**MÁSTER EN INVESTIGACIÓN EN CIBERSEGURIDAD**

Trabajo de Fin de Máster

Análisis de troyano Chameleon en Android

Autor: Raúl Núñez García

Tutor: Ángel Manuel Guerrero Higueras

Julio 2024

|  |  |  |
| --- | --- | --- |
| **UNIVERSIDAD DE LEÓN**  **Escuela de Ingenierías Industrial, Informática y Aeroespacial**  **MÁSTER EN INVESTIGACIÓN EN CIBERSEGURIDAD**  **Trabajo de Fin de Máster** | | |
| **ALUMNO:** Raúl Núñez García | | |
| **TUTOR:** Ángel Manuel Guerrero Higueras | | |
| **TÍTULO:** Análisis de troyano Chameleon en Android | | |
| **CONVOCATORIA: j**ulio 2024 | | |
| **RESUMEN:**  En este Trabajo de Fin de Máster (TFM) se ha llevado a cabo un análisis detallado de una muestra de malware de tipo troyano que forma parte de la familia conocida como Chameleon, y que ha sido desarrollado infectar dispositivos de la plataforma Android.  Se ha aplicado un análisis estático del código del malware con una fase de deofuscación previa para hacer el código más legible y fácil de comprender. En este análisis se ha estudiado el código del malware para tratar de extraer una lista de posibles tareas que puede llevar a cabo durante su ejecución y los métodos que utiliza para poder tener éxito en la infección y sus pasos posteriores.  Paralelamente se ha llevado a cabo una fase de análisis dinámico, ejecutando el malware en un entorno controlado con dispositivos emulados, recabando información acerca de su comportamiento y realizando comparativas respecto a los datos extraídos en el análisis estático.  Finalmente, se ha documentado toda la información relevante extraída durante los procesos mencionados, tratando de plasmar el funcionamiento en detalle del malware, su peligrosidad y cómo encaja dentro del ecosistema actual de troyanos y malware en Android. | | |
| **Palabras clave: malware, Android, troyano, infostealer, Chameleon** | | |
| **Firma del alumno:** | **VºBº Tutor:** | **VºBº Cotutor:** |

**Abstract**

In this Master Thesis Project, a detailed analysis of a malware sample has been carried out. It revolves around a malware of trojan type that belongs to the family known as Chameleon, and it has been developed for targeting and infecting Android devices.

Static analysis has been applied over the code of the sample, with a previous phase of deobfuscation to make the code more legible and easier to understand. In this static phase the code has been analyzed in detail to try and understand the tasks the malware could use during its execution on an infected device, as well as the methods it employs to try and achieve success in its objectives.

In parallel, a dynamic analysis phase has been carried out, executing the sample in a controlled environment with emulated devices and compiling all the information derived from the workings and behavior the malware exhibits when running, comparing the information extracted with the one obtained during the static analysis.

Finally, all the information collected during the processes mentioned has been documented to try and showcase the detailed workings of the malware, the dangers it poses to users and how it fits within the current Android malware and trojan ecosystem.

**Índice de contenidos**

[Índice de figuras 4](#_Toc169088903)

[Índice de tablas 5](#_Toc169088904)

[Glosario de términos 6](#_Toc169088905)

[Introducción 9](#_Toc169088906)

[Capítulo 1: Estudio del problema 12](#_Toc169088907)

[1.1 EL CONTEXTO DEL PROBLEMA 12](#_Toc169088908)

[1.2 EL ESTADO DEL ARTE 14](#_Toc169088909)

[1.2.5 Conclusión 15](#_Toc169088910)

[1.3 DEFINICIÓN DEL PROBLEMA 15](#_Toc169088911)

[1.4 MALWARE ELEGIDO 15](#_Toc169088912)

[1.4.2 Tecnologías utilizadas (a rellenar) 15](#_Toc169088913)

[Capítulo 2: Gestión del proyecto 17](#_Toc169088914)

[2.1 ALCANCE DEL PROYECTO 17](#_Toc169088915)

[2.2 PLAN DE TRABAJO 17](#_Toc169088916)

[2.2.1 Identificación de tareas 17](#_Toc169088917)

[2.2.2 Estimación de tareas 18](#_Toc169088918)

[2.2.3 Planificación de tareas 18](#_Toc169088919)

[2.3 GESTIÓN DE RECURSOS 19](#_Toc169088920)

[2.3.1 Especificación de recursos 19](#_Toc169088921)

[2.3.2 Presupuesto 19](#_Toc169088922)

[2.4 GESTIÓN DE RIESGOS 19](#_Toc169088923)

[2.4.1 Identificación de riesgos 19](#_Toc169088924)

[2.4.2 Análisis de riesgos 19](#_Toc169088925)

[2.5 LEGISLACIÓN Y NORMATIVA 20](#_Toc169088926)

[Capítulo 3: Solución 21](#_Toc169088927)

[3.1 DESCRIPCIÓN DE LA SOLUCIÓN 21](#_Toc169088928)

[3.2 EL PROCESO DE DESARROLLO 21](#_Toc169088929)

[3.2.1 Análisis estático 21](#_Toc169088930)

[3.2.1.1 Especificación de requisitos 21](#_Toc169088931)

[3.2.1.2 Consideraciones adicionales 21](#_Toc169088932)

[3.2.2 Análisis dinámico 21](#_Toc169088933)

[3.2.2.1 Especificación de requisitos 21](#_Toc169088934)

[3.2.2.2 Consideraciones adicionales 21](#_Toc169088935)

[3.3 RESULTADOS DEL DESARROLLO 21](#_Toc169088936)

[Capítulo 4: Evaluación 22](#_Toc169088937)

[Capítulo 5: Conclusión 22](#_Toc169088938)

[5.1 APORTACIONES REALIZADAS 22](#_Toc169088939)

[5.2 AMPLIACIONES FUTURAS 22](#_Toc169088940)

[5.3 PROBLEMAS ENCONTRADOS 22](#_Toc169088941)

[5.4 OPINIONES PERSONALES 22](#_Toc169088942)

[Bibliografía 23](#_Toc169088943)

[Anexo A: Control de versiones 24](#_Toc169088944)

[Anexo B: Seguimiento del proyecto 25](#_Toc169088945)

[B.1 FORMA DE SEGUIMIENTO 25](#_Toc169088946)

[B.2 PLANIFICACIÓN INICIAL 25](#_Toc169088947)

[B.3 PLANIFICACIÓN FINAL 25](#_Toc169088948)

**Índice de figuras**

1 4

**Índice de tablas**

**Glosario de términos**

Malware: también conocido como código dañino, se trata de un programa informático que realiza acciones en un sistema sin el conocimiento ni autorización del usuario y con objetivos maliciosos como puede ser el robo de datos o la toma de control del sistema.

Troyano: tipo de malware que en la mayoría de los casos se hace pasar por legítimo, pero realmente permite a los atacantes tomar el control del dispositivo infectado.

Infostealer: tipo de malware diseñado con el objetivo de realizar una exfiltración de datos del dispositivo infectado para obtener información que pueda resultar de interés a los atacantes.

Keylogger: tipo de malware que destaca por su funcionalidad de registro de teclas utilizadas, permitiendo la exfiltración de las cadenas de texto creadas por el usuario del dispositivo del usuario con fines maliciosos como puede ser el fraude. En algunos casos funciona como uno de los componentes de un malware infostealer.

Ofuscación: técnica de modificación del código fuente de un sistema que busca aumentar la dificultad de su comprensión mediante el empleo de estrategias como la reordenación del código o el empleo de cifrado. Este sistema puede ser utilizado por atacantes para tratar de hacer más difícil la labor de análisis estático del malware.

APK: tipo de archivo empaquetado utilizado en Android para las aplicaciones que se distribuyen en el ecosistema. Contiene todos los materiales necesarios para la instalación de una aplicación, incluyendo su código, recursos y certificados. Se construye a partir de código fuente en Java o Kotlin.

ART: acrónimo de Android Runtime, es la máquina virtual que utilizan los dispositivos Android. Es el sistema sucesor de Dalvik y hace las veces de máquina virtual Java, pero cuenta con su propio sistema de bytecode. Realiza la traducción entre el bytecode generado y las instrucciones nativas del dispositivo.

Kernel: núcleo del sistema operativo de un sistema, sirviendo de interfaz entre el hardware y los procesos que se ejecutan. Suele ser el componente del sistema que tiene el máximo privilegio otorgado, y por lo tanto control total sobre las operaciones a realizar.

DEX: iniciales de Dalvik Executable, formato de bytecode que utilizan las máquinas virtuales de ART y Dalvik de las aplicaciones Android. El compilador DEX es el encargado de traducir el código fuente Java o Kotlin en DEX.

JAR: formato contenedor Java Archive, que sirve para empaquetar clases en lenguaje Java, así como sus recursos y metadatos. Se utiliza para la distribución de aplicaciones Java. Puede ser de tipo ejecutable, en cuyo caso indica una clase de entrada en su manifiesto.

APT: acrónimo en inglés de Advanced Persistent Thread. Hace referencia a grupos de atacantes informáticos de gran sofisticación y poseedores de cantidad de recursos para realizar sus operaciones. Suelen ser grupos con financiación y apoyo de grandes organizaciones, algunas de ellas a menudo estados.

IoC: indicadores de compromiso en español, son conjunto de datos y evidencias generados por una actividad realizada que indican las características de una amenaza. Pueden englobar ficheros creados, procesos ejecutados, dominios o direcciones IP contactadas entre otros.

MD5: algoritmo criptográfico de reducción que genera un valor de 128 bits a partir de una entrada de valor arbitrario. Se utiliza a entre otros a modo de checksum para verificar la integridad de los datos.

DDoS: ataque distribuido de denegación de servicio, consistente en la sobrecarga de una infraestructura informática como puede ser un sitio web o un servidor y provocando problemas de disponibilidad en este. En gran parte de los casos, estos ataques utilizan redes de dispositivos infectados y controlados remotamente, conocidos como botnets, para llevar a cabo las ofensivas.

Cryptojacking: tipo de delito informático en el que los dispositivos infectados son utilizados por los atacantes para el minado de criptomonedas sin el conocimiento de la víctima que posee el dispositivo y con el fin de obtener un beneficio económico.

DNS: sistema de nombres de dominio jerárquico y distribuido que establece un sistema de nombres y direcciones para dispositivos que forman parte de Internet. Delega la responsabilidad de gestionar nombres y relaciones para cada dominio a sus servidores DNS autoritativos correspondientes.

API: también conocida como interfaz de programación, es un grupo de definiciones y protocolos que sirven para la integración de diferentes componentes de software.

C&C: del inglés Command & Control, también conocido como C2, alude a la infraestructura utilizada por los atacantes para controlar los equipos infectados de manera remota a través de una comunicación entre los servidores del C2 y los dispositivos.

**Introducción**

**PLANTEAMIENTO DEL PROBLEMA**

**OBJETIVOS**

El objetivo principal de este trabajo es el análisis en profundidad de una muestra de malware de tipo troyano desarrollado para plataformas Android. Los subobjetivos que se esperan cumplir mediante la realización del proyecto son los siguientes:

O1 – Enmarcar el malware elegido y la familia de la que procede dentro del ecosistema actual de troyanos de plataformas Android.

O2 - Realizar un análisis de los mecanismos de distribución del malware.

O3 - Llevar a cabo un análisis estático del código malware para tratar de comprender sus capacidades y modus operandi.

O3 – Realizar un análisis dinámico del malware en un entorno controlado, que permita observar cómo se comporta una vez instalado y ejecutado en un dispositivo infectado

O4 – Comparar los datos obtenidos en el análisis estático con los extraídos en la fase dinámica para confirmar el comportamiento esperado y extraer posibles diferencias en el caso de que estas sean identificadas.

O5 – Obtener una serie de indicadores de compromiso y procedimientos que permitan identificar el malware en el caso de encontrarse este presente en un dispositivo.

**METODOLOGÍA**

El trabajo y la gestión del proyecto se llevarán a cabo siguiendo de manera aproximada la metodología ágil (también conocida como agile), ya que, pese a que existe una clara demarcación entre algunas de las fases debido a su diferente naturaleza, no se trata de unidades aisladas. Los datos obtenidos en cada una de las etapas serán de ayuda en otras fases, permitiendo ampliar de manera continua el conocimiento acerca del funcionamiento del malware. El análisis estático y dinámico se complementarán entre sí, aportando información que permitirá progresar globalmente. No obstante, es importante que no se ha seguido completamente la modalidad con sprints claramente definidos y entregables al final de cada uno como se indica en algunas variantes de agile como puede ser Scrum. Las razones de esto son, por un lado, la dificultad de concretar plazos para ciertas tareas al tratarse de un proyecto que cuenta con un componente importante de investigación, y por lo tanto de varias incógnitas que impiden estimar tiempos con una alta precisión y, por otro lado, debido a la complejidad de compatibilizar el trabajo en el proyecto de manera totalmente definida con el horario laboral del alumno. Sin embargo, se ha tratado en la medida de lo posible de seguir el modelo y su filosofía de adaptación a cambios e iteración frecuente.

Tras la planificación inicial, las etapas del proyecto quedan definidas de la siguiente manera:

1. Diseño de los análisis y planificación de las fases que los compondrán
2. Realización de los análisis y pruebas.
3. Documentación de los resultados obtenidos.

**ESTRUCTURA DEL TRABAJO**

Esta memoria tiene una estructura dividida en capítulos, y cuenta en su parte final con anexos relativos a temas generales del proyecto. A continuación, se muestra un listado de cada uno de ellos, así como una descripción detallada de los contenidos que componen cada ítem:

* Introducción. Planteamiento del problema a resolver, resumen de objetivos, metodología de trabajo y presentación de la estructura de la memoria.
* Capítulo 1. Investigación previa realizada, estado del arte en la temática del proyecto y el proceso de selección del malware a analizar.
* Capítulo 2. Planificación temporal creada inicialmente, junto con presupuestos, la gestión de riesgos y la materia de legislación aplicable al proyecto.
* Capítulo 3. Organización de los análisis a realizar, siendo estos el análisis estático y el dinámico. Presentación del método de trabajo a utilizar durante la realización de las actividades y posibles problemas a encontrar, así como soluciones o mitigaciones previstas.
* Capítulo 4. Presentación de los resultados obtenidos a raíz de los análisis aplicados. Comparativa entre los resultados de las dos fases y corroboración de las coincidencias observadas, además de las diferencias que puedan producirse.
* Capítulo 5. Aportaciones realizadas a través de los resultados obtenidos, al igual que comentario y contextualización de estos. En otro orden de cosas, también se hará una enumeración de los problemas encontrados durante la realización del proyecto, así como posibles futuras mejoras o ampliaciones a aplicar.
* Anexo A. Documentación del progreso realizado y materiales generados, control de versiones del trabajo.
* Anexo B. Seguimiento del proyecto y cambios efectuados respecto a las planificaciones iniciales.

**Capítulo 1: Estudio del problema**

## 1.1 EL CONTEXTO DEL PROBLEMA

En este proyecto se ha llevado a cabo un análisis de una muestra de malware Android de tipo troyano perteneciente a la familia Chameleon. Se han empleado aproximadamente 3 meses en llevar a cabo la tarea incluyendo las investigaciones iniciales, ya que, a pesar de ser un tema cercano a las temáticas del máster, no se cuentan con conocimientos extensos en profundidad para la realización del análisis, especialmente para plataformas móviles como Android. No obstante, algunas asignaturas del título han servido para sentar las bases y permitir avanzar más rápidamente, así como para familiarizarse con algunas de las herramientas a utilizar.

Para llevar a cabo un análisis en profundidad que identifique los vectores de ataque del malware, así como sus procedimientos y objetivos, se han de realizar los siguientes pasos:

* Investigación de la literatura existente respecto al tipo de malware elegido y las familias relacionadas, con el fin de conocer su naturaleza y los procedimientos más habituales que utilizan.
* Análisis estático de la muestra obtenida, que permita entender el funcionamiento interno del malware y las tareas que puede realizar, así como código que pese a no ser necesariamente ejecutado en las pruebas dinámicas también forma parte de la amenaza, pudiendo ser ejecutado bajo ciertas condiciones especiales.
* Análisis dinámico del malware en un entorno controlado, que posibilite visualizar y obtener datos acerca del funcionamiento del troyano en un dispositivo Android, que en este caso será una versión virtualizada del sistema.
* Análisis de los resultados obtenidos en las fases anteriores, determinando las amenazas que presenta el malware para los sistemas Android, por qué tiene éxito en sus ataques y cuáles son los signos o indicadores de su presencia en un dispositivo.

## 1.2 EL ESTADO DEL ARTE

En esta sección se hará una introducción al malware desde la perspectiva de Android resultado de la evaluación y revisión de literatura realizada, dividiéndose en cinco subsecciones según la temática, comenzando por una visión general del malware para la plataforma.

**Malware en Android**

La plataforma Android surgió en 2008 con la llegada al mercado de los primeros dispositivos móviles inteligentes (conocidos popularmente como smartphones). Está basada en el kernel Linux y es un proyecto de código abierto liderado por Google, que cuenta hasta el momento con 21 versiones diferentes y una gran fragmentación en lo relativo a estas [X]. Este es uno de los puntos a destacar desde el punto de vista de la seguridad, ya que esta división con dispositivos que cuentan con versiones significantemente diferentes provoca que existan distintos niveles de protección y mitigaciones en función de la fecha de publicación de la versión concreta de Android, así como de las posteriores actualizaciones de seguridad. Asimismo, existen diferentes variantes de Android dentro de las propias versiones, ya que muchos fabricantes optan por personalizar Android para sus dispositivos con el objetivo de ofrecer características adicionales a los usuarios de sus dispositivos u optimizar el sistema operativo para aprovechar de manera más eficiente los recursos de los que estos disponen. Pese a que el sistema de Google es el más popular, grandes fabricantes como Samsung, Xiaomi o Huawei con su propio sistema Android, lo que aumenta aún más la fragmentación y también supone en algunos casos cambios en las políticas y medidas de seguridad aplicadas.

Hoy en día el sistema Android en sus diferentes versiones engloba a la mayoría de los dispositivos móviles en el mercado, ya que cuenta con una cuota superior al 70%. Este dato, aunado a la fragmentación previamente mencionada, hace de la plataforma Android el objetivo número uno de los atacantes en lo concerniente a sistemas móviles, con iOS en la segunda posición a gran distancia en términos de volumen de ataques.

Pese a la relativa irrupción reciente en el mundo de la informática, Android ha sido un blanco de malware desde sus inicios. En 2010 surgió el primer malware específico para Android, aunque ya existía malware para sistemas operativos móviles desde 2004 [X] y desde entonces el número de amenazas en Android ha crecido rápidamente en comparación con otras plataformas, aumentando principalmente en volumen, pero también en sofisticación. Los fabricantes tratan de incluir nuevas medidas de prevención y reducción de riesgos, así como realizar campañas de prevención, algunas de ellas en colaboración con organismos públicos.

No obstante, el malware en Android sigue siendo una actividad altamente lucrativa, no solamente en términos económicos, si no también geopolíticos, como ha sido demostrado por algunos de los últimos ataques de APTs, en los que Android ha sido la plataforma y la finalidad es la exfiltración de datos de carácter confidencial o sensible.

**Troyanos en Android**

El malware de tipo troyano es uno de los más populares en términos globales sin tener en cuenta la plataforma, y el sistema Android sigue el patrón. Se estima que entorno a un 30% del malware de Android se puede encuadrar dentro de esta tipología.

Dentro de las muestras categorizadas como troyanos se pueden encontrar dos modalidades cuando hablamos de Android. La primera de ellas se trata de una aplicación que simula ser legítima, y siendo en esencia una versión modificada de la aplicación a la que trata de suplantar. En esta versión alterada se encuentra el código malicioso que permite al troyano la realización de sus acciones maliciosas. La tasa de éxito de esta tipología de troyano viene dada por su capacidad para emular las características de la aplicación suplantada, tratando de ser una copia casi exacta y en muchos casos contando con gran parte del código de la aplicación original. Para tratar de aumentar las posibilidades de infección, los atacantes suelen elegir aplicaciones populares y conocidas para suplantar.

Por otro lado, existen troyanos dentro de aplicaciones que simulan ser benignas sin suplantar a otras existentes, como pueden ser utilidades de sistema o plataformas de juegos que cuentan con funcionalidades legítimas en algunos casos, pero que siempre contienen componentes maliciosos.

Al basarse Android en un sistema aplicaciones y siendo gran parte de ellas ajenas al sistema original que se instala en el dispositivo, los troyanos se adaptan perfectamente a este modelo y son una de las tipologías de malware más eficaces y con mayor capacidad lucrativa para los grupos de atacantes.

**Métodos de distribución**

Uno de los factores importantes a tener en cuenta en los troyanos en Android es el canal de distribución. Los atacantes tratan de minimizar el nivel de sospecha por parte de los usuarios para facilitar la instalación de las aplicaciones maliciosas, y para ello utilizan diferentes mecanismos.

El primero de ellos y uno que aumenta la credibilidad del troyano es su distribución directa mediante canales oficiales de aplicaciones. El principal de estos canales es la Play Store de Google, que cuenta en la actualidad con más de 3 millones de aplicaciones para el sistema Android. En el sistema Android por defecto, esta es la única tienda de aplicaciones disponible, aunque algunos fabricantes utilizan también plataformas propias para la distribución de aplicaciones, como es el caso de Samsung con su Galaxy Store o la Appstore de los dispositivos Amazon.

Además, existen canales de distribución alternativos no asociados a los fabricantes, pero que también cuentan con un gran volumen de usuarios debido a las características particulares que ofrecen, como por ejemplo F-Droid [X] , que distribuye únicamente aplicaciones de código abierto o Aptoide [X] , un mercado secundario popular debido a que almacena y permite la descarga de aplicaciones que originalmente son de pago de manera gratuita, y también facilita la descarga de versiones específicas de aplicaciones que ya no están disponibles a través de los canales oficiales debido a su antigüedad.

Asimismo, una de las opciones del sistema Android es la instalación de aplicaciones directamente por parte de los usuarios si estos cuentan con el archivo APK y lo transfieren al dispositivo. Pese a que por defecto la seguridad del sistema previene la instalación de aplicaciones que provienen de fuentes desconocidas, esta prevención puede ser desactivada por los usuarios de manera sencilla en los ajustes del sistema. Esto supone un vector adicional de entrada de aplicaciones al sistema, y en este caso se cuenta con un nivel de protecciones bajo relativo a las otras opciones de distribución mencionadas previamente.

Por defecto, la tienda Play Store de Google y los mercados de los fabricantes mayoritarios cuentan con comprobaciones y análisis que son efectuados en las aplicaciones previa a su publicación en el catálogo del mercado. Esto minimiza la cantidad de aplicaciones maliciosas que logran la publicación, pero sigue siendo una cantidad importante de ellas la que logra pasar las barreras y establecerse en los mercados oficiales, con millones de descargas por parte de usuarios [X].

Además, Google cuenta con funcionalidades de seguridad adicionales en el caso de que la aplicación ya se encuentre instalada en el sistema. Con el objetivo de identificar PHAs (término de Google para englobar aplicaciones que pueden provocar daños al dispositivo o sus datos, que proviene del inglés Potentially Harmful Applications [X]) Google ha desarrollado el sistema Google Play Protect, que realiza escaneos diarios de todas las aplicaciones presentes en el dispositivo, además de intentar prevenir instalaciones de aplicaciones externas a Google Play si estas son detectadas como maliciosas en función de las muestras dañinas de las que poseen información previa. Como medidas adicionales, este servicio Play Protect permite a los usuarios la petición de escaneos suplementarios, y también tiene un modo de funcionamiento que permite los escaneos cuando el dispositivo carece de conexión a Internet.

En definitiva, Google (y en menor medida otros de los principales fabricantes) cuentan con numerosos sistemas de prevención y defensa para tratar de combatir a las aplicaciones maliciosas. No obstante, esto no impide que sigan apareciendo casos en los que los atacantes son capaces de sobrepasarlas y cumplir sus objetivos maliciosos, bien a través de canales oficiales, bien mediante el tráfico de APKs en mercados secundarios. Además, las medidas de seguridad no cubren todos los casos, ya que algunas funcionalidades, como es el caso de los permisos de accesibilidad que se verán en detalle a continuación, quedan fuera de las pruebas y comprobaciones de seguridad realizadas.

**Sistema de permisos de accesibilidad**

El abuso de los permisos es uno de los vectores de ataque más comunes por parte de aplicaciones Android, contando muchas veces con la colaboración del usuario, que desconoce qué permisos está realmente otorgando o no presta demasiada atención al proceso inicial al instalar una aplicación. Debido a esto, Android cuenta desde su versión 12 con medidas adicionales de notificación para tratar de alertar a los usuarios en casos como la activación de la cámara o el micrófono del dispositivo, conocidas como “Privacy indicators” [X]. Por esto, algunos de los desarrolladores de aplicaciones maliciosas han modificado sus mecanismos de acceso a estos recursos, haciendo uso de otro tipo de permisos que otorga el sistema con la esperanza de que no alerten al usuario de manera tan repentina y este acepte los permisos sin levantar sospechas.

Algunos de estos permisos son relativos a la accesibilidad, la herramienta de los sistemas operativos móviles (y sistemas informáticos en general) con la finalidad de facilitar y hacer lo más práctica posible la experiencia de uso para las personas con discapacidades visuales, auditivas, cognitivas o físicas. En el caso de Android, las primeras opciones de accesibilidad fueron introducidas en su versión 1.6 (conocida popularmente como Donut) y se centraron en el lector de pantalla TalkBack, con nuevas características surgiendo en versiones posteriores del sistema, como la navegación por gestos, el control del dispositivo por voz mediante el sistema Voice Access, la generación automática de subtítulos o la conexión con audífonos o implantes cocleares, además de contar con Android Accesibility Suite [X], que reúne la gestión de varios de estos servicios en una única aplicación oficial de Google.

Otro de los mecanismos que los atacantes utilizan en algunos casos es el sistema de administración de dispositivo. Este sistema es una API que Android proporciona desde su versión 2.2 (conocida como Froyo) [X] y que permite a los desarrolladores de aplicaciones manejar a través de estas la gestión de ciertos servicios del dispositivo que por lo general se encuentran fuera del sistema habitual de permisos. Esta funcionalidad está diseñada principalmente para entornos de empresa en los que se busca facilitar la gestión de numerosos dispositivos empresariales de manera controlada y segura, permitiendo la instalación de aplicaciones y el control sobre las políticas de bloqueo de pantalla entre otros. La lista de permisos que otorga este rol de administrador a través de la API varía en función de los requeridos por la aplicación que cuenta con el rol. Por lo tanto, pese a que esta característica puede resultar útil en ciertos casos, también puede ser abusada para tratar de obtener mayores privilegios bajo el pretexto del control seguro del dispositivo. No obstante, para que una aplicación sea asignada como administradora del dispositivo, el usuario ha de aceptar los términos, por lo tanto, nuevamente es la falta de atención o el desconocimiento de los usuarios que abre la puerta a la entrada del malware.

**Análisis de malware en Android**

Pese a que en sus inicios el malware para sistemas móviles era rudimentario en comparación con el existente en los sistemas informáticos más maduros como los ordenadores, con el tiempo los primeros han ido aumentando su nivel de sofisticación.

En muchos de los casos, el análisis de malware para los sistemas móviles, y particularmente Android al ser la plataforma más popular, está basado en métodos y técnicas similares provenientes del análisis de malware tradicional, al ser estrategias que han demostrado ser útiles en la identificación y prevención de software dañino. En ambas plataformas, las principales técnicas son el análisis estático y el análisis dinámico. En primer lugar, hablaremos del análisis estático aplicado a Android.

Pese a utilizar técnicas similares al análisis tradicional de malware, para el análisis estático en Android las herramientas comunes para plataformas como Windows o Linux, como pueden ser los analizadores de cabeceras o ficheros, no son normalmente de gran utilidad. Además, el código fuente del malware rara vez se encuentra disponible, teniendo que ser analizado en su lugar el bytecode que se encuentra dentro de los ficheros APK de las aplicaciones maliciosas, y que en algunos casos también contará con medidas de protección similares a las encontradas en sistemas como Windows, como pueden ser el packing [X] de aplicaciones para la ocultación de sus clases reales y su posterior carga dinámica [X] para evadir la detección estática de manera sencilla, o la ofuscación del propio código para hacerlo más difícil de comprender en el futuro por parte de posibles analistas [X]. Para lidiar con estos casos, se cuenta con una variedad de herramientas específicas para para Android, como pueden ser simplify [X] para tratar de revertir algunos de los métodos de ofuscación más comunes y populares, como pueden ser los aplicados por el programa ProGuard [X], o el unpacker HoseDex2Jar [X]. Una vez obtenido el fichero DEX definitivo que declara las clases que utilizará la aplicación, se puede tratar de obtener código Java a partir de él, haciendo uso de herramientas populares como JADX [X].

No obstante, en el caso de ser superada la fase de medidas de anti-análisis comentadas, y estando en posesión de bien código Java o tratarse de un caso de malware que utiliza librerías nativas compiladas a partir de código C o C++, algunas herramientas que forman parte del arsenal de análisis estático estándar son de gran utilidad, como puede ser Ghidra [X], además de técnicas establecidas en el análisis tradicional como el análisis de dependencias mediante grafos.

En lo relativo al análisis dinámico, Android cuenta con menos opciones para la virtualización de dispositivos que las disponibles para plataformas más establecidas en el mundo del malware como son Windows o Linux. Se distinguen dos tipos de análisis en este caso, conocidos como in-the-box y out-the-box. El primero de ellos hace referencia al análisis en el que se ejecuta la muestra maliciosa y el análisis junto con la recogida de información se produce en el mismo nivel y en el mismo dispositivo. Para llevar a cabo este método se necesita de utilidades que trabajen en el mismo nivel que la máquina virtual Dalvik y tratan de instrumentar procesos internos de Android para obtener información acerca de las modificaciones realizadas por el malware, como por ejemplo es el caso de la herramienta DIVILAR [X]. Un punto en contra de este tipo de métodos que trabajan en un nivel de abstracción igual que el propio malware, es que este último puede realizar comprobaciones para tratar de detectar si el entorno está siendo alterado con fines analíticos y modificar su comportamiento para tratar de pasar desapercibido, o simplemente dejar de funcionar [X].

El segundo método, out-the-box, también conocido como análisis de tipo VM, se trata del uso de emuladores y entornos virtuales aislados (también denominados sandbox) como pueden ser el propio emulador de la suite Android Studio de Google o herramientas de terceros como GenyMotion [X] o Android-x86 [X], realizando el análisis desde un entorno externo en la misma máquina. Pese a que las diferencias entre la emulación y un entorno real son casi imperceptibles, incluyendo la simulación de elementos como el sensor GPS, la cámara, o el lector de huellas dactilares entre otros, existen diferencias y nuevamente el malware puede detectar que está siendo ejecutado en un entorno modificado, respondiendo adecuadamente para tratar de evitar el análisis. Pese a que este segundo método no trabaja en un nivel tan cercano al propio malware, permite la utilización de multitud de herramientas para tratar de instrumentar la ejecución de este, como pueden ser FRIDA [X], herramientas derivadas como objection [X] o medusa [X], además de utilidades como Wireshark para la captura de tráfico o la suite de Burp para realizar operaciones de proxy como el Certificate Pinning [X].

Es importante mencionar que las herramientas de análisis dinámico enumeradas hasta el momento son todas ellas aplicables a entornos locales, y se ha de tener en cuenta que hay también un ecosistema de ofertas en la nube como por ejemplo JoeSandbox [X], que permiten la asignación de espacios aislados en los que ejecutar aplicaciones Android potencialmente maliciosas y recabar información acerca de su comportamiento, que pese a no ofrecer el control ni la granularidad de análisis que se pueden obtener con métodos locale, pueden ser una solución adecuada para un análisis de menor profundidad pero que proporcione datos importantes sobre el malware. No obstante, en este espacio de soluciones online el ecosistema Android carece del número de plataformas dedicadas a otros sistemas como Windows o Linux.

Por último, además de los dos métodos principales de análisis, también se puede considerar un tercero conocido como análisis híbrido, que aúna el análisis estático y el dinámico, ejecutándolos de manera cercana al paralelismo para tratar de explorar todos los caminos de ejecución que puede ofrecer una aplicación y tratar de maximizar el área del código del malware cubierta por el análisis. Herramientas como el módulo A5 [X] o el algoritmo de tipo concólico (que aúna análisis concreto y simbólico) ALLSEQS [X] son algunos ejemplos de este tipo de técnicas.

### 1.2.5 Conclusión

Como se ha expuesto en la sección anterior, el estado actual tanto del malware en Android, específicamente de las aplicaciones de tipo troyano, como también de las herramientas y técnicas que existen para su análisis son ampliamente variadas.

Pese a que el entorno móvil no cuenta todavía con el mismo abanico de posibilidades y recursos que los entornos mayoritarios de escritorio como Windows o Linux, tanto la sofisticación del malware como los procedimientos para su detección se encuentran en el centro de una carrera constante de innovación entre atacantes y analistas del lado de la defensa que hace que el análisis de malware en Android sea un tema de actualidad y gran importancia dentro del mundo de la ciberseguridad.

## 1.3 DEFINICIÓN DEL PROBLEMA

Pese a las medidas de seguridad existentes en Android, una parte importante del malware sigue teniendo éxito en su cometido, con la combinación de distintos tipos de ataques y mecanismos para tratar de obtener privilegios adicionales. En muchos casos el desconocimiento de los propios usuarios y la falta de comprobaciones adicionales provoca las infecciones y los consiguientes comportamientos maliciosos por parte de las aplicaciones. El abuso de los permisos es uno de los vectores que puede utilizar el malware en Android, particularmente aquellos de tipo troyano.

Es importante recabar información para conocer su funcionamiento en detalle, desde la llegada de la aplicación al dispositivo previa a su instalación hasta sus técnicas para conseguir tomar el control de los dispositivos y las acciones que realiza. De toda esta información que se ha de recabar se puede extraer un perfil que identifique al malware bajo análisis, y que permita crear una serie de IoCs que permitan su reconocimiento preventivo de manera automática, la obtención de información acerca de los grupos de atacantes que pueden estar tras su creación y explotación en el caso de que esto sea posible, además de poder utilizar los datos sobre el proceso del malware para petición de permisos y los pasos iniciales que efectúa para así utilizarlos en la realización de campañas de advertencia con finalidad preventiva destinadas a los usuarios.

## 1.4 MALWARE ELEGIDO

Se ha elegido una muestra de la familia de troyanos conocida como Chameleon. Pese a tener el mismo nombre que una botnet surgida en 2013 [X], la familia en la que se centrará el proyecto tiene únicamente como objetivo la infección de dispositivos Android. Chameleon apareció online por primera vez a principios de 2023 [X], y desde entonces ha pasado por diferentes iteraciones, introduciendo nuevos mecanismos para obtener sus objetivos y evadir la detección por los usuarios. Las razones detrás de esta elección son principalmente dos. En primer lugar, se trata de un malware reciente y que se encuentra actualmente en el foco de la prensa generalista debido a la gran cantidad de amenazas que presenta pese a ser un único malware [X]. Asimismo, esta nueva versión es diferente a las anteriores, y también a otros troyanos para sistemas Android, en la manera en que gestiona los parámetros de accesibilidad para hacerse con el control del dispositivo y evitar la protección biométrica para su desbloqueo [X]. Además, pese a ser un malware relativamente popular en estos últimos años, y aunque como se ha mencionado, sí que existen multitud de artículos de carácter general en medios de comunicación reconocidos haciendo referencia a la amenaza que presenta para los usuarios, no se han publicado artículos especializados ni análisis detallados del malware, siendo el único análisis encontrado el desarrollado por la empresa estadounidense de ciberseguridad Cyble en abril de 2023 para una de las nuevas versiones del troyano [X]. Por todas estas razones expuestas se ha decidido finalmente seleccionar esta familia de malware como temática del proyecto, y más concretamente la versión de Chameleon que se hace pasar por la aplicación Google Chrome y que posee el valor hash SHA-256 de 0a6ffd4163cd96d7d262be5ae7fa5cfc3affbea822d122c0803379d78431e5f6. La muestra ha sido obtenida del popular repositorio de malware y publicaciones de ciberseguridad Vx Underground [X].

### 1.4.2 Tecnologías utilizadas

A continuación, se incluyen las tecnologías utilizadas a lo largo de la realización del proyecto, formando parte de ambas fases de análisis del malware. En primer lugar, se comentan las herramientas utilizadas para realizar el análisis estático del código.



* JADX: se trata de una de las herramientas más populares para descompilar código DEX proveniente de un archivo APK Android a código Java, que es significantemente más fácil de comprender. Además, cuenta con una versión con interfaz gráfica de usuario que permite la visualización de proyectos completos, así como el renombrado de clases, métodos y código junto con el añadido de comentarios, siendo de especial utilidad en la reestructuración y anotación de código ofuscado.



* Apktool: herramienta versátil que sirve para desempaquetar archivos APK y obtener fácilmente a través de la línea de comandos todos los ficheros que componen la aplicación, además de permitir realizar tareas adicionales como el reempaquetado una vez se han aplicado modificaciones en el código.



* Apkid: utilidad para la identificación de packers, protectores y ofuscadores en el proceso de creación de una aplicación. Si algún método popular se ha utilizado para la realización de estas acciones de evasión, Apkid informa de ello, además de proporcionar detalles respecto a la versión particular de la herramienta utilizada.



* Sublime Text: editor de texto ligero que soporta el procesamiento y la visualización de una gran variedad de formatos. Utilizado principalmente en el proyecto para la toma de notas de carácter misceláneo con el fin de guiar los procesos de análisis y permitir registrar de manera rápida eventos o detalles que podrían ser de utilidad en el futuro.



* Android Studio: suite de desarrollo de aplicaciones Android proporcionada por Google. Además de servir para crear aplicaciones, también sirve para la depuración y el análisis de archivos APK. Cuenta con un sistema de emulación de dispositivos Android que también hace que esta herramienta sea utilizada intensivamente en la fase de análisis dinámico. Al ser una herramienta de Google, está integrada totalmente en el ecosistema Android, y por lo tanto ofrece un gran abanico de opciones a la hora de emular dispositivos Android, permitiendo personalizar parámetros como ubicación, simulación de envío de mensajes SMS o simulación de ubicaciones GPS.



* IntelliJ IDEA: entorno de desarrollo integrado orientado al trabajo con código Java y Kotlin. Es uno de los IDEs más populares debido a su gran número de opciones y rendimiento. En este proyecto se utiliza para la creación de programas auxiliares en Java que sirvan para tareas variadas del análisis, como la desencriptación de valores o la instrumentación de código.

En lo relativo a la sección de análisis dinámico del troyano, las herramientas utilizadas fueron las siguientes:

* Android Studio: como se ha mencionado en la sección anterior, el emulador de Android Studio es una pieza muy importante en el análisis dinámico, ya que pese a que contiene algunas limitaciones respecto a otros como la falta de acceso superusuario.



* Adb: parte de las herramientas de depuración de Android, permite la conexión de dispositivos Android (físicos o emulados) a los entornos de depuración elegidos. En este caso será utilizado para la conexión a los dispositivos emulados, con la finalidad de descargar y enviar archivos a estos, así como poder conectar consolas para realizar algunas fases del análisis como el Certificate Pinning.
* Genymotion:
* FRIDA
* Burp Suite
* DB Browser

Por último, se presentan herramientas adicionales utilizadas durante el proyecto sin pertenecer a ninguna de las fases en particular, si no que han sido empleadas para tareas diversas durante todas las fases:

* Brew: gestor de paquetes utilizado en el sistema Mac para la instalación de programas y utilidades.
* Pip: gestor de paquetes y dependencias Python.
* HexFiend: visualización del dump de archivos en hexadecimal para tratar de averiguar el tipo de fichero dada su firma.
* Microsoft Word: realización de la memoria.

# Capítulo 2: Gestión del proyecto

En este apartado se especifica cómo se ha realizado la gestión del proyecto, incluyendo la planificación temporal del mismo, los presupuestos de su realización, la gestión de riesgos existentes, así como el empleo de la legislación aplicable correspondiente.

## 2.1 ALCANCE DEL PROYECTO

## 2.2 PLAN DE TRABAJO

Para llevar a cabo la planificación de este proyecto se han estimado una serie de tareas a realizar y una duración aproximada del tiempo que se podría tardar en ejecutarlas. Más adelante, en el [Anexo B](#_B.1__), se mostrará la duración real que han tenido finalmente dichas tareas.

### 2.2.1 Identificación de tareas

Las tareas principales que se han definido para conseguir llegar a la solución propuesta son las siguientes:

* Tarea 1: realizar un estudio de la literatura existente de malwares troyanos en Android para tener una perspectiva general del estado actual de amenazas en la plataforma.
* Tarea 2: elegir una familia concreta de malware y obtener muestras para su posterior análisis, además de estudiar sus medios de distribución habituales.
* Tarea 3: realizar un análisis estático de las muestras del malware, con la finalidad de entender el código que lo compone y las tareas que realiza en la medida de lo posible. Sin requerir el descifrado completo de su código en el caso de estar este encriptado u ofuscado, será importante comprender una cantidad suficiente como para discernir el funcionamiento general del malware y las acciones (maliciosas o no) que realiza en el dispositivo.
* Tarea 4: realizar un análisis dinámico de las muestras, recabando los IoCs que puedan registrarse y observando el comportamiento del malware, comprobando paralelamente que los resultados obtenidos en el análisis estático se corresponden con los patrones observados en el dispositivo infectado.

### 2.2.2 Estimación de tareas

Para la elaboración de este proyecto se estima una duración total de unas 250 horas repartidas a lo largo de 3 meses, lo que equivaldría a unas 2,7 horas de trabajo diario.

### 2.2.3 Planificación de tareas

## 2.3 GESTIÓN DE RECURSOS

### 2.3.1 Especificación de recursos

* **Recursos materiales**
* **Recursos humanos**

### 2.3.2 Presupuesto

**Coste de recursos materiales**

**Coste de recursos humanos**

**Beneficio industrial**

**Coste total**

## 2.4 GESTIÓN DE RIESGOS

### 2.4.1 Identificación de riesgos

### 2.4.2 Análisis de riesgos

## 2.5 LEGISLACIÓN Y NORMATIVA

**Capítulo 3: Solución**

## 3.1 DESCRIPCIÓN DE LA SOLUCIÓN

## 3.2 EL PROCESO DE DESARROLLO

### 3.2.1 Análisis estático

### 3.2.1.1 Especificación de requisitos

### 3.2.1.2 Consideraciones adicionales

### 3.2.2 Análisis dinámico

### 3.2.2.1 Especificación de requisitos

### 3.2.2.2 Consideraciones adicionales

## 3.3 RESULTADOS DEL DESARROLLO

**Capítulo 4: Evaluación**

**Capítulo 5: Conclusión**

En este apartado se comentan los resultados obtenidos previamente, las aportaciones que se han realizado, se mencionan posibles ampliaciones que se podrían realizar en el futuro, se mencionan los problemas encontrados durante la realización de la tesis y, por último, se encuentra una sección de opiniones en la que se emitirán algunas valoraciones a título personal tras la realización del proyecto.

## 5.1 APORTACIONES REALIZADAS

## 5.2 AMPLIACIONES FUTURAS

## 5.3 PROBLEMAS ENCONTRADOS

## 5.4 OPINIONES PERSONALES

**Bibliografía**

|  |  |
| --- | --- |
| [1] | <https://www.researchgate.net/publication/312129902_Android_Fragmentation_Classification_Causes_Problems_and_Solutions> |
| [2] | <https://www.cs.tufts.edu/comp/116/archive/fall2017/tzhu.pdf>) |
| [3]  [4]  [5]  [6] | <https://f-droid.org/es/>  <https://www.kaspersky.com/blog/malware-in-google-play-2023/49579/>  <https://developers.google.com/android/play-protect/potentially-harmful-applications>  <https://source.android.com/docs/core/permissions/privacy-indicators>) |
| [7]  [8]  [9]  [10]  [11]  [12]  [13]  [14]  [15]  [16]  [17]  [18]  [19]  [20]  [21]  [22]  [23]  [24]  [25]  [26]  [27]  [28]  [29]  [30] | <https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=en>  <https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/admin/device-admin.html>  <https://cyble.com/blog/chameleon-a-new-android-malware-spotted-in-the-wild/>  <https://www.bbc.com/news/technology-21860360>  <https://www.europapress.es/portaltic/ciberseguridad/noticia-malware-chameleon-bloquea-huella-dactilar-forzar-introduccion-pin-hacerse-cuentas-bancarias-20231229135249.html>  <https://www.xatakandroid.com/seguridad/asi-logra-saltarse-biometria-movil-peligroso-troyano-bancario-chameleon>  <https://newsukraine.rbc.ua/news/android-virus-chameleon-banking-learns-to-1703623824.html>  <https://www.ndss-symposium.org/wp-content/uploads/2018/03/NDSS2018_04A-4_Duan_Slides.pdf>  <https://www.researchgate.net/publication/308180878_Structural_analysis_of_packing_schemes_for_extracting_hidden_codes_in_mobile_malware>  <https://developer.android.com/build/shrink-code#obfuscate>  <https://github.com/CalebFenton/simplify>  <https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20presentations/DEF%20CON%2025%20-%20Slava-Makkaveev-and-Avi-Bashan-Unboxing-Android.pdf>  <https://github.com/Guardsquare/proguard>  <https://media.defcon.org/DEF%20CON%2022/DEF%20CON%2022%20presentations/DEF%20CON%2022%20-%20Strazzere-and-Sawyer-Android-Hacker-Protection-Level-UPDATED.pdf>  <https://github.com/skylot/jadx>  <https://www.ghidra-sre.org/>  <https://www.researchgate.net/publication/262412421_DIVILAR_Diversifying_intermediate_language_for_anti-repackaging_on_Android_platform>  <https://kclpure.kcl.ac.uk/ws/portalfiles/portal/112357801/The_Evolution_of_Android_Malware_TAM_2017_GREEN_AAM.pdf>  <https://developer.android.com/studio/run/emulator>  <https://www.genymotion.com/>  <https://www.android-x86.org/>  <https://users.ece.cmu.edu/~tvidas/papers/SPSM14.pdf>  <https://www.cis.upenn.edu/~mhnaik/papers/fse12_full.pdf> |
|  |  |
|  |  |
|  |  |
|  |  |

**Anexo A: Control de versiones**

Se ha creado un [repositorio público en GitHub](https://github.com/raulng9/TFM_MalwareAnalysis) con los materiales utilizados en la realización del proyecto, incluyendo documentos con recursos utilizados, resultados derivados de los análisis, muestras de malware utilizadas y notas acerca del proceso.

**Anexo B: Seguimiento del proyecto**

## B.1 FORMA DE SEGUIMIENTO

A lo largo de todo el proceso se han realizado reuniones periódicas con el tutor para monitorizar el progreso llevado a cabo, así como dar recomendaciones y correcciones al alumno, además de tratar las próximas etapas a realizar.

## B.2 PLANIFICACIÓN INICIAL

## B.3 PLANIFICACIÓN FINAL