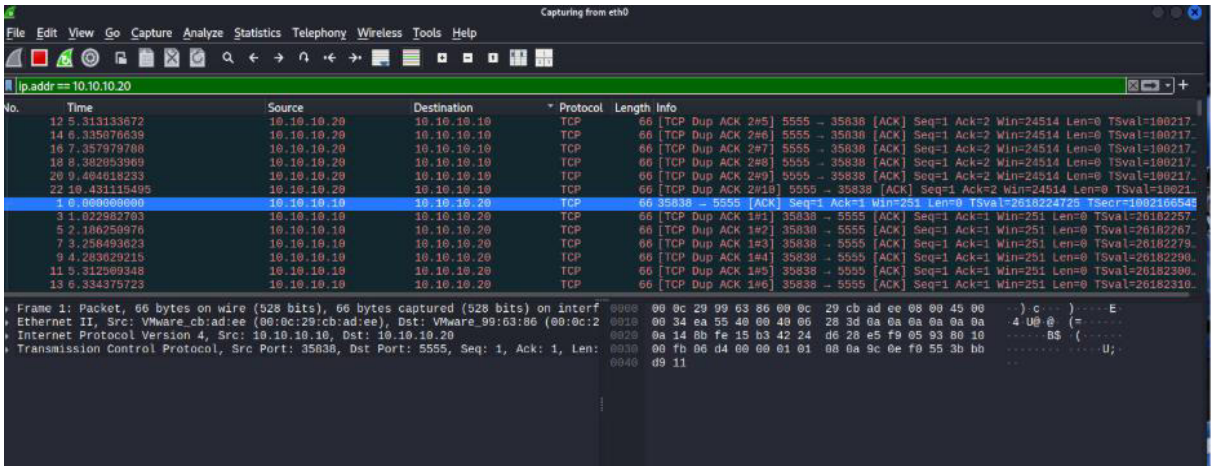
Terminal Fakenet para simular una red.



Nota. Terminal Fakenet.

Figura 91

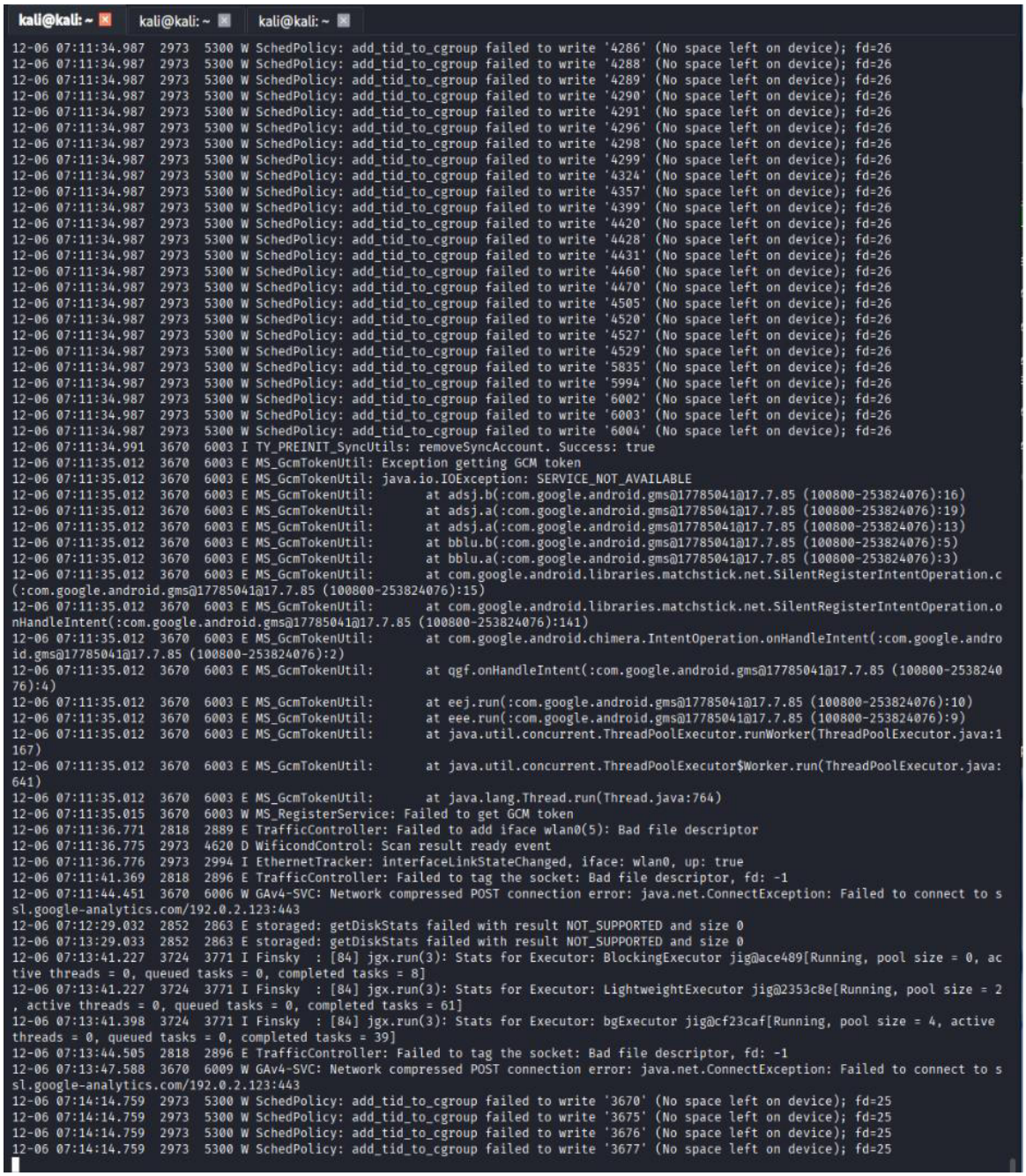
Pantalla con Wireshark



Nota. Terminal Fakenet.

Figura 92

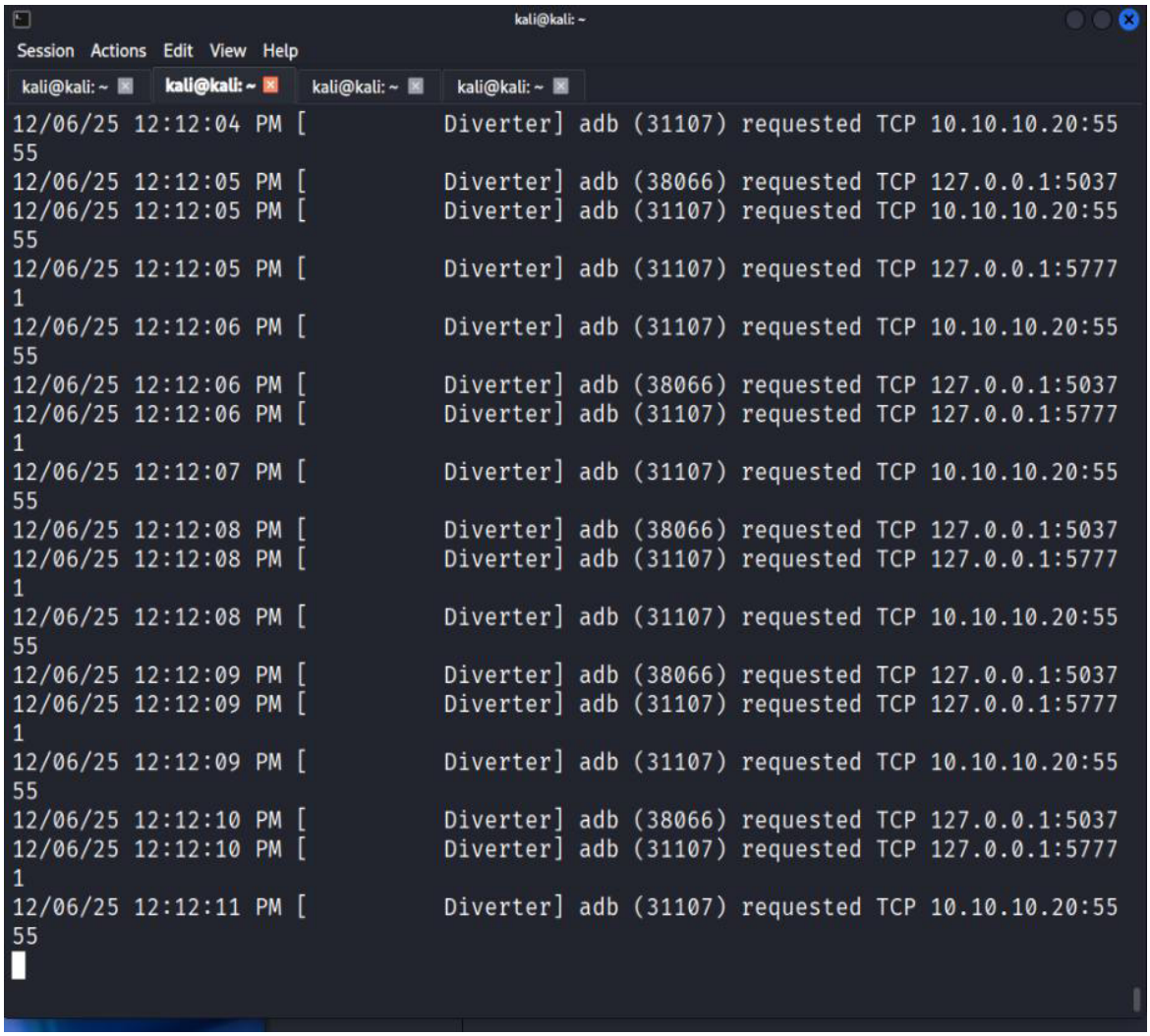
Terminal con adb Logcat.



Nota. Terminal con adb Logcat.

Figura 93

Terminal para ejecutar comandos.



Nota. Terminal Para ejecutar comandos.

Se logra confirmar que el nombre del paquete identificado es malicioso: 'com.bilbiliyyq.bbbyq1025.d176304597462225934', el cual difiere del paquete oficial 'tv.danmaku.bili'. La nomenclatura observada es consistente con un malware ofuscado, más en concreto adware y droppers de origen chino.

Antes de la ejecución, 'adb shell ps' no indica procesos relacionados y 'adb shell netstat -tunp' no evidencia sockets abiertos, lo cual es lo esperado al momento que la aplicación no corre servicios en background ni auto-start; el malware se activa al abrir la app.

Figura 94

Terminal para ejecutar comandos.

Nota. Terminal Para ejecutar comandos.

Al abrir la aplicación, esta solicita permisos de acceso al almacenamiento y presenta una interfaz de estilo oriental (chino), con mensajes engañosos, inclusive se muestra un correo de contacto 'bzhl001@pm.me' y la opción de descargar una versión externa. Este comportamiento indica un dropper/adware basado en WebView.

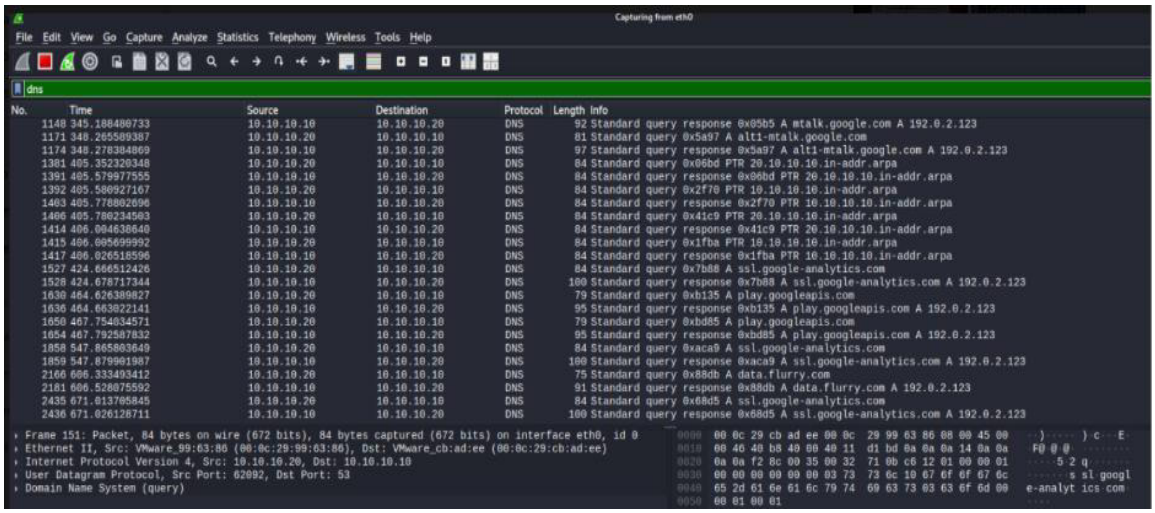
Figura 95

Terminal para ejecutar comandos.

Nota. Terminal Para ejecutar comandos aquí ejeucutamos los comandos para obtener el nombre de la aplicación que se instalo.

Figura 96

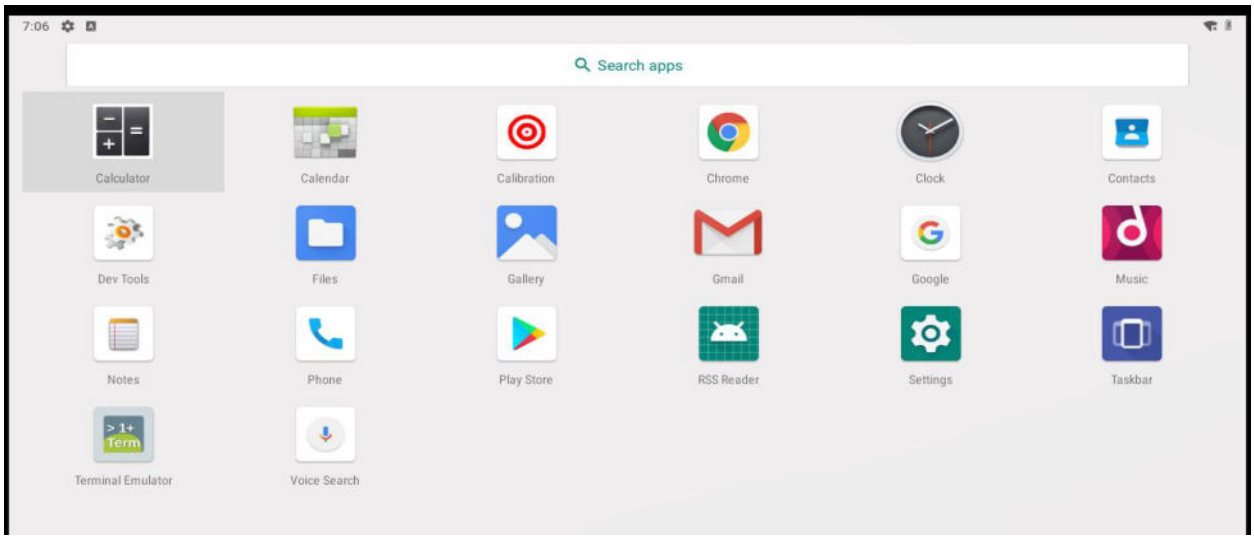
Terminal de Wireshark.



Nota. Terminal de Wireshark con trafico inusual filtrado mediante el comando DNS en la barra de filtros.

Figura 97

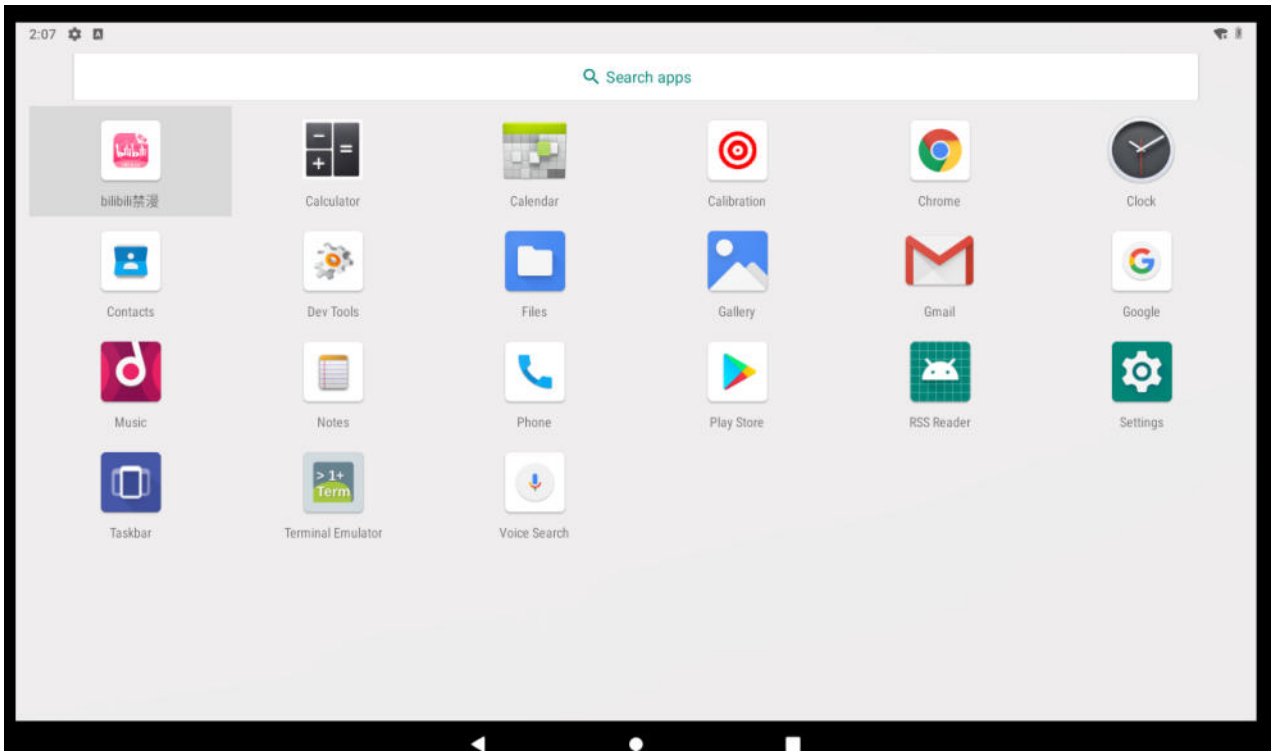
Interfaz de Android antes de la instalación



Nota. Interfaz de Android antes de la instalación

Figura 98

Interfaz de Android después de la instalación



Nota. Interfaz de Android despuess de la instalación

Figura 99

Interfaz de la aplicación fraudulenta.

Nota. Interfaz de la aplicación fraudulenta.

La pantalla que se muestra textualmente dice:

APP线路检测失败

解决方案:

请检测您的网络是否正常...

请联系在线客服...

请打开网络获取地址...

邮箱: bzhl001@pm.me

Lo que traducido significa:

"Fallo en la detección de red"

"Verifique su red"

"Contacte al servicio al cliente"

"Envíe un correo a bzhl001@pm.me"

"Descargue una versión nueva desde el sitio oficial"

"Por favor encienda su red para obtener la dirección"

Conclusión:

Con esto se concluye que es malware, ya que las aplicaciones originales y legitimas no piden que contactes a un correo privado y menos a un pm.me (ProtonMail).

Con la herramienta FakeNet activado, las consultas DNS hacia dominios desconocidos fueron redirigidas a 192.0.2.123, impidiendo contacto con infraestructura real. Sin embargo, se logran observar resoluciones hacia múltiples dominios maliciosos, patrón típico de distribución de payloads y encubrimiento de C2.

CAPITULO 5:
CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

Se concluye que el malware bancario moderno ha desplazado su vector de ataque desde la explotación de vulnerabilidades hacia el abuso de permisos de accesibilidad y superposición de pantalla. Esto implica que las amenazas como Coper pueden obtener control total del dispositivo y exfiltrar credenciales sin necesidad de acceso root, complicando su detección por el sistema operativo.

El estudio de la muestra SpyNote demostró que el uso de Carga Dinámica de Código (DCL) y renombramiento con caracteres no imprimibles hace que el análisis estático sea insuficiente por sí solo. Los Loaders actuales reconstruyen su carga maliciosa directamente en la memoria RAM, evadiendo eficazmente las firmas de antivirus tradicionales que solo escanean el archivo físico.

Se determinó que la incoherencia entre el nombre del paquete y la firma digital es el indicador de compromiso más confiable para detectar FakeApps y campañas de Phishing. A diferencia del malware complejo, estas estafas dependen de la ingeniería social y de infraestructura web precaria (kits de phishing), siendo fácilmente identificables mediante la auditoría de sus certificados y rutas de red.

La Muestra 3 se clasifica como adware/dropper de origen chino basado en WebView, con capacidades para mostrar contenido fraudulento, solicitar permisos peligrosos y contactar infraestructura maliciosa (DGA + CDN). Aunque no evidencia persistencia ni servicios en segundo plano, su actividad de red y solicitud de permisos la posicionan como una amenaza

activa. El uso de FakeNet en el laboratorio impidió la descarga de payloads adicionales, permitiendo capturar IoCs sin comprometer el entorno.

El análisis dinámico permitió detectar comportamiento malicioso típico de adware y droppers distribuidos mediante aplicaciones falsas. La aplicación analizada no mostró persistencia ni ejecución en segundo plano, pero sí realizó múltiples intentos de contacto con infraestructura de red sospechosa. Estos hallazgos validan su clasificación como amenaza activa.

Recomendaciones:

Como se ha visto que las amenazas modernas como SpyNote y Loaders utilizan técnicas DCL (Carga Dinámica de Código) y ofuscación para evadir las firmas estáticas, se recomienda a los analistas y organizaciones institucionales no depender solo de un antivirus tradicional. Es necesario adoptar una metodología de análisis hibrido que contenga a ejecución dinámica en entornos controlados (Sandboxing), permitiendo activar la carga maliciosa en la memoria RAM para observar el comportamiento real.

Se recomienda establecer políticas de seguridad en dispositivos corporativos que bloqueen o alerten sobre la activación de permisos no autorizados por parte de aplicaciones no certificadas, así mismo se debe implementar mecanismos de detección de overlays activos, en las aplicaciones financieras para mitigar el robo de credenciales en tiempo real.

Se recomienda antes de instalar una aplicación realizar o incorporar validaciones automatizadas de la cadena de confianza de los certificados digitales.

Apéndice A. Laboratorio práctico 1

En el siguiente link, Laboratorios se puede descargar las máquinas virtuales utilizadas.

Referencias Bibliográficas

Android Developers. (2023). Android security features. Google. https://source.android.com/security

Android-x86 Project. (2023). Android-x86: Android OS for x86 platforms [Software]. https://www.android-x86.org/

Andrango, S. (2022). Análisis comparativo de malware en teléfonos inteligentes Android.

CERT. (2021). Guía de tácticas, técnicas y procedimientos para análisis de amenazas.

Cervera, P., & Goussens, M. (2024). Evaluación de vulnerabilidades en dispositivos IoT médicos.

Cuckoo Sandbox. (2022). Cuckoo Sandbox Documentation. https://cuckoosandbox.org

Dinaburg, A., Royal, P., Sharif, M., & Lee, W. (2008). Ether: Malware analysis via hardware virtualization extensions. Proceedings of the 15th ACM Conference on Computer and Communications Security, 51–62.

EC-Council. (2022). Ethical hacking and malware analysis guidelines.

Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2012). A survey on automated dynamic malware analysis techniques and tools. ACM Computing Surveys, 44(2), 1–42. https://doi.org/10.1145/2089125.2089126

Enck, W., Octeau, D., McDaniel, P., & Chaudhuri, S. (2014). A study of Android application security. USENIX Security Symposium, 21–21.

Europol. (2023). Mobile Malware Activity Report. Europol Cybercrime Centre.

Gartner. (2024). Threat intelligence report: Mobile malware trends. Gartner Research.

Google. (2023). Android Debug Bridge (adb) [Software]. https://developer.android.com/tools/adb

Google Research. (2024). Android ecosystem security overview.

INCIBE. (2023). Panorama actual de amenazas móviles. https://incibe.es

ISO/IEC. (2012). Directrices para la identificación, recolección y preservación de evidencia digital (ISO/IEC 27037:2012).

ISO/IEC. (2013). Information Security Management Systems – Requirements (ISO/IEC 27001:2013).

ISO/IEC. (2015). Investigación digital y respuesta ante incidentes (ISO/IEC 27043:2015).

INTERPOL. (2023). Cybercrime and mobile malware report.

Kaspersky. (2024). Taxonomía de malware móvil.

Kaspersky Lab. (2023). Mobile malware evolution report.

Kili, A. (2023). How to Install Android Debug Bridge (adb) in Linux. Tecmint. https://www.tecmint.com/install-android-debug-bridge-linux/

Krishnan, A. (2023). Mobile Security Framework (MobSF) [Software]. https://mobsf.github.io/

Mandiant. (2023). FakeNet-NG: Dynamic network analysis tool [Software]. https://github.com/mandiant/flare-fakenet-ng

MISTIC TFM. (2021). Análisis de malware en dispositivos móviles: técnicas estáticas y dinámicas.

Montenegro, L., Vaca, J., & Valle, R. (2025). Auditoría de ciberseguridad en dispositivos médicos IoMT.

NIST. (2022). NIST SP 1800-21: Mobile Device Security. https://doi.org/10.6028/NIST.SP.1800-21

OWASP. (2023). OWASP Mobile Security Testing Guide (MSTG). https://owasp.org/www-project-mobile-security-testing-guide/

Pham, C. (2023). APKTool: Android application reverse engineering tool [Software]. https://ibotpeaches.github.io/Apktool

Plohmann, D., Wicherski, G., Brandt, M., Backes, M., & Holz, T. (2016). A comprehensive measurement study of domain generating malware. USENIX Security Symposium, 263–278.

Rapid7. (2024). Threat intelligence for mobile malware.

Regan, C., & Belcic, I. (2022). Tipos de malware y técnicas de evasión.

Sáenz de Santa María Fernández, J. M. (2023). Seguridad en la información. Universidad de León. https://servicios.unileon.es/formacion-pas/files/2023/04/Seguridad-en-la-informacion.pdf

Sikorski, M., & Honig, A. (2020). Practical malware analysis: The hands-on guide to dissecting malicious software. No Starch Press.

Skylot. (2023). JADX: Dex to Java decompiler [Software]. https://github.com/skylot/jadx

Symantec. (2023). Mobile threat landscape report.

Tavella, F. (2022). Herramientas de Linux para el análisis de malware. WeLiveSecurity. https://www.welivesecurity.com/la-es/2022/01/05/herramientas-de-linux-para-el-analisis-de-malware

Trend Micro Research. (2023). Mobile Malware Report.

Vaca, J., & Valle, R. (2024). Evaluación de vulnerabilidades en ecosistemas móviles IoT.

Vásquez, J. (2013). Manual de Wireshark en español. Blogspot. http://manualwireshark.blogspot.com/2013/06/manual-de-wireshark-en-espanol.html

VMware. (2023). VMware Workstation Pro [Software]. https://www.vmware.com

Wireshark Foundation. (2023). Wireshark: Network protocol analyzer [Software]. https://www.wireshark.org