Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin De Grado

Grado en Ingeniería Informática
Mención en tecnologías de la información

Creación de una botnet para dispositivos móviles

Autor:
D. Carlos Macías Valdivieso

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin De Grado

Grado en Ingeniería Informática
Mención en tecnologías de la información

Creación de una botnet para dispositivos móviles

Autor:
D. Carlos Macías Valdivieso

Tutor:
D. Blas Torregrosa García

Resumen

En el presente proyecto se diseña, desarrolla y despliega una botnet para dispositivos móviles con fines académicos. El objetivo es que haya un conjunto de bots móviles que se comuniquen anónimamente con el botmaster (administrador de la botnet), del que recibirán las instrucciones que deben ejecutar. Además, el botmaster podrá obtener información confidencial de los bots, como por ejemplo conversaciones de chats, credenciales... También podrá monitorizar las acciones que realiza el usuario sobre el terminal infectado. Añadido a esto, el bot deberá implementar persistencia, de forma que si se reinicia, este debe seguir consultando y ejecutando las instrucciones ordenadas por el botmaster.

Para ello, se estudian las diferentes técnicas que utilizan los malware en sistemas móviles para implementar persistencia, comando y control y exfiltración de información. También se prepara un entorno de laboratorio controlado sobre el que se realizarán las pruebas de funcionamiento de la botnet. La botnet consta de: los clientes (que son los bots), el servidor C&C y la base de datos realtime de Firebase. Esta base de datos hace de intermediaria entre los bots y el servidor C&C, añadiendo una capa de anonimización.

Para que la interacción del botmaster con la botnet sea lo más intuitiva posible, se implementa en el servidor C&C una aplicación web. Por una parte el bot está desarrollado en lenguaje Java para la lógica de la aplicación y XML para la interfaz. Por otra parte, el servidor C&C está codificado en Python, utilizando el framework de Flask, en conjunto con otras librerías utilizadas para el front-end de la aplicación. Esta aplicación Python se ha desplegado sobre un contenedor Docker.

Abstract

This project designs, develops and deploys a botnet for mobile devices as well as for academic purposes. The aim of this project is to accomplish a group of mobile bots which communicates with the botmaster (the botnet's administrator), from which they will receive the instructions to execute. The botmaster will also be able to obtain some confidential information from the bots; for exemple: chats conversations, credentials... The botmaster will also be able to monitor the user's actions that it gets in the infected terminal. Apart from that, the bot should implement persistence so that, if the bot reboots by itself, it will keep receiving and executing instructions.

For this purpose, we study the different techniques that the malware uses in mobile systems to implement persistence, comand, control and information exfiltration. We also prepare a controlled laboratory environment, in which we will test the botnet. The composition of the botnet is: the customers (the bots), the C&C server and the realtime database of Firebase. This database acts as intemediary between the bots and the C&C server, adding an anonymization layer.

Furthermore, so that the botnet could be as intuitive as possible, we will implement in the C&C server a web application. On the one hand, the bot is developed in Java language for the logic of the application, and XML for the interface. On the other hand, the C&C server is encrypted in Python, using the Flask framework in conjunction with other libraries for the application front-end. The Python application have been deployed on a Docker container.

Índice general

Capítulo 1

Introducción

1.1. Motivación

En los últimos años el proceso de digitalización se ha acelerado. La pandemia también ha ayudado a esto. Es por ello que el volumen de información almacenada digitalmente cada vez es mayor. Por consiguiente, se deben de tomar las medidas necesarias para garantizar la confidencialidad, disponibilidad e integridad de la información. Además, es importante prestar atención a los sistemas móviles y los dispositivos IoT, los cuales han aumentado exponencialmente en la última década [1]. En consecuencia, el número de ataques a este tipo de dispositivos a aumentado paralelamente. También se han desarrollado distintos tipos de malware específicos para este tipo de dispositivos como las botnets, las APTs u otros. En este proyecto, se desarrollará un malware completamente funcional con el que se creará una botnet. De esta forma se podrán analizar las distintas técnicas que se utilizan en esta tipología de malware y así poder prevenirlo de una manera más eficaz.

1.2. Objetivos

Los objetivos del presente trabajo de fin de grado son principalmente el diseño y desarrollo de un malware con fines académicos que origine una botnet cuyos bots sean dispositivos móviles con sistema operativo Android. Los bots se comunicarán con el botmaster de forma anónima y recibirán las instrucciones del mismo para realizar ciertas acciones o proporcionarle información. Todo esto, implementando persistencia, es decir, que aun cerrando la aplicación maliciosa o aun reiniciando el sistema, esta siga ejecutándose en segundo plano, recopilando información y realizando las instrucciones dadas por el botmaster.

El botmaster podrá ordenar en los bots que forman la botnet las siguientes acciones: el envío masivo de SMS a todos los contactos o a un contacto en particular, obtención de la ubicación del terminal, recopilación de las características del dispositivo, adquisición de un listado completo de todos los contactos, exfiltración de los mensajes de WhatsApp u otras redes sociales, monitorización de acciones del usuario del terminal etc. La comunicación entre los bots y el botmaster

es centralizada y el servidor podrá comunicarse en cualquier momento con cualquiera de los bots mediante la realización de las acciones remotas mencionadas.

El malware se podrá propagar mediante técnicas de ingeniería social como por ejemplo el envío masivo de SMS por parte de los dispositivos móviles infectados, cuyo contenido sea un enlace para descargar la aplicación y así infectar más dispositivos. Además, este camuflará en una aplicación que compruebe el estado de la batería o realice cualquier otro tipo de acción que visiblemente sea legítima para el usuario y así evitar sospechas por parte del mismo.

De esta forma, se podrán conocer las diferentes técnicas que utiliza este tipo malware y así poder mejorar la detección, eliminación y prevención de los mismos.

1.3. Objetivos personales

Los objetivos a nivel personal son por una parte el adquirir nuevos conocimientos sobre el funcionamiento de las botnets tanto a nivel general como específicamente en dispositivos móviles. Además, conocer en mayor profundidad el funcionamiento de un malware persistente que se ejecuta en un entorno móvil. Por otra parte, obtener mayor experiencia en el campo del desarrollo de aplicaciones Android. Otro objetivo es mejorar mis competencias en el desarrollo web utilizando frameworks, como en este caso Flask. Por otra parte, mejorar mi destreza en los lenguajes de Python y Java.

1.4. Entorno de pruebas

Se configurará un entorno de pruebas controlado para las distintas comprobaciones que se realizarán. El entorno consta de una máquina virtual con sistema operativo Kali Linux 2021.4a, la cuál será el cliente del servidor C&C y será la máquina que utilizará el botmaster. Por otra parte, consta del servidor C&C que se alojará en una Raspberry Pi 4. El entorno constará de diferentes redes. Por una parte, la red en la que se hallará el servidor de Comando y Control. Por otra parte, en otra red separada, la red en la que se emplazará el PC del botmaster. Y finalmente, otras dos redes domésticas separadas de las otras redes por dos router, que tendrá a su vez un cortafuegos (firewall). En las dos redes domésticas se encuentran los móviles que serán los objetivos del ataque, los cuales accederán mediante un punto de acceso inalámbrico (AP), que estará conectado al router (aunque normalmente el router contiene el mismo punto de acceso inalámbrico integrado). El objetivo del atacante será que la botnet se propague por todos los smartphones de la red cuyo sistema operativo sea Android, o incluso que se pueda extrapolar a smartphones que no estén dentro de esas redes en cuestión.

Capítulo 2

Estado del arte

2.1. Botnets

2.1.1. Definición de botnet

Según el Instituto Nacional de Ciberseguridad (INCIBE), una botnet es un conjunto de sistemas infectados que son controlados por parte del ciberatacante con el objetivo de realizar de manera conjunta tareas coordinadas.[2]. Por otra parte, según la Agencia Europea para la ciberseguridad (ENISA), una botnet es un conjunto de computadoras infectadas por bots. Esta agencia considera el bot como una pieza de software maliciosa que recibe órdenes de un maestro. [3] Los sistemas que son infectados pueden ser: ordenadores, servidores, routers, móviles o incluso dispositivos IOT. Por tanto, el objetivo de los atacantes que utilizan las botnets, principalmente no es infectar un dispositivo en particular, sino infectar a un gran número de dispositivos para realizar una acción coordinada con los mismos. Por ejemplo, el envío masivo de SMS, o ataques de denegación de servicio distribuidos (DDoS), exfiltración masiva de información... A los dispositivos que han sido infectados y por tanto forman parte de la botnet se les denomina bots o zombies. Las instrucciones que reciben los bots suelen provenir de un servidor central conocido como C&C (Command & Control), aunque existen otras variedades, tal y como se hablará en apartados posteriores. Las tareas más comunes que suele realizar una botnet son:

- Ataques de denegación de servicio Distribuidos. Este tipo de ataque consiste en lanzar un gran número de peticiones desde los bots para llegar a conseguir que el servidor al que van dirigidas las mismas se desborde y deje de estar disponible. Al utilizar una botnet, se utiliza la potencia no de un dispositivo, sino de varios para lanzar las peticiones, por lo que las probabilidades de éxito del ataque aumentan con respecto a si se realizara desde un único dispositivo.

- Envío masivo de spam, de contenido engañoso o fraudulento o envío de malware. Al enviar de forma masiva este tipo de contenido, se pueden lanzar más tipos de ataques como los relacionados con la ingeniería social (phishing).

- Generar tráfico de internet falso en un sitio web. Esto incluso se podría utilizar por ejemplo

para posicionar páginas en Google, ya que el robot de Google va posicionando en los primeros puestos aquellas páginas que generen más tráfico.

- Mostrar anuncios o notificaciones para que se pague un rescate a cambio de liberar el sistema de la botnet, o también para instalar un supuesto antimalware con dicho fin. El atacante en este caso obtendría beneficios económicos.

- Robo de información. Los datos son cada vez más valiosos por lo que el atacante podría obtener beneficio económico al venderlos. Al infectar muchos dispositivos se podría obtener una gran cantidad información sensible, desde datos personales hasta credenciales bancarias.

- Minado de criptomonedas. El minado de criptomonedas requiere una elevada potencia de cómputo, por lo tanto, el uso de una gran multitud de dispositivos para sumar la potencia de cada uno permitiría el minado sin ningún coste para el atacante.[4]

Por otra parte, los principales métodos de infección o vectores de ataque de las botnets son los que se suelen utilizar en los malware habituales:

- Vulnerabilidades no conocidas hoy en día (0-days). Estas vulnerabilidades son de las más peligrosas debido a que se desconocen y por tanto no han sido parcheadas aún en ningún sistema. Por tanto, se podrían aprovechar para infectar a cualquier sistema que las posea, por lo que se podrían infectar un gran número de sistemas.[1].

- Configuraciones del sistema inseguras y obsoletas. El uso de contraseñas débiles o la no aplicación de los parches de seguridad ofrecidos es un método de infección muy habitual, no solo utilizado en las botnets, sino en otro tipo de malware.

- Ingeniería social. Otro método muy utilizado es aprovecharse del fallo humano induciendo al error o al engaño mediante el uso de técnicas de ingeniería social. Por ejemplo, ofreciéndose el malware que infecte el dispositivo como una aplicación legítima y engañar así al usuario para que se lo descargue.

2.1.2. Orígenes de las botnets

El término de bot proviene de un viejo protocolo de comunicación llamado Internet Relay Chat (IRC), que permitía implementar un servicio de chat. En este protocolo, los usuarios podían desarrollar los llamados bots, los cuales permitían mantener vivos los canales, evitando la inactividad y entreteniendo a los participantes del canal entre otras acciones.

La primera botnet notable fue el Spammer de EarthLink. Esta botnet data del año 2000 y fue creada por Khan K. Smith. La principal función de dicha botnet fue enviar la mayor cantidad posible de correos electrónicos fraudulentos para realizar ataques de phishing. Gracias a esta botnet se enviaron aproximadamente 1.25 millones de emails fraudulentos. El objetivo era recopilar información confidencial como números de tarjeta de crédito. Smith utilizó la red de EarthLink

para estos propósitos.[5]

Otra botnet a destacar fue Storm, una de las primeras botnets conocidas cuya arquitectura era peer-to-peer (P2P). Por tanto los bots eran controlados desde varios servidores diferentes. La amplitud de la botnet era inmensa, ya que tenía entre 250 mil y 1 millón de ordenadores infectados. Esta botnet estuvo involucrada en multitud de actividades delictivas desde alquilar parte de ella a otro criminal hasta ataques DDoS o robos de identidad. Como es una botnet P2P es difícil determinar si sigue activa o inactiva debido a la falta de centralización. Actualmente se cree que está inactiva.[5]

Otra botnet a recalcar fue Cutwail, que enviaba 51 millones de correos electrónico por minuto, lo cual llegó a representar en torno a un 46 % del volumen total de spam mundial. El volumen de máquinas infectadas es notable (1.5 millones), de hecho debido a este elevado volumen los intentos por las autoridades de desmantelar dicha botnet han resultado ineficaces hasta la fecha. Actualmente sigue activa y disponible para alquilarse.[5]

Por otra parte, la botnet kraken fue de las primeras observadas en utilizar técnicas de evasión que le permitían evitar ser detectada mediante los software antimalware, incluso al actualizarse automáticamente. Aunque esta siga inactiva, sus restos han sido detectados por diferentes sistemas de seguridad y es posible que resurjan nuevamente en algún momento.[5]

En 2016, la botnet denominada Methbot, adquirió de forma fraudulenta cientos de miles de direcciones IP de dos registros DNS globales de internet y los asoció con Proveedores de Internet (ISP) con sede en EE.UU. Los que operaban esta botnet crearon más de 6000 dominios y 250000 URL distintas que parecían provenir de editores premium. De esta forma, lograron que los anunciantes pujaran por ellos y posteriormente enviaron sus bots para ver hasta 3000 millones de anuncios de vídeo todos los días. Methbot fue descubierto y desmantelado.[5]

Otra botnet a reseñar es Mirai [1]. Esta, se destaca en el presente trabajo, no porque estuvo detrás de un ataque DDoS masivo que dejó gran parte de Internet inaccesible en la costa este de EEUU (hecho reseñable), sino porque fue la primera red de bots importante que infectó dispositivos IoT inseguros. Esta botnet, que llegó a infectar gran cantidad de dispositivos (en torno a 600000), fue creada por un grupo de universitarios cuyo objetivo era obtener una ventaja en un popular juego llamado Minecraft.[5]

2.1.3. Clasificación de las botnets

Según la Agencia europea para la ciberseguridad ENISA, se podrían clasificar las botnets según su arquitectura en 3 grupos: Botnets centralizadas, botnets con proxies y botnets Peer-to-Peer.[3] El modelo de botnet más antiguo y simple es el modelo centralizado o modelo cliente-servidor. Este modelo funciona de forma que las instrucciones que reciben los bots son dadas mediante un único servidor central, llamado de Comando y Control (C&C). Este servidor, juega un papel

fundamental en la botnet, al ser centralizada, sin el servidor central, la botnet no tendría quien mande instrucciones, por lo que los bots no serían útiles, ya que no recibirían ninguna instrucción. Si los que se encargan de defender (blue team) toman el control del servidor central y lo inutilizan pueden desmantelar la botnet al completo. Encontrar el servidor no es una tarea difícil, basta con analizar el bot y realizar ingeniería inversa o analizar el tráfico del bot para saber cuál es el servidor con el que se comunica.

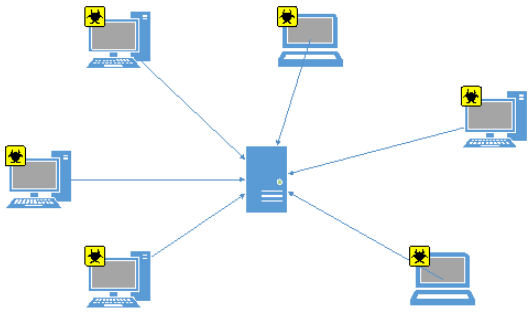

Figura 2.1: ENISA: Botnets centralizadas

Una variante del modelo mencionado en el párrafo anterior que permite dificultar la tarea de encontrar el servidor central C&C, es la de incluir proxies en la arquitectura de la botnet. Mediante este modelo, los bots no contactan con el servidor C&C directamente, sino que utilizan máquinas intermediarias que sirven de relays o de proxies. Los proxies pueden ser servidores operados por el botmaster o pueden ser los bots infectados que pertenecen a la botnet. Esto también permite que la infraestructura de la botnet sea más resiliente. Aún así, si el servidor central falla o es desmantelado la botnet dejaría de funcionar.

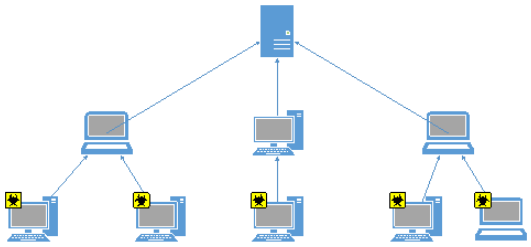

Figura 2.2: ENISA: Botnets con proxies

Otra arquitectura de las botnets, más compleja pero que ofrece ventajas adicionales, es la arquitectura Peer-to-Peer (P2P). En este modelo los bots contactan con otros bots y no con el servidor central C&C. La información y las instrucciones son propagados en la red desde un bot hacia otro bot. Para mantener el control de la botnet, el botmaster solamente necesita contactar con cualquiera de los bots infectados. Debido a la descentralización de la botnet, resulta muy complicado localizar a la persona que opera la botnet, y también resulta una tarea compleja el desmantelarla, ya que no hay un servidor central C&C propiamente dicho. Una forma, aunque

algo laboriosa, sería ir eliminado el malware de todas las máquinas infectadas y sacar parches de seguridad. Estos parches se liberarían en el caso de que se aproveche de una vulnerabilidad para propagarse o prevenir a los usuarios sobre los métodos de infección de dicha botnet.

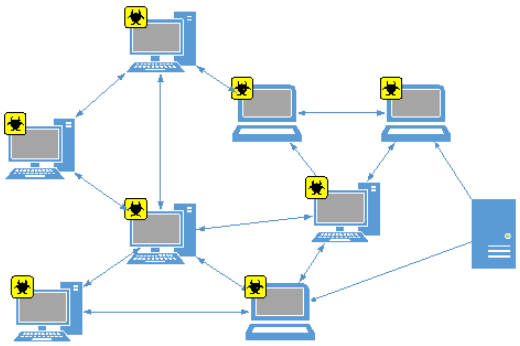

Figura 2.3: ENISA: Botnets peer to peer (p2p)

2.1.4. Panorama actual de las botnets

En los últimos años ha habido un crecimiento notable de los dispositivos IoT (Internet of Things). Esto ha hecho que estos dispositivos sean también un blanco para los cibercriminales. A esto hay que sumarle que diversas implementaciones de estos dispositivos que son accesibles desde internet han resultado inseguras. La heterogeneidad de fabricantes de estos dispositivos y la falta de actualizaciones de seguridad de los mismos los ha dejado expuestos a numerosos exploits. [1]

Ante todo esto, han surgido una nuevas amenazas para estos dispositivos, siendo una de ellas las botnets de IoT. Desde la primera mitad de 2020 no han dejado de crecer este tipo de botnets. Una de las que se puede destacar es Dark Nexus. Esta botnet se caracteriza por sus elaborados mecanismos para obtener información de los procesos en ejecución del dispositivo infectado. Para intentar reducir esta amenaza, los gobiernos están trabajando en nuevos estándares y nuevas leyes. Por ejemplo, en EE.UU. se aprobó recientemente la Ley de mejora de la seguridad cibernética de IoT, por las que el NIST desarrollará estándares de seguridad para el uso y administración de dispositivos IoT federales.

Otras botnets que resultan una amenaza importante en la actualidad, son aquellas cuyo objetivo es la infección con diferentes tipos del malware, principalmente de tipo troyano y stealer. En el primer trimestre de 2020 se identificaron un total de 2738 servidores C&C. Una botnet a resaltar es Emotet, que tuvo un enorme impacto y que, tras un descanso, volvió a resurgir recientemente destacando por su campaña de envío masivo de emails cuyo contenido era spam utilizando la temática del coronavirus. Además de emotet, recientemente se han seguido identificando campañas de otras amenazas como QNodeService, una nueva familia de malware que se instala en los equipos víctima a través de un script desarrollado en NodeJS. El uso de Javascript tiene diferentes ventajas como los bajos índices de detección por parte de los antivirus y una mayor compatibilidad con diferentes sistemas operativos. [1]

Otros objetivos potenciales de las botnets que cada vez son más demandados son los sistemas móviles entre los que se puede destacar los que tienen Android como sistema operativo instalado. Es muy similar a las botnets de dispositivos IoT, en cuanto a la tipología de ataques y malware objetivo. Las botnets que se pueden destacar en este ámbito son IPStorm y Matryosh. Un método de infección habitual en este tipo de dispositivos es la interfaz de depuración USB (ADB), la cual en ocasiones se encuentra expuesta en dispositivos con Android como por ejemplo Smart TV. Los fines de este tipo de botnets suelen ser parecidos que las botnets de IoT: ataques de denegación de servicio distribuidos o campañas de envío masivo de correos electrónicos o SMS en el caso específico de los móviles.

2.1.5. Perspectivas de futuro

Se espera que en los próximos años el número de dispositivos IoT aumente exponencialmente alcanzando los 30000 millones en 2025 y habiendo una media de 4 dispositivos de este tipo por persona. [1] Este aumento probablemente los hará más codiciados por los atacantes ya que podrían tener el control de una botnet con una cantidad enorme de dispositivos. A esto hay que sumarle los riesgos que tiene para la seguridad no solo lógica sino física por ejemplo de disponer una botnet cuyo botmaster controle multitud de dispositivos que se encarguen de la extinción automática de incendios. Incluso se prevé que las botnets se utilicen en guerra cibernética o ciberespionaje entre diferentes países. Además se estima que el número de botnets para móviles aumente debido a la enorme cantidad de datos sensibles que estos almacenan y al gran número de dispositivos que hay de este tipo.

2.1.6. Cómo protegerse de las botnets

La mayoría de las personas cuyos dispositivos han sido infectados y pertenecen a una botnet ni siquiera saben que la seguridad de los mismos ha sido comprometida. [6] La forma de protegerse de las botnets radica principalmente en la prevención y la concienciación para evitar que sea infectado el dispositivo por los métodos de infección habituales. Entre estas medidas se pueden recalcar:

- Actualizar el sistema operativo y las aplicaciones aplicando así los últimos parches de seguridad de los mismos. Como se ha mencionado un método habitual de infección es aprovecharse de las vulnerabilidades ya conocidas.

- Evitar descargarse ficheros que no provengan de fuentes conocidas y fiables, así como abrir enlaces desconocidos. Ante la duda es mejor no abrir el enlace por mucha curiosidad que entre.

- En el caso de Android, sospechar de las apps que soliciten multitud de permisos o de aquellas que soliciten permisos que no tengan sentido. Por ejemplo no tiene sentido que una calculadora solicite permisos de ubicación.

- Utilizar cortafuegos o firewall cuando se navegue por internet para evitar conexiones no deseadas. Tanto en Mac OS como en Windows suele haber un cortafuegos por defecto. Sin embargo en Android no hay ningún cortafuegos por defecto. Se recomienda instalar un cortafuegos fiable.

- En el caso de los sistemas móviles, evitar conectar el dispositivo a puertos USB o cargadores los cuales no sean conocidos, como por ejemplo puertos públicos. Cada vez es más habitual que este tipo de puertos USB contengan un hardware específico que por ejemplo simule un teclado que realice acciones no deseadas en el dispositivo como aprovecharse de la interfaz USB para instalar malware.

- Otra medida es la de evitar conectarse a redes WiFi públicas o puntos de acceso desconocidos que están abiertos. Este tipo de puntos de acceso suelen utilizarse tanto para robar credenciales como para redirigir a una página que descargue malware e infecte el dispositivo.

2.2. Android

2.2.1. Sobre el sistema operativo Android

Android es un sistema operativo de código abierto basado en el kernel de linux diseñado para dispositivos móviles. Su proyecto de código abierto está dirigido por Google. Al ser código abierto, es habitual que los fabricantes de dispositivos móviles modifiquen el código fuente con el objetivo de adaptarlo para que sea compatible con sus dispositivos y así ofrecer al usuario un entorno más confiable y robusto. Android se puede adaptar a prácticamente cualquier dispositivo. Aunque lo más habitual es encontrar versiones de Android para procesadores con arquitectura ARM, hay versiones de este sistema operativo adaptadas para procesadores x86, utilizadas principalmente para virtualización. La arquitectura de este sistema operativo está dividida en los siguientes componentes[7]:

- Framework de aplicaciones. Los desarrolladores de apps son los que utilizan con más frecuencia este framework. Es la capa de mayor abstracción, es decir, de más alto nivel.

- Binder IPC Proxies. Permite que el framework de aplicaciones cruce los límites del proceso y haga llamadas al código de los servicios del sistema Android. En el framework de aplicaciones esta comunicación está oculta para el desarrollador.

- Servicios del sistema. Son componentes modulares enfocados, como por ejemplo el administrador de ventanas, el servicio de búsqueda o el administrador de notificaciones. Las funcionalidades que se exponen por las APIs del framework de aplicaciones se comunican con los servicios del sistema para acceder al hardware subyacente. Android aglutina los servicios en dos grupos: sistema y medios.

- Capa de abstracción de hardware (HAL). Una HAL define una interfaz estándar para que la implementen los proveedores de hardware. Esto permite que Android sea independiente

de las implementaciones de controladores de más bajo nivel. El uso de una HAL es muy útil para implementar la funcionalidad sin afectar o modificar el sistema de más bajo nivel. Las implementaciones de HAL se empaquetan en módulos y el sistema Android las carga en el momento adecuado.

- Núcleo de Linux. Al estar basado Android en Linux, el desarrollo de sus controladores de dispositivo o drivers es similar al desarrollo de un controlador típico de linux. Android usa una versión del kernel de Linux con algunas funciones especiales, como el Low Memory Killer, que es un sistema de administración de memoria más agresivo para preservar memoria puesto que los dispositivos móviles hasta hace muy poco solían tener más limitaciones de memoria que los ordenadores portátiles o de sobremesa. Otras funciones especiales son Wake Locks, un servicio del sistema de gestión de energía, el controlador Binder IPC y otras características que se consideran importantes para una plataforma integrada para móviles. Estas adiciones especiales son principalmente para la funcionalidad del sistema en sí y no afectan al desarrollo de controladores.

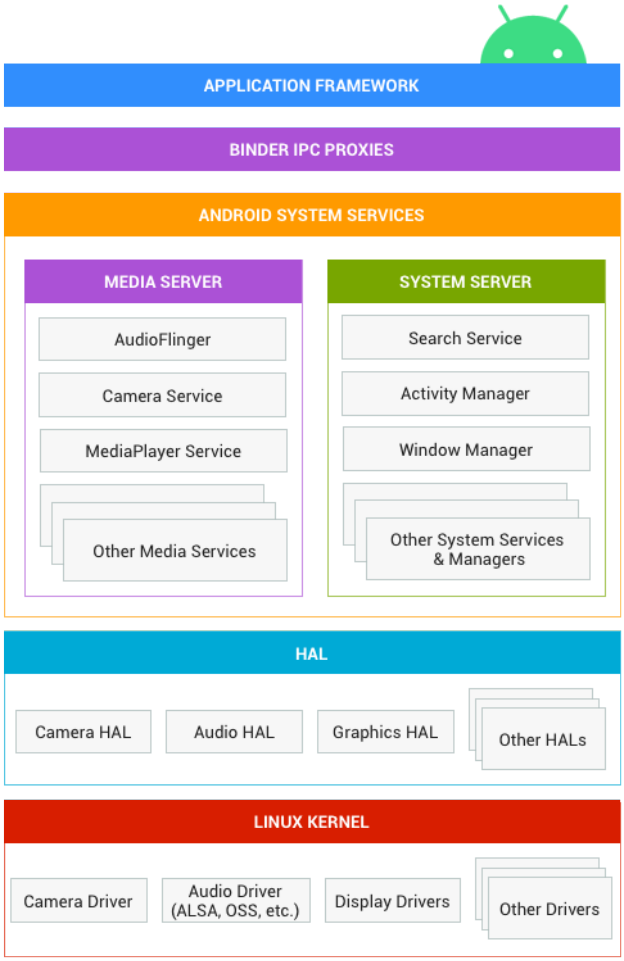

Figura 2.4: Arquitectura del sistema operativo Android

2.2.2. Arquitectura de seguridad de Android

App sandbox

Android, como se ha mencionado, al estar basado en el núcleo de linux aprovecha la protección basada en el usuario de Linux para identificar, encapsular y aislar los recursos de la aplicación. Para ello, asigna un identificador de usuario único (UID) a cada aplicación y ejecuta la misma en su propio proceso. Android usa este identificador para configurar una zona de pruebas de aplicaciones a nivel de kernel (app sandbox). Esto evita por ejemplo que una aplicación maliciosa pueda interferir con otra por ejemplo inyectando código malicioso en la memoria de la otra aplicación.[8]

Firma de aplicaciones

La firma digital de aplicaciones permite a los desarrolladores identificar al autor de la aplicación y actualizar su aplicación, sin crear interfaces o permisos complejos. Además, la tienda de aplicaciones de Google Play Store utiliza esta firma para identificar al autor y evitar también plagios. Todas las aplicaciones que se ejecutan en la plataforma Android por defecto deben estar firmadas por el desarrollador. Si el usuario desea instalar una aplicación que no esté firmada o cuyo desarrollador no se ha reconocido en la Google Play Store deberá activar la opción de permitir de los orígenes desconocidos en ajustes.[9]

Autenticación

Android utiliza el concepto de claves criptográficas, las cuales son controladas por autenticación de usuario. Esta, requiere el almacenamiento de las claves criptográficas y proveedores de servicios así como de autenticadores de usuarios. El subsistema Gatekeeper realiza la autenticación de patrón / contraseña de dispositivo en un entorno de ejecución confiable (TEE). A partir de Android 9 se incluye la confirmación protegida, la cual proporciona a los usuarios una forma de confirmar las transacciones críticas, como por ejemplo los pagos.[8]

Cifrado

Desde la versión 6.0 Marshmallow, todos los dispositivos están cifrados, es decir todos los datos del almacenamiento interno están cifrados. En un dispositivo cifrado, todos los datos que son creados por el usuario se cifran automáticamente antes de escribirlos en disco. De la misma forma, todas las lecturas que se realizan desde disco se descifran automáticamente antes de entregar los datos al proceso que realiza la llamada al sistema o syscall. El cifrado garantiza que en el caso de que una entidad no autorizada o maliciosa intente acceder a los datos, no podrá leerlos, asegurando así la confidencialidad de la información.[9]

Almacén de claves

El sistema Android incluye un almacén de claves apoyado por hardware, el cual permite generar, importar y exportar claves asimétricas, importar claves simétricas sin procesar y también da soporte a cifrado asimétrico, siendo compatible con algoritmos como RSA. La API de Android Keystore

y el Keymaster del HAL subyacente proporcionan un conjunto de primitivas criptográficas que permite la implementación de protocolos utilizando claves de acceso controlado y respaldado por hardware.[9]

SELinux

Security-Enhanced Linux, también llamado SELinux, que consiste en un módulo de seguridad que se puede añadir al kernel de linux, utilizado para el control de acceso obligatorio (MAC) en todos los procesos, incluyendo los que se ejecutan con privilegios de superusuario o root. La arquitectura de este módulo se enfoca en separar las decisiones de las aplicaciones de seguridad, de las políticas de seguridad y en racionalizar la cantidad de software encargado de las aplicaciones de seguridad.[9]

Entorno de ejecución confiable (TEE)

Trusty consiste en un sistema seguro que proporciona un entorno de ejecución confiable o TEE para Android. El sistema se ejecuta en el mismo procesador que Android. Sin embargo, Trusty está aislado del resto del sistema, tanto por hardware como por software. Cabe recalcar, que Trusty y Android funcionan paralelamente. Trusty tiene acceso a toda la potencia del procesador principal y a la memoria del dispositivo, pero está completamente aislado. Esto añade una capa de abstracción y encapsulación adicional que es de gran utilidad para proteger el hardware del dispositivo de aplicaciones maliciosas instaladas por el usuario. Los entornos de ejecución de confianza (TEE) son cada vez más comunes en los móviles debido a las necesidades de seguridad que surgen. En los dispositivos con una implementación de TEE, el procesador principal se le suele denotar como "no confiable". Esto significa que no puede acceder a ciertas áreas de la memoria principal, registros de hardware y fusibles de escritura única donde se encuentran datos secretos. El software que se ejecuta en el procesador principal debe delegar cualquier operación que requiera el uso de datos secretos al procesador TEE.[8]

Arranque verificado

El principal objetivo de un arranque verificado es garantizar que todo el código ejecutado provenga de una fuente confiable (como OEM), en lugar de una fuente maliciosa, corrupta o una fuente manipulada por un atacante. El arranque verificado funciona gracias a una cadena de confianza completa que comienza desde una raíz de confianza que está protegida por hardware. La cadena de confianza continua por el cargador de arranque o bootloader, la partición de arranque y otras particiones verificadas (como `/system`, `/vendor` o `/oem`). En cada etapa del arranque del dispositivo se verifica la integridad y autenticidad de la siguiente etapa antes se entregar a la siguiente etapa el relevo de la ejecución. Esto adicionalmente ayuda a comprobar que la versión de Android que se está ejecutando es la correcta utilizando protección contra reversión. Este tipo de protección ayuda a prevenir que una posible vulnerabilidad se vuelva persistente al garantizar que los dispositivos solo se actualicen a versiones más recientes de Android y no a versiones anteriores con sus vulnerabilidades ya conocidas.[**13**]

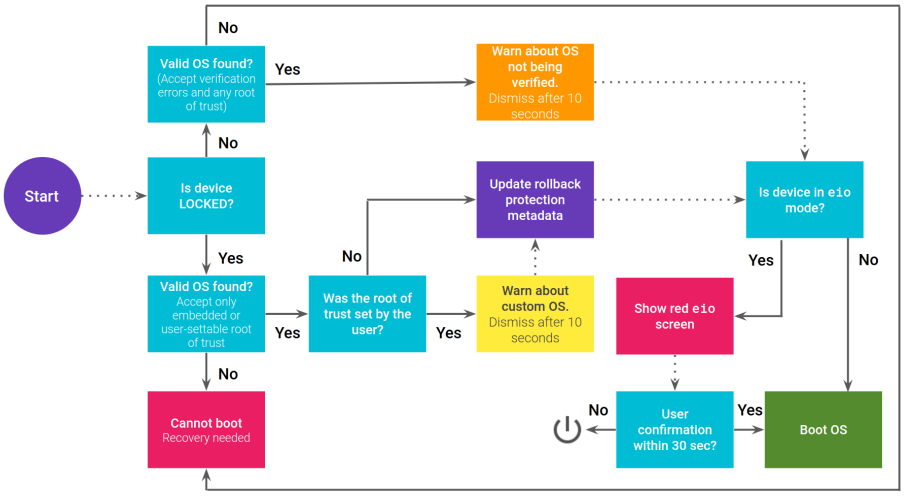

Figura 2.5: Diagrama de flujo de la secuencia del arranque verificado - Android

Seguridad de red

Adicionalmente, Android proporciona mecanismos de seguridad de red con el objetivo de proteger los datos que se envían a través de la red.

Las versiones más recientes de Android incluyen el soporte integrado para utilizar DNS sobre TLS, que permite que el tráfico de las consultas y respuestas para resolver un dominio vaya cifrado. Además, ofrece mecanismos de protección del tráfico entrante y saliente del dispositivo utilizando TLS. Por defecto en las últimas versiones de Android, la configuración de red bloquea todo el tráfico sin cifrar (texto plano). Por otra parte, además soportan la versión 1.3 de TLS, que incluye mejoras de seguridad y rendimiento. Además, se agregaron mejoras en el manejo de certificados TLS donde a los usuarios solo se les pide que elijan de certificados que cumplen con los requisitos especificados por el servidor (en cumplimiento con RFC5246).

En a las conexiones WiFi, a partir de la versión 10.0 se incorpora la compatibilidad con WPA3 y estándares abiertos mejorados de WiFi. WPA3 y WiFi Enhanced Open mejoran la seguridad en redes inalámbricas WiFi en general y proporcionan una mayor privacidad y solidez frente a ataques. Proporciona cifrado y privacidad al conectarse a redes abiertas no protegidas mediante contraseñas en aéreas como cafeterías. hoteles, aeropuertos, bibliotecas etc.

Además se admite la conexión segura empresarial mediante una red privada virtual (VPN). Incluso la VPN se puede configurar para que las apps no tengan acceso a la red hasta que se establezca una conexión VPN, lo que evita que las apps envíen datos a otras redes que no sean la de la VPN. Además los usuarios pueden configurar manualmente clientes VPN siempre activos que implementen métodos VPNService.[8]

2.2.3. Buenas prácticas de seguridad para dispositivos Android

En el presente trabajo de fin de grado se demuestra lo peligroso que puede llegar a ser una aplicación aparentemente inofensiva. Por ello, es fundamental la concienciación de los usuarios en la materia de ciberseguridad. A continuación se enumeran distintas buenas prácticas que pueden realizar los usuarios que según el CCN-CERT [10] se consideran importantes para contribuir a que sus dispositivos móviles sean más seguros.

Bloquear el móvil con un código de acceso o utilizando biometría

Bloquear el móvil con una contraseña robusta o mediante huella dactilar es el primer paso para tener un dispositivo móvil relativamente seguro. Si el smartphone dispone de mecanismos de autenticación mediante biometría como la huella dactilar es recomendable utilizarlos. Aunque se configure una huella dactilar en un terminal, los sistemas como iOS y Android exigen una contraseña o un PIN por si falla dicho mecanismo. Es recomendable utilizar una contraseña preferiblemente robusta que tenga al menos 10 caracteres, que no tengan mucho sentido ni se asemejen a ninguna palabra y además es recomendable que contenga al menos un carácter especial, una mayúscula y un número. Además de esto se recomienda limitar las funcionalidades y la información que muestra la pantalla de bloqueo, como las notificaciones que muestren datos sensibles.

Tener especial precaución al conectar el móvil por USB

El juice jacking es una técnica ampliamente utilizada por los cibercriminales que consiste en realizar acciones no autorizadas como instalar malware o robar información aprovechando la información que se puede transferir mediante el puerto USB del terminal [10]. Para ello crean estaciones de carga falsas a las que los usuarios conectan sus dispositivos a los puertos USB de las mismas creyendo que solamente van a cargar su terminal. Esto puede comprometer el móvil si este se conecta a un puerto o a un dispositivo que aproveche dicha conexión para realizar acciones no autorizadas en el móvil. A mayores se recomienda desactivar la depuración USB, salvo casos estrictamente necesarios. La depuración USB mediante la interfaz ADB permite, realizar comandos remotos en el móvil e incluso instalar aplicaciones en el mismo.

Actualizar el sistema operativo y las aplicaciones a su versión más reciente

Con mucha frecuencia se descubren diferentes vulnerabilidades en dispositivos móviles. Unas de las más habituales son las vulnerabilidades que permiten escalar de forma remota de privilegios sin necesidad de privilegios de ejecución adicionales. En uno de los últimos boletines de seguridad de Android en el que se detallan las vulnerabilidades descubiertas y los parches aplicados, se la vulnerabilidad considerada de grado más crítico fue una de este tipo [11]. Es por ello por lo que los fabricantes ponen a disposición de los usuarios actualizaciones de software que contienen parches que solucionan las mencionadas vulnerabilidades. Los cibercriminales utilizan herramientas ofensivas que explotan estas vulnerabilidades que han sido descubiertas anteriormente y por tanto son conocidas. Por lo que no actualizar el smartphone hace que este sea más vulnerable a posibles

ataques. Aún así, aunque se actualice a la última versión existen vulnerabilidades no conocidas, al menos públicamente. Estas vulnerabilidades se denominan 0-day. Para hacer frente a estas vulnerabilidades se recomienda tener especial precaución a la hora de abrir SMS o archivos multimedia o enlaces web que no se conocen o se consideran extraños, principalmente si el origen del mensaje se desconoce.

Cifrar el dispositivo móvil

Se recomienda utilizar las capacidades nativas de cifrado del dispositivo móvil para proteger la confidencialidad de los datos e información asociada al usuario. En las últimas versiones de Android es obligatorio el cifrado por lo que por defecto estará habilitado. En iOS también se habilita el cifrado automáticamente en cuanto se establece un código de acceso. Sin embargo, un aspecto a recalcar es que el cifrado de los datos se realiza sobre la memoria interna del móvil y no sobre una memoria externa como la tarjeta SD. Si se tiene una tarjeta SD insertada en el smartphone se recomienda cifrarla manualmente o evitar almacenar en ella información que se considere sensible.

Modificar configuración por defecto

La configuración de fábrica de los dispositivo móviles en muchas ocasiones puede resultar insegura y habilitar ciertas funcionalidades que podrían permitir al atacante comprometer los datos del terminal (por ejemplo el NFC activado por defecto). Además está configuración puede contribuir a dar información de las especificaciones del dispositivo. Es bastante frecuente encontrar como nombre de emparejamiento bluetooth del dispositivo la marca y el modelo del dispositivo. Esto da información al atacante para lanzar un exploit adaptado a ese dispositivo en concreto. También en la configuración por defecto puede que haya servicios que no se vayan a utilizar activados que pueden ser aprovechados para atacar el dispositivo. En consecuencia, se recomienda desactivar los servicios que no se vayan a utilizar permanentemente por parte del usuario.[10]

Copias de seguridad

El móvil, es un dispositivo que se suele transportar de un lugar a otro con mucha frecuencia. Por tanto, es más susceptible a que se pierda o se robe. En tales casos la información que está en el mismo se perderá. Para prevenir y mitigar en el caso de que ocurra, se recomienda realizar una copia de seguridad de los datos del mismo en un almacenamiento externo, en otro dispositivo o a la nube. Hay aplicaciones que realizan estas copias de forma automática por defecto. Sin embargo, hay otros datos como las imágenes que puede que el dispositivo no realice de forma automática la copia de los mismos a la nube.

Familiarizarse con las herramientas de gestión remota del terminal

Muchos dispositivos móviles modernos ofrecen herramientas que permiten la gestión remota del terminal. Estas, disponen de funcionalidades como la geolocalización de forma remota del dispositivo, el bloqueo del mismo, hacer que suene para poder encontrarlo o mostrar un mensaje para

que en el caso de que alguien lo encuentre pueda devolverlo a su propietario. Se recomienda que el usuario se familiarice con este tipo de herramientas y que compruebe si funcionan correctamente. Para hacer uso de estos servicios es habitual que los usuarios se deban registrar en la web donde le indique el fabricante. Se aconseja proteger la cuenta con la que acceda a las herramientas de gestión remota con al menos 2 factores de autenticación para evitar que los cibercriminales o gente no deseada hagan un uso indeseado de las mismas.[10]

Medidas en conexiones inalámbricas

Los dispositivos móviles más modernos disponen de una tecnología llamada NFC que permite comunicaciones inalámbrica de corto alcance para realizar transacciones sin contacto como pagos o para autenticarse físicamente. Para evitar transacciones o pagos fraudulentos y otro tipo de ataques se debe activar el NFC solamente cuando sea necesario utilizarlo. Lo mismo se recomienda con el bluetooth, ya que tener activado el mismo en todo momento podría permitir a un atacante manipular las comunicaciones y acciones asociadas al resto de dispositivo así como la información intercambiada entre estos.[10]

En cuanto al WiFi, una de las amenazas más habituales en los dispositivos móviles es la conexión de los mismos a redes públicas sin proteger o a puntos de acceso falsos. Por ello es recomendable conectarse a solamente redes WiFi que sean de confianza del propietario del terminal y que además estén protegidas mediante contraseña. En el caso de que sea necesaria la conexión a una red pública sin proteger, se sugiere el uso de una red privada virtual (VPN) mediante la cual los datos transmitidos por la red WiFi irán cifrados mediante una capa adicional.

Otro aspecto para tener en cuenta es que en las redes móviles hay que ser conscientes que las redes 2G se siguen utilizando cuando no hay una antena 3G o 4G próxima. Las redes 2G al ser antiguas no hacen uso de ningún mecanismo de seguridad que permita al dispositivo móvil estar seguro de que se está conectando a una red legítima del operador. Son habituales los ataques en los que el cibercriminal suplanta al operador de red y así conseguir que se conecte a un punto de acceso manipulado por el en el que pueda interceptar el tráfico del móvil. En ningún caso se debe configurar el móvil para dar prioridad a las redes 2G y si es posible se recomienda desactivarlas.

Por otra parte la compartición de la ubicación del dispositivo de manera constante o en tiempo real puede suponer un riesgo para la privacidad y seguridad de los usuarios de los dispositivo móviles. También hay que tener especial precaución a los metadatos que tienen las fotografías realizadas por el terminal. En muchas aplicaciones de cámara está activado por defecto el geoetiquetado de las imágenes. Por tanto cualquier persona que obtenga esa imagen podrá saber exactamente donde se tomó.

Medidas en las aplicaciones

Se recomienda no instalar aplicaciones desde mercados de terceros o en los sistemas con Android descargándose ficheros APK. Estas aplicaciones, como se expone en el presente TFG, pueden tener código dañino. Se recomienda instalar aplicaciones provenientes de los mercados oficiales como la App Store o la Play Store. También se recomienda encarecidamente comprobar los permisos que se le otorgan a cada aplicación y desactivar los permisos que no se utilicen o que se consideren innecesarios o irrelevantes. Por ejemplo no tendría sentido y sería sospechoso que una aplicación como la que se expone en el presente trabajo, que sirve aparentemente para conocer ciertos parámetros de la batería, tenga permisos de ubicación. Ante la duda se sugiere desinstalar la aplicación o desactivar los permisos de los que se dude su relevancia.
En cuanto al navegador se recomienda usar siempre conexiones cifradas HTTPS y no pinchar en enlaces que se desconozcan o que requieran introducir información sensible. Se debe comprobar si la dirección URL que se puede visualizar en la parte superior de la pestaña del navegador coincide con la original de la web a la que se está accediendo.

2.3. MITRE ATT&CK

MITRE ATT&CK ® es un proyecto que permite a cualquier persona que desee conocer en profundidad diversas tácticas y técnicas del ciberatacante las cuales están basadas en observaciones del mundo real. Esto permite tanto a entidades gubernamentales como a empresas del sector privado elaborar metodologías y modelos de amenazas específicos. MITRE inició este proyecto en 2013 con el objetivo de documentar tácticas, técnicas y procedimientos comunes los cuales eran utilizados por las amenazas persistentes avanzadas (APTs) contra las redes empresariales. Se creó debido a la necesidad de documentar los comportamientos de los ciberatacantes para su uso dentro de un proyecto de investigación MITRE llamado FMX, el cual pretendía mejorar la detección posterior de estas amenazas. Basándose en este proyecto de investigación los investigadores del MITRE decidieron que era necesario un marco para abordar 4 problemas: Los comportamientos de los ciberatacantes cambiantes, modelos del ciclo de vida del malware (Cyber Kill Chain) que no encajaban, aplicabilidad a entornos reales y agrupar tácticas y técnicas de diferentes tipos de grupos de atacantes en una taxonomía común. [12]

ATT&CK organiza las técnicas en un conjunto de tácticas con el objetivo facilitar el entendimiento de las mismas y proporcionarlas cierto contexto. Las tácticas representan el porque de una técnica, es decir sirven como categorías contextuales que son útiles para técnicas individuales. Describen a alto nivel qué hacen los adversarios durante una operación. Un ejemplo de táctica sería persistencia, descubrir información, movimiento lateral etc. Por otra parte según MITRE las técnicas representan el cómo un adversario consigue su objetivo táctico al realizar una acción. Un ejemplo de técnica puede ser el volcado de credenciales de una red para utilizarlas para conseguir movimientos laterales. Las técnicas también pueden representar lo que gana un adversario al realizar una acción. [13]

En la figura se puede apreciar la Matriz ATT&CK en la que se representa la relación entre las tácticas y las técnicas. Por ejemplo bajo la categoría de la táctica de persistencia hay diferentes técnicas como Dylib Hijacking, Servicios Remotos Externos, aprovechar las debilidades de los permisos de los permisos del sistema de archivos etc. Además MITRE ofrece una matriz específica para sistemas móviles para dispositivos Android e iOS En el presente TFG se documentarán las diferentes tácticas y técnicas utilizadas.[13]

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command And Control
10 items	31 items	56 items	28 items	59 items	20 items	19 items	17 items	13 items	9 items	21 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Command-Line Interface	AppCert DLLs	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Control Panel Items	AppInit DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	File and Directory Discovery	Exploitation of Remote Services	Data from Information Repositories	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Application Shimming	Clear Command History	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Local System	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Execution through API	Authentication Package	Bypass User Account Control	CMSTP	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Network Shared Drive	Exfiltration Over Command and Control Channel	Data Encoding
Spearphishing via Service	Execution through Module Load	BITS Jobs	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Password Policy Discovery	Pass the Ticket	Data from Removable Media	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Dylib Hijacking	Component Firmware	Forced Authentication	Peripheral Device Discovery	Remote Desktop Protocol	Data Staged	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Graphical User Interface	Browser Extensions	Exploitation for Privilege Escalation	Component Object Model Hijacking	Hooking	Permission Groups Discovery	Remote File Copy	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	InstallUtil	Change Default File Association	Extra Window Memory Injection	Control Panel Items	Input Capture	Process Discovery	Remote Services	Input Capture		Multi-hop Proxy
	Launchctl	Component Firmware	File System Permissions Weakness	DCShadow	Input Prompt	Query Registry	Replication Through Removable Media	Man in the Browser		Multi-Stage Channels
	Local Job Scheduling	Component Object Model Hijacking	Hooking	Deobfuscate/Decode Files or Information	Kerberoasting	Remote System Discovery	Shared Webroot	Screen Capture		Multiband Communication
	LSASS Driver	Create Account	Image File Execution Options Injection	Disabling Security Tools	Keychain	Security Software Discovery	SSH Hijacking	Video Capture		Multilayer Encryption
	Mshta	DLL Search Order Hijacking	Launch Daemon	DLL Search Order Hijacking	LLMNR/NBT-NS Poisoning	System Information Discovery	Taint Shared Content			Port Knocking
	PowerShell	Dylib Hijacking	New Service	DLL Side-Loading	Network Sniffing	System Network Configuration Discovery	Third-party Software			Remote Access Tools
	Regsvcs/Regasm	External Remote Services	Path Interception	Exploitation for Defense Evasion	Password Filter DLL	System Network Connections Discovery	Windows Admin Shares			Remote File Copy
	Regsvr32	File System Permissions Weakness	Plist Modification	Extra Window Memory Injection	Private Keys	System Owner/User Discovery	Windows Remote Management			Standard Application Layer Protocol
	Rundll32	Hidden Files and Directories	Port Monitors	File Deletion	Replication Through Removable Media	System Service Discovery				Standard Cryptographic Protocol
	Scheduled Task	Hooking	Process Injection	File System Logical Offsets	Securityd Memory					Standard Non-Application Layer Protocol
	Scripting	Hypervisor	Scheduled Task	Gatekeeper Bypass	Two-Factor Authentication Interception					Uncommonly Used Port
	Service Execution	Image File Execution Options Injection	Service Registry Permissions Weakness	Hidden Files and Directories						Web Service
	Signed Binary Proxy Execution	Kernel Modules and Extensions	Setuid and Setgid	Hidden Users						
	Signed Script Proxy Execution	Launch Agent		Hidden Window						
	Source			HISTCONTROL						
	Space after Filename			Image File Execution Options Injection						

Figura 2.6: Matriz ATT&CK

Capítulo 3

Plan de proyecto

3.1. Actividades y entregables

En el cuadro 3.1, se pueden distinguir las distintas tareas del proyecto, así como la estimación temporal de cada una de ellas. Todas ellas suman un total de 300 horas, que es el tiempo previsto para la realización del presente trabajo. En primer lugar hay que recalcar que todas las tareas llevan implícita su respectiva fase de documentación. Por ejemplo, la tarea `T005` que consiste en el análisis de requisitos y el diseño del software, no solamente se centra en el proceso en sí, sino que se también incluye la elaboración de la documentación resultante de la misma tarea. Esta documentación forma parte de los entregables de la tarea. Hay 2 tareas cuyo código es `T001`, `T002` y `T020`, que son exclusivamente de documentación. El resto incluye tanto la realización de la tarea como la elaboración de la documentación resultante.

Las primeras tareas consisten en investigación en diferentes fuentes, recopilación de información y documentación sobre el tema. Una vez se ha investigado y documentado, se evalúan las distintas alternativas para desarrollar la solución más adecuada al problema que se está afrontando (`T003`). Una vez se han evaluado las distintas soluciones y se ha elegido la más adecuada, se despliega un primer entorno de laboratorio inicial sobre el que se trabajará. Este entorno consta de distintas máquinas virtuales, tanto las que son los bots con Android instalado, como las del botmaster y el servidor C&C. Una vez hecho esto se comienza a realizar el diseño inicial del software. Esta tarea incluye: análisis de requisitos, diagramas de dominio, diagramas de paquetes... A continuación se comenzaría con el diseño de la base de datos. Una vez diseñada, se repasan los distintos conceptos avanzados de desarrollo de una aplicación Android. Esto incluye la formación sobre: tareas programadas, alarmas, broadcast receivers, servicios etc. De esta forma se podrá evaluar la que mejor se adecúe al presente proyecto. Una vez realizado lo anterior, se desarrollan paralelamente el módulo de conexión del servidor con la base de datos distribuida y el módulo de conexión del bot con la misma base de datos distribuida. A continuación, se desarrollaría el módulo que se encarga de ejecutar las tareas de consulta y ejecución de los comandos proporcionados por el servidor C&C y del almacenamiento del resultado en la base de datos distribuida. Todo ello en segundo plano. Una vez hecho esto se realizará una primera prueba de concepto del funcionamiento básico

de la botnet con la ejecución de un comando sencillo escribiendo en la base de datos distribuida dicho comando. Una vez realizada la prueba de concepto, se evaluarían las distintas técnicas para adquirir persistencia en Android y se elige la que se considere más adecuada. Una vez elegida, se implementa la persistencia, de forma que aunque cierre la app o la borre de recientes o incluso reinicie, esta se siga ejecutando en segundo plano. Posteriormente se desarrollarían paralelamente la interfaz web del servidor y el módulo de la aplicación que se encarga de recopilar información confidencial del terminal y monitorizar las acciones del usuario. Finalmente se realizarían varias pruebas para comprobar que funciona la botnet correctamente. Hay que recalcar que el tiempo estimado para la realización del proyecto es de 300 horas.

Por otra parte, se puede visualizar en la figura 3.1, las dependencias de las tareas. En un principio, se considera que todas las tareas depende de la anterior, ya que en su mayoría ocurre que hasta que no esté terminada la anterior no puede arrancar la siguiente. Además como se dispone de solamente un recurso personal si se desearan realizar de forma paralela varias tareas no mejoraría el tiempo de entrega del proyecto ya que se le dedicaría parcialmente el tiempo a una y la otra parte del tiempo a la otra.

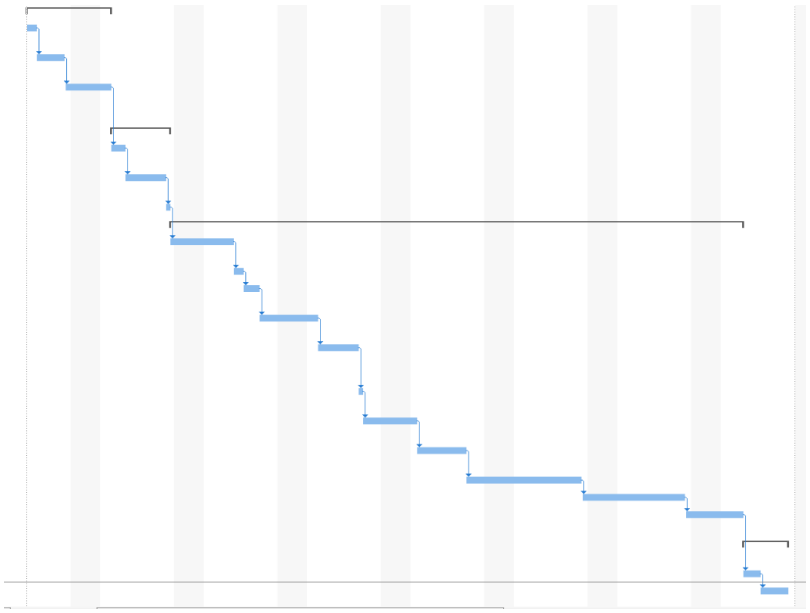

Figura 3.1: Diagrama de Gantt - MS Project

3.2. Plan de riesgos y estimación de costes

3.2.1. Plan de riesgos

Según el PM-BOK [14], un riesgo es un acontecimiento o condición inciertos que, si se producen, tienen un efecto positivo o negativo en los objetivos de un proyecto. Por otra parte, el PRINCE2 [14] define el riesgo como la la posibilidad de exponerse a las consecuencias adversas de acontecimientos futuros. De cualquier forma, se recomienda seguir una serie de pasos para realizar

Código	Nombre	Duración
T001	Redacción del apartado de introducción y objetivos de alto nivel	5h
T002	Investigación en diferentes fuentes y redacción del apartado del estado del arte	15h
T003	Investigación, aprendizaje y evaluación de las distintas alternativas de comunicación del servidor C&C con los bots (Firebase, twitter, GRPC...)	10h
T004	Creación y configuración del entorno del servidor en Linux. Instalación de Apache o Nginx, Python3, Docker y las librerías necesarias	7h
T005	Análisis de requisitos y diseño preliminar del software	20h
T006	Diseño de la base de datos distribuida	5h
T007	Formación y Repaso sobre programación avanzada en Android (tareas programadas, broadcasts recivers, servicios etc.)	18h
T009	Despliegue de un servidor web con Python	5h
T010	Desarrollo del módulo de conexión del servidor web C&C con la base de datos distribuida	10h
T011	Desarrollo del módulo de conexión del cliente Android (bot) con la base de datos distribuida	15h
T012	Desarrollo del módulo de la app Android realiza tareas en segundo plano de consulta de los comandos, ejecución de los mismos y almacenamiento de la respuesta en la BBDD distribuida	20h
T013	Desarrollo de una primera prueba de concepto (POC) de la botnet	5h
T014	Evaluación de las diferentes técnicas para adquirir persistencia	10h
T015	Desarrollo del módulo que permite la adquisición de persistencia	30h
T016	Implementación de las diferentes instrucciones o comandos que ofrece el bot	45h
T017	Desarrollo de la interfaz web del servidor C&C	40h
T018	Desarrollo del módulo encargado de la recopilación de información del bot	15h
T019	Realización de diferentes pruebas	10h
T020	Redacción de aspectos restantes de la memoria	15h
Total	-	**300h**

Cuadro 3.1: Tareas del proyecto

una correcta planificación de los riesgos. En primer lugar, se debe realizar una identificación de los riesgos. Esto se puede realizar revisando listas de comprobación de riesgos dados en proyectos anteriores semejantes o mediante una lluvia de ideas. Posteriormente se deben analizar y priorizar. Esto se realizará mediante una elaboración de matriz de riesgos como la que se expone en la figura 3.2. Esta cuantificar el riesgo, analizando la probabilidad de que dicho riesgo se materialice y el impacto que pueda generar. Una vez priorizados, siendo los más prioritarios que tienen mayor impacto y mayor probabilidad de desencadenarse, se deben de proponer una serie de contramedidas que sirvan tanto para reducir la posibilidad de que se de como para mitigarlo en caso de que se materialice. Una vez hecho esto, durante la ejecución del proyecto se debe de realizar un seguimiento y monitorización de los mismos. En la tabla 3.2.1 se puede apreciar los principales riesgos encontrados para el presente proyecto.

A continuación se realiza el análisis y priorización de riesgos que culmina con la realización de la matriz que se puede apreciar en la tabla 3.2

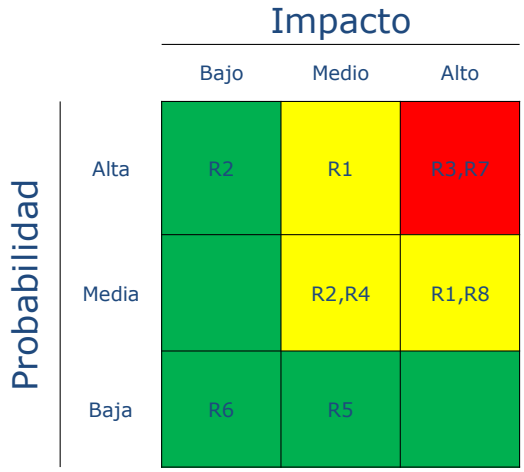

Figura 3.2: Matriz de riesgos

3.2.2. Estimación de costes

Para la realización del presente proyecto, se dispone de una única persona que dispone de 0-3 años de experiencia laboral, por tanto, sería un desarrollador junior. Se estiman unas 300 horas para la realización del mismo. Suponiendo que un programador junior gana unos 19700 euros brutos al año [15], lo que supondría unos 1641.6 euros brutos al mes, y por tanto 9.46 euros la hora. Si se multiplica los euros que se gana cada hora por las 300 horas planificadas saldría un coste total de 2838 euros.

Puesto que el presente proyecto se realiza desde casa no es necesario disponer de una oficina u otro espacio para realizarlo. Tampoco se requiere adquirir materiales adicionales ya que es un

Cod.	Riesgo	Posibles soluciones
R1	El alcance del proyecto no se ajusta a las horas estipuladas y por tanto se excede de las horas planificadas anteriormente, lo que afecta a los tiempos de entrega	En la planificación del proyecto identificar de forma minuciosa las tareas que forman parte de la cadena crítica. Ajustar el alcance desde el comienzo del proyecto a las horas estipuladas dejando un colchón de tiempo para imprevistos (margen de flotabilidad).
R2	La IU del servidor desarrollada no es intuitiva	Creación de prototipos, intervención de usuarios para probar la interfaz, realización de diferentes encuestas de satisfacción.
R3	Android lanza una nueva versión en la que se limite los procesos en segundo plano y esto afecte a la persistencia del malware o a otras funcionalidades	Acotar las versiones objetivo de Android para las que está desarrollado el malware. Dado que la mayoría de dispositivos Android no disponen de la última versión se puede considerar que no es necesario desarrollar para dispositivos con esa última versión. Tener en cuenta como posibles mejoras a posteriori del proyecto. Si se dispone de tiempo suficiente, formación y documentación sobre otros mecanismos para implementar las funcionalidades en esa versión.
R4	No se dispone de la suficiente formación y experiencia en las tecnologías empleadas en el proyecto	Dedicar mayor tiempo a la formación en las materias necesarias para afrontar el proyecto. Adquisición de experiencia mediante la realización en primer lugar de pequeños retos, comenzando por desarrollar las funcionalidades más sencillas y una vez desarrolladas, continuar por funcionalidades más complejas.
R5	Cambio de requisitos durante la ejecución del proyecto	Análisis minucioso y escrupuloso de requisitos. Tener un tiempo extra que sirva como colchón para estos imprevistos.
R6	Cambio no previsto de las tecnologías empleadas	Formación en las nuevas tecnologías empleadas. Tener previsto en el plan de proyecto un colchón de tiempo para afrontar estos imprevistos y utilizarlo si fuera necesario para la formación las nuevas tecnologías a emplear
R7	Gran dificultad en implementar la adquisición de persistencia del malware	Poner énfasis en la formación sobre las diferentes técnicas para implementar dicha funcionalidad.
R8	Una persona involucrada en el proyecto cae enferma y no puede realizar las tareas planificadas	Reducir ligeramente el alcance. Dejar la tarea para más adelante aunque conlleve hacer más horas.

Cuadro 3.2: Riesgos identificados de alto nivel

proyecto de software y este se prueba en máquinas virtuales bajo un PC portátil y una Raspberry Pi para realizar las pruebas oportunas. Tampoco resulta necesario comprar licencias de software ya que las librerías utilizadas y los programas son Open Source.

Por último, se deben tener en cuenta los costes de los equipos donde se ejecutarán las máquinas virtuales. Estos son un ordenador portátil valorado en 1055 euros y una Raspberry Pi 4B con 4G de RAM valorada en 70 euros. Se ha También se debe tener en cuenta su amortización durante la realización del proyecto. Según la Agencia Tributaria [16] la amortización lineal máxima para equipos que procesan información es de un 25 %. Por lo que se puede concluir que el coste del hardware durante la realización del proyecto suponiendo la distribución mencionada de las 300 horas en 4 meses será de 1055*0.25*(4/12) + 70*0.25*(4/12), lo que equivale a 93.75 euros.

En consecuencia los costes finales del proyecto supondrán un total de 2838 euros + 93.75 euros, lo que equivale a 2931.75 euros.

Capítulo 4

Tecnologías utilizadas

4.1. LaTeX y Overleaf

LaTeX es un sistema de composición tipográfica orientado principalmente para la realización de documentación técnica y científica[17]. Es el estándar de facto para la publicación de documentos científicos. Este sistema está disponible como software gratuito. La dificultad de LaTeX puede ser el hecho de que para darle la estructura y el estilo que se desee se deben de emplear distintas sentencias. Sin embargo, cuando se aprende a manejarlo con soltura el potencial que tiene es muy grande. Por ejemplo para escribir ecuaciones, introducir código tiene paquetes específicos para ello. Otros sistemas de composición no contienen funcionalidades tan específicas.

Overleaf es el editor que se usa en el presente trabajo. Este consiste en un editor de LaTeX basado en la nube. Esto significa que no es necesario almacenar el documento en el ordenador. Se asocia con una gran variedad de editoriales de carácter científico para proporcionar plantillas para revistas, papers, trabajos de investigación etc.

4.2. Astah

Astah[18] es un software de modelado de UML utilizado en este proyecto para realizar los diagramas UML de dominio, despliegue y paquetes. Permite realizar, entre otros, diagramas de UML 2.x, diagrama de E/S, diagramas de flujo, CRUD, diagrama de flujo de datos, tablas de requisitos, diagrama de casos de uso etc. Tiene complementos de ingeniería inversa de lenguajes como Java, C# o C++, o viceversa, es decir, de generación de código Java, C#, C++ y PHP. Los diagramas se pueden exportar en documentos HTML y RTF.

4.3. Diagrams.net

Diagrams.net[19], es un software que es Open Source y gratuito que está desarrollado en HTML5. CSS y JavaScript. Permite la creación de todo tipo de diagramas. Por ejemplo: diagramas UML, diagramas de flujo, wireframes, organigramas, árboles, y diagramas de red. En este

trabajo se ha utilizado para realizar en diseño de la base de datos NoSQL de Firebase. Esta base de datos, al no ser relacional, no se modela con el diagrama entidad relación. La base de datos se representa en forma de árbol de nodos. También se ha utilizado para realizar el diagrama de la arquitectura de red.

4.4. Python y Flask

Python[20] es un lenguaje de alto nivel interpretado. Su objetivo es la legibilidad de su código y facilitar la codificación de aplicaciones complejas. Es multiparadigma, es decir, soporta varios paradigmas, como orientación a objetos, imperativo y programación funcional. Soporta varias plataformas lo que permite ejecutar sus programas sobre prácticamente cualquier sistema. Contiene librerías de alto rendimiento específicas para big data, análisis de datos, inteligencia artificial, entre otros. En este caso se utiliza para desplegar una aplicación web para el servidor C&C utilizando el framework de Flask. Flask es un framework desarrollado en Python que facilita el desarrollo de aplicaciones web siguiendo el patrón de diseño Modelo-Vista-Controlador (MVC).

4.5. Bootstrap 5

Bootstrap[21] es una librería que facilita el diseño de aplicaciones web. Se utiliza para la parte front-end de la web del servidor C&C. Permite realizar un diseño de una GUI atractiva e intuitiva para el usuario. Este contiene plantillas de formularios, botones, pop-ups o modals, toast, menús de navegación, entre otros. Está basada en HTML, CSS y JavaScript.

4.6. jQuery

jQuery[22] es una librería multiplataforma basada en JavaScript, que facilita la forma de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y utilizar la técnica AJAX. AJAX permite desarrollar aplicaciones web asíncronas. Esto significa que la parte del cliente de la aplicación puede ejecutarse y mientras en segundo plano comunicarse con el servidor. En este caso se utiliza jQuery para interactuar con el DOM y poder ejecutar correctamente otras librerías que lo requieren, las cuales se detallarán a continuación.

4.7. Datatables.net

Datatables[23] es una librería JavaScript y CSS de tablas que utiliza jQuery. Esta librería permite generar tablas dinámicas y paginadas. Entre sus funcionalidades están: la búsqueda en las tablas que genera, ordenamiento por columnas etc. En este proyecto se utiliza para la implementación de tablas que muestran información de diversa índole, como los bots que componen la botnet, los contactos de un bot, las tareas, el historial de respuestas...

4.8. OpenLayers

OpenLayers[24] es una librería de Javascript de código abierto, que es útil para mostrar mapas interactivos en las aplicaciones web. Esta librería forma parte de los proyectos de Open Source Geospatial Foundation. Permite crear diferentes capas o plantillas en base a un mapa base. En el presente trabajo, se utiliza para representar la ubicación de los bots, dada en coordenadas de latitud y longitud en un mapa interactivo.

4.9. Android Studio y Android SDK

Para desarrollar la aplicación del bot se utiliza el IDE nativo y oficial de Android que es Android Studio. Este permite el desarrollo de aplicaciones Android en distintos lenguajes como Java, Kotlin y C++. En este caso, se utiliza Java para desarrollar la aplicación. Para desarrollar la interfaz de la aplicación se utiliza dentro del mismo IDE, el lenguaje de marcas XML. Para compilar y empaquetar la aplicación se utiliza Gradle. Por otra parte, el Kit de desarrollo de Android incluye un conjunto de herramientas de desarrollo de gran utilidad. Por ejemplo, depurado de código, biblioteca, simulador de teléfono, documentación etc.

4.10. Firebase

Firebase [25] es una plataforma de Google utilizada para el desarrollo tanto de aplicaciones web como de aplicaciones móviles. De Firebase se utiliza la Base de datos en tiempo real, que es NoSQL y está alojada en la nube. Esta base de datos almacena los mismos en formato JSON y permite sincronizar los datos con todos los clientes en tiempo real. Contiene una API REST que facilita la conexión y sincronización con dicha base de datos. En este caso, se utiliza como intermediario entre el servidor C&C y los bots, añadiendo una capa de anonimización y evitando que se pueda revelar la identidad del servidor C&C.

4.11. VirtualBox

Virtualbox [26] es un software utilizado para la virtualización y orientado para arquitecturas x86/amd64. Permite desplegar máquinas virtuales de todo tipo. En este proyecto se utiliza para preparar el entorno de laboratorio que contiene máquinas virtuales Android y la máquina del botmaster. Para las máquinas virtuales Android, como se está en un sistema basado en arquitectura amd64/x86[27], se utiliza un proyecto llamado Android-x86, que contiene imágenes Android hasta la versión 9.0, orientadas a este tipo de arquitecturas de CPU.

4.12. Docker

Docker [28] es un proyecto código abierto que permite la automatización del despliegue de aplicaciones alojadas dentro de contenedores de software. De esta forma se proporciona una capa de

abstracción y de virtualización de las aplicaciones. Esto proporciona mayor seguridad y versatilidad. Por ejemplo si se desea mover una aplicación a otro sistema no hay que preocuparse por las dependencias y software que este necesita para ejecutarse. En este trabajo se ha utilizado para desplegar el servidor C&C en un contenedor.

4.13. Git y Gitlab

Git[29] consiste en un sistema de control de versiones, distribuido, gratuito y de código abierto, que es concebido para manejar todo tipo de proyectos, desde proyectos pequeños hasta proyectos de gran amplitud. Se pueden programar líneas de código basadas en roles, cambiar de contexto sin fricción, entre otras cosas. Lo que diferencia Git de los demás sistemas de control de versiones, principalmente es su modelo de ramificación. GitLab es un software que permite gestionar, administrar, crear y conectar repositorios con diferentes aplicaciones ofreciendo una plataforma en la cual se pueden realizar varias etapas de SDLC/ADLC y DevOps. Los mismos se utilizan en este proyecto para almacenar el código en el repositorio GitLab de la Escuela de Ingeniería Informática.

4.14. Raspberry Pi

Raspberry Pi es un ordenador de bajo coste y en formato compacto, destinado al desarrollo. Su objetivo es hacer accesible la informática a todos los usuarios. Está basada en la arquitectura ARM y es utilizada para la formación sobre informática y electrónica en los colegios, y para desarrollar prototipos. En este caso se utiliza para desplegar el servidor C&C dentro de una red controlada en conjunto con las máquinas virtuales alojadas en un ordenador anfitrión diferente conectado a la misma red. Se ha utilizado para aprender cómo se podría desplegar una aplicación de este tipo, en un contenedor en el sistema empotrado (Ubuntu para este tipo de hardware) que está instalado en la Raspberry Pi.

Capítulo 5

Diseño del software

5.1. Introducción

En el presente apartado se explicará el diseño de la aplicación que engloba la botnet al completo siguiendo la pautas de la ingeniería de software. La aplicación se basará en el modelo cliente/servidor. El servidor será el encargado de enviar las instrucciones correspondientes a los diferentes bots (mando y control). Los clientes serán los distintos dispositivos móviles que reciben y ejecutan las órdenes del servidor. Sin embargo el servidor no se comunicará directamente con los clientes. Será a través de una base de datos distribuida a la que se conectarán los clientes periódicamente para consultar las instrucciones a ejecutar Y, una vez ejecutadas se almacenarán en la misma base de datos las respuestas. Dado que se basa en el modelo cliente/servidor, serán los clientes los que inicien la conexión. El servidor C&C escribirá las instrucciones de cada bot en esta base de datos. Para evitar levantar sospechas esta BBDD será Firebase. Una vez ejecutada la instrucción, almacenarán el resultado de la misma en la base de datos en la que consultaron el comando, utilizando el protocolo HTTP/2 y mediante la propia API que ofrece. Cuando el botmaster se conecta con un dispositivo cliente para gestionar la botnet, este consulta al servidor C&C. El servidor recibe la consulta y la traslada a Firebase. Puesto que es el cliente el que se conecta inicialmente con Firebase, y no al revés, es más sencillo que la conexión pase los filtros de los cortafuegos. Además al ser un servicio legítimo no levantaría sospechas ante los diferentes elementos de monitorización. Para facilitar la administración de los bots, el envío de las instrucciones al bot correspondiente y también conseguir una visualización de las respuestas intuitiva, se realizará una interfaz gráfica de usuario en la aplicación del servidor. Para implementar esta GUI, se desarrollará una aplicación web.

5.2. Análisis de requisitos

5.2.1. Requisitos funcionales

RF001 El sistema constará de un cliente el cual será el dispositivo infectado (bot). Es decir, el bot siempre iniciará la comunicación con el servidor y no al revés. Esto permite la evasión de los

cortafuegos.

RF002 El sistema del cliente deberá ser persistente, es decir deberá funcionar a pesar de los reinicios eventuales del dispositivo. Aunque el usuario cierre la aplicación, esta trabajará en segundo plano.

RF003 El sistema del cliente constará de una interfaz que aparente una funcionalidad que sea útil para el usuario.

RF004 El servidor deberá de disponer de una interfaz gráfica de usuario (GUI)

RF005 La pieza de software del servidor permitirá mostrar los bots infectados

RF006 El sistema deberá de permitir que desde el servidor se manden diferentes instrucciones a los bots que se deseen, pudiendo ser la instrucción dirigida a un bot en concreto o a todos los bots de la botnet.

RF007 Las instrucciones que podrá mandar el servidor a los bots serán al menos las siguientes: Obtención de todos los contactos, geolocalizar en tiempo real el dispositivo, enviar SMS al destinatario con el contenido que se desee, ejecución remota de un comando (RCE) y realizar una petición HTTP a la URL que se desee.

RF008 También la botnet incluirá la funcionalidad de monitorizar las acciones de los bots. Estas pueden ser, abrir una app, capturar pulsaciones del teclado etc.

RF009 El bot no deberá comunicarse en ningún caso directamente con el servidor C&C. En su lugar podrá utilizar servicios en la nube como Twitter, Firebase etc.

5.2.2. Requisitos no funcionales

RNF01 El sistema se basará en el modelo cliente/servidor

RNF02 El sistema deberá ser programado del lado del servidor en lenguaje Python y del lado del cliente en Java utilizando el IDE de Android Studio

RNF03 El sistema deberá utilizar como base de datos distribuida Firebase

RNF04 El tiempo para realizar una tarea por parte del bot no será superior a 10 minutos salvo que este esté apagado

RNF05 La versión objetivo del malware deberá ser superior a Android 7.0 Nougat

RNF06 La versión de Python del servidor deberá ser igual o superior a la versión 3.

5.2.3. Requisitos de información

RI01 El sistema deberá almacenar una lista de todos los bots de la botnet.

RI02 De cada bot se almacenará al menos un identificador único, su fabricante o marca, su modelo y el comando a ejecutar en ese momento.

RI02 De cada comando se deberá almacenar su nombre o etiqueta que deberá ser única y si ha sido ejecutado o no. Además se debe almacenar los parámetros del comando en el caso de que los tuviera. Estos parámetros sirven para especificar ciertos aspectos del comportamiento de esa instrucción. Por ejemplo, si el comando es `SEND_SMS` se deberá pasar como parámetros el número de teléfono al que se desea enviar los SMS y el contenido del SMS.

RI03 El sistema también deberá de almacenar todas las acciones ejecutadas por los bots. De estas acciones, también llamadas respuestas se almacenará un identificador único, la fecha y hora de la acción realizada, el comando que se ha ejecutado y la información que se enviará al servidor en formato JSON. Por ejemplo, en el caso de ejecutar el comando de obtener los contactos, esta información será un array JSON con todos los datos de los contactos.

RI04 El sistema permitirá almacenar el historial de acciones que realice el bot

5.3. Modelo de dominio

En la figura 5.1, se puede apreciar el modelo de dominio del cliente. De este se pueden recalcar 2 clases: la clase `Bot` y la clase `Command`. La clase Bot contiene 4 atributos, un identificador único, el nombre del dispositivo, el fabricante y el modelo. La clase `Command` contiene 3 atributos, el nombre del comando que lo identifica de forma unívoca, un flag que indica si ha sido ejecutado o no y los parámetros. Como el cliente se conecta cada muy poco tiempo, para evitar que ejecute varias veces de forma innecesaria el mismo comando, se introduce este flag. En cuanto el cliente ejecuta un comando y envía la respuesta, establece este flag a 1. El servidor en cuanto manda una instrucción pone este flag a 0 para indicar al bot que lo debe ejecutar. Por otra parte, los parámetros del comando sirven para incluir información adicional al comando. Esta información la debe tener en cuenta el bot al ejecutar la instrucción. Por ejemplo en el caso de enviar un comando que indica que el bot debe enviar un SMS, se debe especificar al menos el destinatario y el contenido del SMS. También se puede visualizar la clase `Response` la cual pertenece solamente al bot y este solamente podrá mandar una respuesta al comando al mismo tiempo. Esta clase tiene 1 atributo que es la respuesta (resp_json) que contendrá un String que representará un objeto JSON.

Por otra parte se pueden observar otras clases. La clase `MainActivity` es la clase que enmarca la lógica de la actividad. Esta clase instanciará otra clase (`OneTimeWorkRequest`) que consistirá en una petición al sistema para iniciar una tarea programada, que se ejecutará con cierta frecuencia. Las tareas programadas permiten ejecutar ciertas acciones con regularidad, en segundo plano,

aun cuando el smartphone esté bloqueado, o esté cerrada la aplicación. La lógica de la tarea programada se encuadra dentro de la clase `GetCommandWorker`. Esta lógica consistiría en consultar en la base de datos de Firebase la instrucción que le corresponda en ese momento, ejecutarla y mandar el resultado de la misma al servidor. La consulta de un dato en Firebase se realiza de forma asíncrona por lo que resulta necesario implementar una interfaz que contenga un método que servirá de callback en el momento de recibir ese dato.

Además se puede apreciar la clase `RebootBroadcastReceiver`, que hereda de una clase predefinida llamada BroadcastReceiver. Esta clase permite al bot seguir recibiendo y ejecutando instrucciones aun reiniciando el mismo, y así adquirir persistencia. Finalmente, otra clase relevante es la clase `KeyloggerService`, que hereda de `AccesibilityService`. Esta implementa la funcionalidad de un servicio de accesibilidad el cual servirá para obtener información de las acciones que se ejecutan en el terminal, así como credenciales, fragmentos de conversaciones y otra información privada.

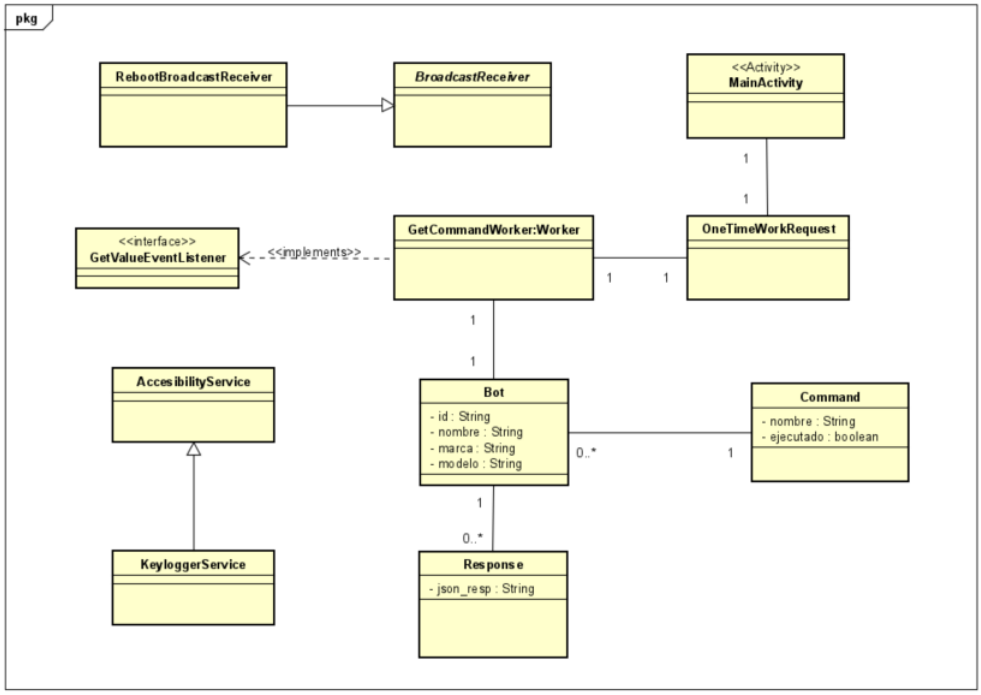

Figura 5.1: Modelo del dominio para el cliente

En la figura 5.2, se puede observar el modelo de dominio para la parte del servidor. Este diagrama contiene 3 clases. Por un lado la clase Bot que contiene los mismos atributos que en la parte del cliente. Lo mismo ocurre con la clase Command. Un bot puede haber ejecutado varios comandos y un comando puede ser ejecutado al mismo tiempo por varios bots. Por otra parte a diferencia del cliente, el diagrama contiene una clase Response que hace referencia a las respuestas de los clientes a las instrucciones enviadas. La respuesta solamente puede ir asociada únicamente a un único comando. Sin embargo un comando puede tener varias respuestas asociadas, por ejemplo cuando un bot ejecuta en diferentes momentos el mismo comando, o cuando el mismo comando

es ejecutado por varios bots diferentes.

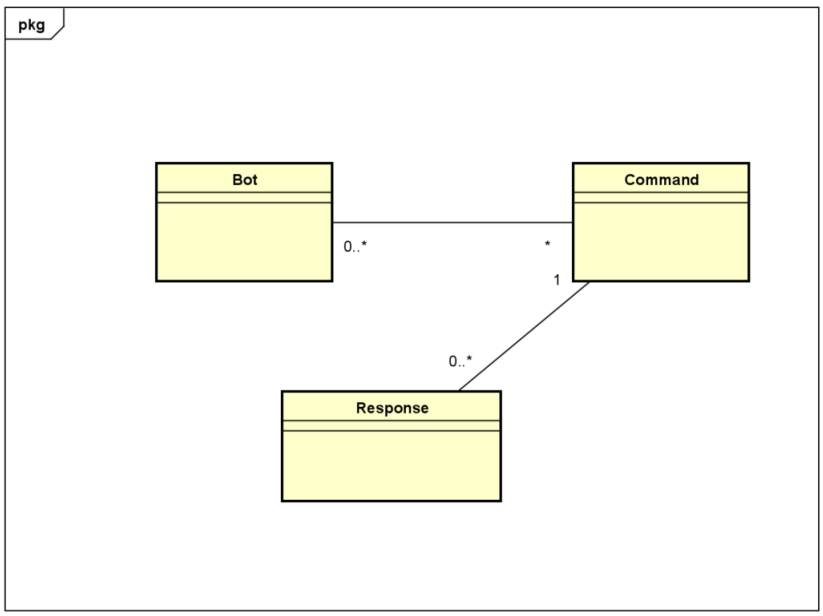

Figura 5.2: Modelo del dominio para el servidor

5.4. Diagrama de paquetes

La aplicación seguirá el diseño arquitectónico por capas. En la figura 5.3 se puede ver el diagrama de paquetes para la parte del cliente. Esta parte se divide en 3 paquetes principales: presentación, dominio y soporte. En el paquete de presentación se agrupan los paquetes con las clases necesarias para mostrar una interfaz. Esta interfaz, en el caso del cliente servirá para evitar que el usuario de la aplicación sospeche de que está realizando otro tipo de tareas. Contiene 2 paquetes: View y Layouts. El paquete Layouts contiene el diseño de la interfaz de la aplicación utilizando el lenguaje de marcas XML. El paquete View, contiene las clases que vienen con el propio framework de Android Studio para dar soporte a la interacción de las vistas con la lógica de la aplicación (las actividades).

El paquete de dominio o capa de negocio contiene el diagrama de dominio antes mencionado. Este paquete encuadra toda la lógica de la aplicación. Esta lógica gira en torno a las actividades. Esta aplicación tendrá solamente una actividad que será la principal (`MainActivity`), es decir la que arrancará al iniciar la aplicación.

Por otra parte el paquete Support contiene otros aspectos de más bajo nivel, como los denominados `BroadcastReceivers`, agrupados en un paquete llamado Receivers, las clases del modelo, agrupadas dentro del paquete Model, que interactúan con la base de datos de Firebase. También contiene un paquete llamado Globals que contiene una clase que almacena todas las constantes estáticas que almacenan ciertos parámetros como la lista de comandos que se pueden ejecutar. Por otra parte, el paquete Support contiene un paquete llamado Services, el cual agrupa los servicios

existentes de la aplicación. En la aplicación solo hay un servicio, que es un servicio de accesibilidad (`KetloggerService`) personalizado que recopila información del bot.

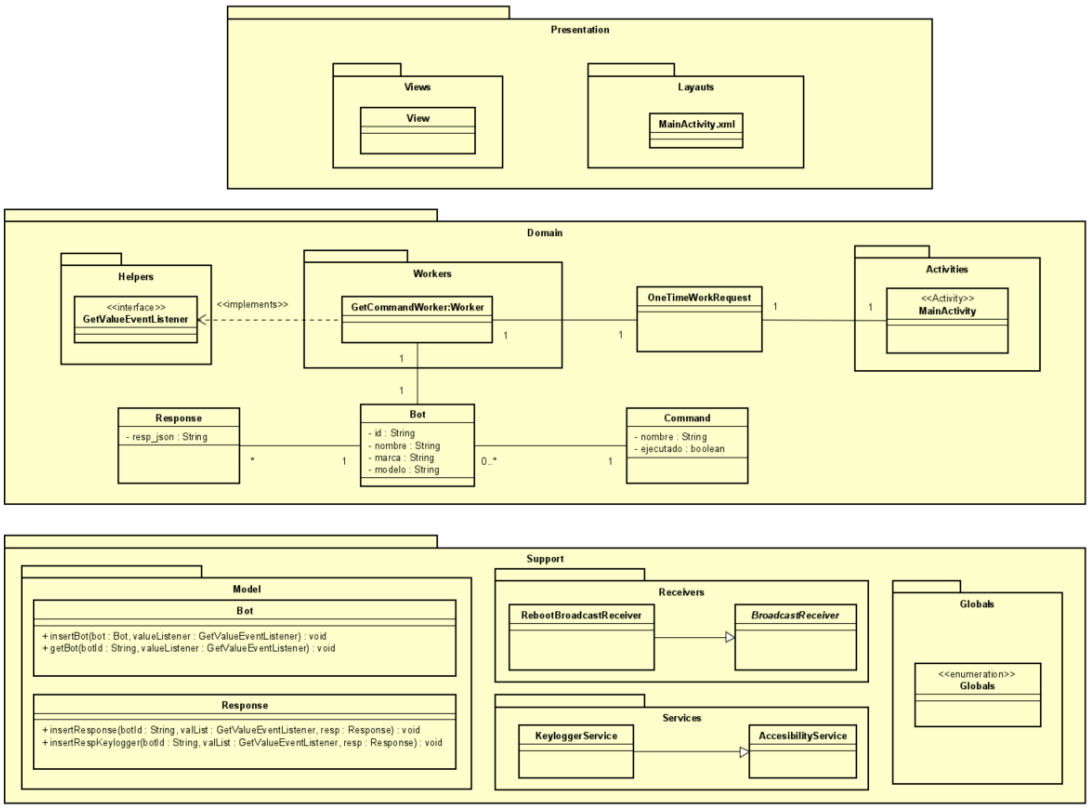

Figura 5.3: Diagrama de paquetes - Cliente

Como se puede ver en la figura 5.4, la parte del servidor se basa en el patrón de diseño de Modelo Vista Controlador (MVC). Por tanto, se divide en 3 paquetes principalmente. El primer paquete, de más alto nivel es el que agrupa las vistas. Las vistas consisten en los ficheros HTML y CSS y JavaScript que encuadran la parte del Font-end de la aplicación, que es la que formará la interfaz gráfica que se mostrará al usuario. Este paquete se subdivide a su vez en 2: Templates, que contiene las plantilla HTML y Static, que contiene los ficheros CSS y JavaScript.

El siguiente paquete sería el controlador, que se encarga de la lógica de la aplicación y también de procesar las peticiones de la vista, y además, realiza las consultas necesarias al modelo y a su vez si es necesario manda los datos consultados del modelo a la vista con el objetivo de que esta se actualice. Este paquete se subdivide en otro paquete llamado server que contiene el único controlador de la aplicación.

A más bajo nivel se encuentra el paquete del modelo que contiene las 3 clases: Bot, Command y Response. Estas clases interactúan con la base de datos de Firebase según para obtener, insertar o actualizar los datos almacenados. Esto permite abstraer al controlador de conocer con qué base de datos conectarse y evitar que este directamente interactúe con la misma. El controlador interactuará con el modelo, no con la base de datos directamente. El paquete modelo contiene

otro paquete llamado Support, que contiene lo necesario para obtener la instancia de la conexión con la base de datos a la que corresponda conectarse.

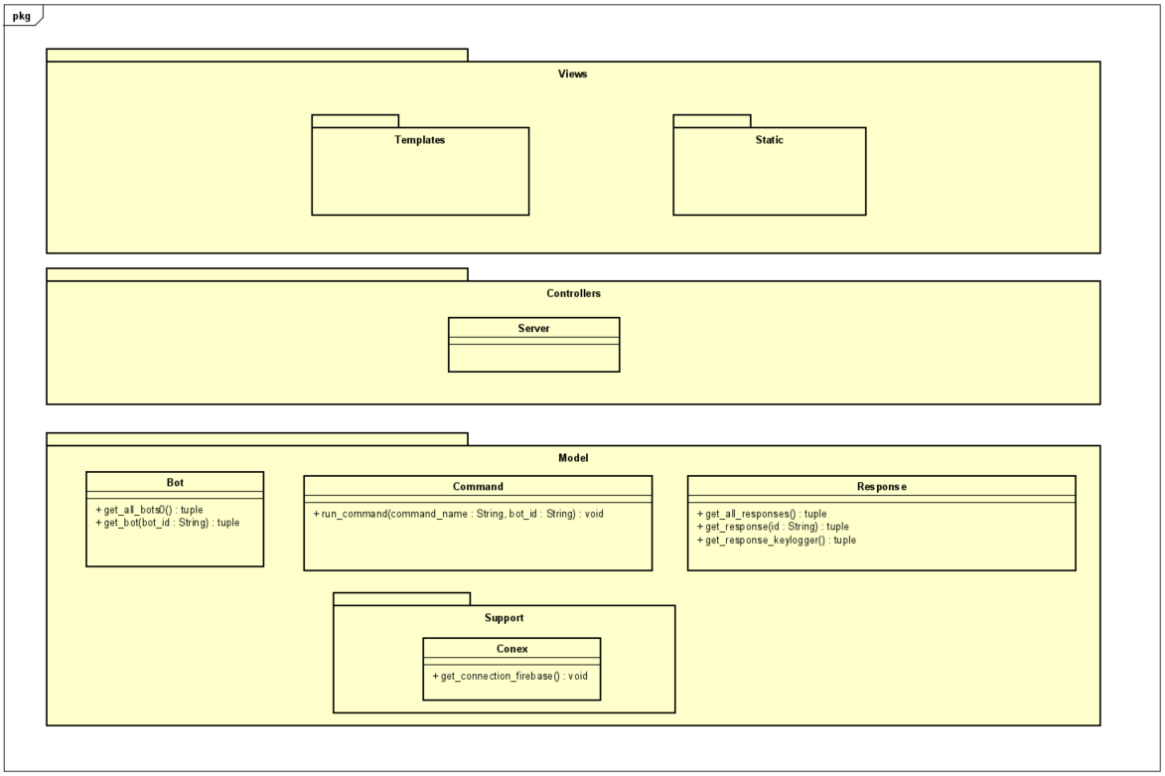

Figura 5.4: Diagrama de paquetes - Servidor

5.5. Modelo de despliegue

El diagrama de despliegue (figura 5.5) consta de 3 dispositivos. Por una parte, el smartphone con sistema operativo Android que sería el bot controlado por el servidor C&C gracias a la aplicación que tendría instalada el bot (BotApp). Esta app está comprendida dentro de la aplicación global la parte del cliente de la misma. Por otra parte tendríamos el dispositivo que albergaría el servidor de Comando y Control (C&C). Este dispositivo tendría instalado el sistema operativo Linux con el entorno de Python instalado sobre el que ejecutará el servidor que desplegará la aplicación web utilizando el framework de Flask. Por otra parte estará el dispositivo del botmaster que es la persona u organización que administra la botnet. Este dispositivo se conectará al servidor web del servidor C&C para consultar las respuestas a los comandos, enviar instrucciones a los bots, consultar información relativa a los bots entre otras acciones. A su vez el servidor se conectará con la base de datos distribuidas de Firebase para realizar las tareas de comando y control que le transmita el botmaster de la forma mencionada y para recopilar información cuando el botmaster se la solicite. Finalmente aunque no sea un dispositivo en sí es necesario incluir el entorno de Cloud en el que está alojado Firebase en el presente diagrama. Este entorno contiene una base de datos en tiempo de ejecución (Real Time DB), la cual es una base de datos NoSQL que almacena los bots y de cada bot el comando que debe ejecutar en ese momento. Cabe recalcar que el servidor de

comando y control y el bot no se comunican directamente, sino que lo hacen mediante Firebase. De esta forma se consigue que sea más complicado localizar al servidor C&C. El bot se comunica con Firebase tanto para obtener el siguiente comando a ejecutar como para almacenar la respuesta en dicha base de datos. El servidor C&C se conectará a Firebase, a través de un proxy por ejemplo o de una VPN o directamente para obtener la información relativa al bot, enviar comandos a los bots o para obtener las respuestas de los bots.

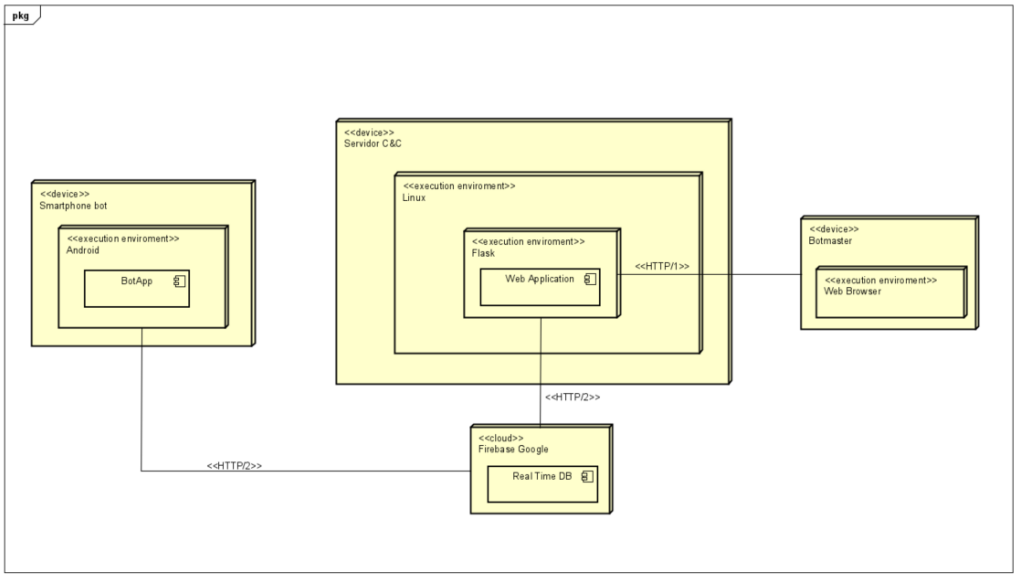

Figura 5.5: Diagrama de despliegue de la aplicación

5.6. Diseño de la base de datos

La figura 5.6 muestra el diseño de la base de datos de la aplicación. Para almacenar todos los datos relativos a la botnet se utiliza base de datos distribuida en la nube llamada Firebase. Esta es una base de datos NoSQL que almacena los datos en forma de árbol JSON [25]. El tipo de base de datos que se utiliza de Firebase es Firebase Realtime Database, que permite que, cuando haya una actualización del contenido de dicha base de datos los dispositivos conectados a ella reciban la actualización el milisegundos. Esto permite que los bots puedan recibir y ejecutar el comando con mayor rapidez. En la figura 5.6, se puede apreciar que no es un diagrama entidad-relación sino que es un árbol con distintos nodos. El nodo raíz es el nodo bot. Este nodo se marca con un corchete ya que contiene a su vez varios hijos de tipo bot. Estos hijos (bots) se indexan mediante una clave única que genera Firebase automáticamente. Esta se la denota como botKey, aunque no tiene un nombre específico. Cada bot contiene diferentes hijos. En primer lugar, contiene el modelo, el nombre y la marca. También contiene un hijo que consiste en el comando a ejecutar. Este a su vez contiene 3 hijos que son si ha sido ejecutado que tomará los valores de 0 o 1, los parámetros que tenga el comando, y el nombre del comando. Otro nodo hijo que contiene es el nodo Responses. Este nodo contendrá varios nodos de tipo respuesta que serán indexados por una clave que se la ha denotado como resp_key. De cada respuesta hay 3 hijos, el comando, la fecha y la respuesta en forma de JSON. La respuesta hace referencia a las respuestas provenientes de los

bots que han ejecutado un comando. Finalmente el nodo Keylogger puede contener varios hijos que representan el historial de acciones que ha realizado el bot. Por ejemplo un click, escribir texto etc. Todas estas acciones, vienen en formato JSON y se almacenan en el nodo resp_json de cada nodo indexado por su eventKey. Esta clave también es generada automáticamente por Firebase y es única.

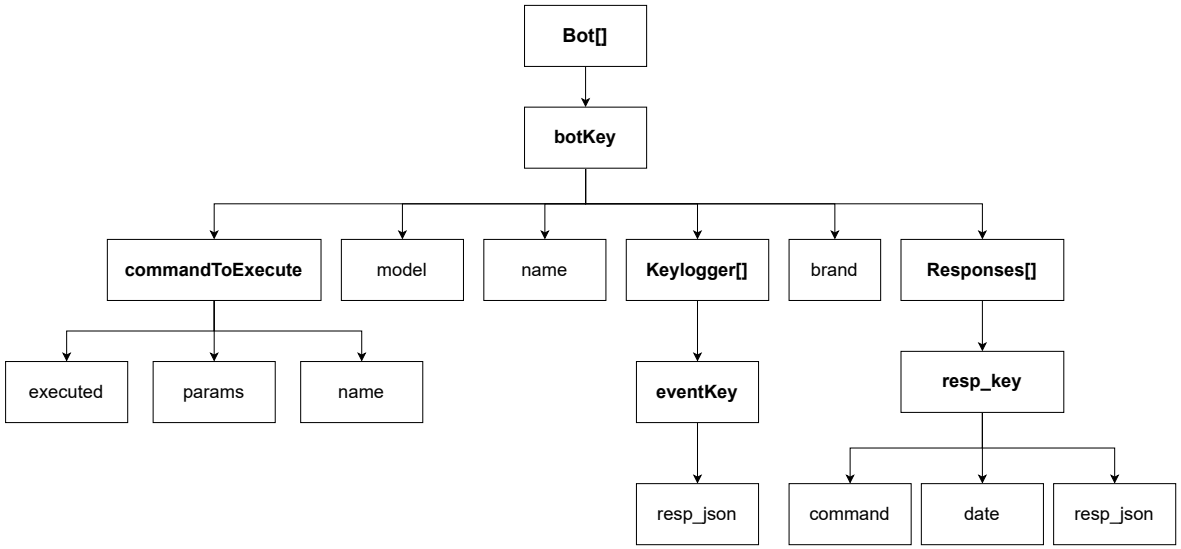

Figura 5.6: Diseño de la base de datos

Capítulo 6

Implementación

6.1. Arquitectura de red y entorno de trabajo

En la figura 6.1 se puede apreciar la arquitectura de red de la botnet. En primer lugar en la parte inferior se aprecian los smartphones y tablets de las víctimas. Cabe recalcar que, tal y como se puede ver representado en la figura, estos smartphones o tablets pueden estar en redes totalmente diferentes. En este caso se representa dos conjuntos de dispositivos móviles víctimas los cuales están en una red que puede ser tanto empresarial como doméstica. Estos están conectados de forma inalámbrica a un punto de acceso (AP01 y AP02). Este punto de acceso estará conectado a su vez a un router y deberá pasar por un cortafuegos. Cabe reseñar que en las redes domésticas el router de la vivienda suele contener estos 3 dispositivos: el punto de acceso inalámbrico, el cortafuegos (normalmente por software), y el router en sí que encamina los paquetes desde/hacia otras redes. Estos dos routers, que ya son los routers del proveedor ISP estarían conectados, tras varios routers más, al core de la red, que es representado en este caso por un solo router.

Por otra parte, se tiene el servidor C&C de comando y control, que está ubicado en otra red totalmente distinta a las redes de las víctimas. Este servidor estará conectado mediante un cable Ethernet a un switch y el switch estará conectado al router y antes pasará también por otro cortafuegos. Otro aspecto que se puede reseñar es que puede ocurrir lo mismo que en las redes de las víctimas, es decir, que las funciones de router, switch y firewall estén integradas en el mismo dispositivo. Por otra parte, se representa el PC de la persona que administra la botnet, el botmaster. Este puede estar en la misma red o en una red distinta a la del servidor C&C. Normalmente, estará conectado en una red distinta como viene representado en la figura.

Finalmente, se representa la base de datos Realtime de Firebase. Esta base de datos hará de intermediaria entre el servidor C&C y los smartphones de las víctimas o bots. De esta forma el servidor C&C será más complicado de detectar ya que las víctimas se conectarán a Firebase, y no al servidor. Además, esto levantará menos sospecha en los sistemas de monitorización (IDS, IPS etc.), y en los cortafuegos que haya, ya que este servicio es aparentemente legítimo y los paquetes que viajan desde/hacia el mismo van cifrados mediante TLS. El servidor C&C interactuará con la

botnet utilizando Firebase, escribiendo en dicha base de datos cuando se desee enviar instrucciones a la botnet y leyendo de la misma cuando se desee obtener información sobre los bots, respuestas a los comandos, etc. El botmaster, cuando desee interactuar con la botnet, se conectará al servidor C&C, a la aplicación web que viene desplegada en él, y este realizará las consultas correspondientes a la base de datos realtime de Firebase.

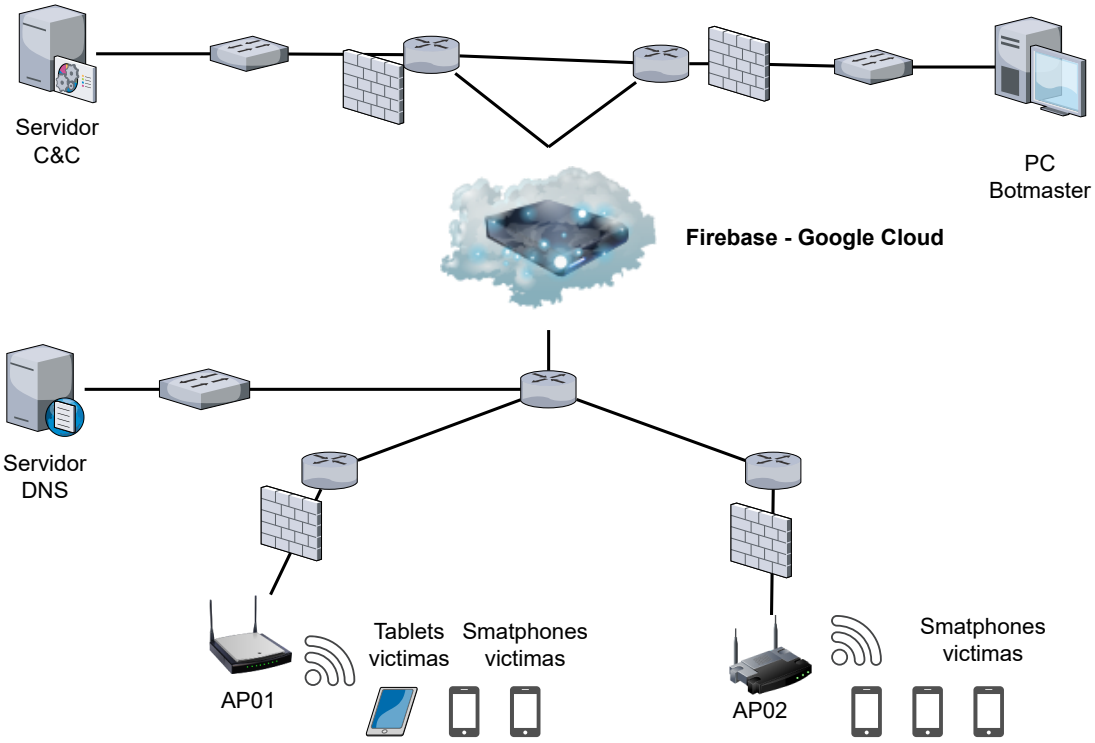

Figura 6.1: Arquitectura de red de la botnet

6.2. Tácticas y técnicas empleadas

A continuación se explicarán las tácticas implementadas y las técnicas usadas para desplegar las mismas en la botnet. La mayoría de estas están basadas en la Matriz ATT&CK del MITRE [13] ya mencionada en el apartado del estado del arte del presente documento. En la mencionada Matriz, las tácticas se denotan como TAxxxx y las técnicas como Txxxxx.

6.2.1. Acceso inicial - TA0027

En primer lugar, el acceso inicial[30] podría llevarse a cabo mediante un enlace enviado por otro bot infectado en el que envíe un enlace para que se descargue la aplicación. Como se tiene acceso a los contactos se envía el SMS a un contacto del bot. Como es probable que el contacto de ese bot le conozca puede que confíe en el mismo para descargarse la aplicación y así infectarse. Esto es lo denominado ingeniería social.

Otra opción, es promocionando dicha aplicación en redes sociales como una aplicación legítima que simplemente mide el estado de la batería del smartphone. También se puede realizar envíos masivos de emails o de SMS para que se descarguen la app.

Otra alternativa, consiste en inyectar código malicioso que permita descargarse la app en un sitio legítimo, por ejemplo, mediante técnicas de Cross-Site-Scripting (XSS). Esta técnica viene etiquetada en la matriz de ataque del MITRE como T1456 y es denominada como Drive-By-Compromise.

Otra manera, algo más elaborada es utilizar la técnica T1458 de la matriz de ataque que consiste en replicar la aplicación que contiene el malware utilizando medios extraíbles como falsos cargadores USB que realmente simulan un periférico como un teclado para ejecutar acciones remotamente en el móvil. Este tipo de dispositivos se denominan comúnmente como rubberducky.

Si se dispone de acceso físico de forma temporal al terminal se puede utilizar la técnica T1461, que consiste en intentar desbloquear el móvil utilizando técnicas de fuerza bruta, entre otras, con el objetivo de descargar e instalar manualmente la aplicación con el malware.

6.2.2. Ejecución - TA0041

La táctica de ejecución[31] consiste en un conjunto de técnicas que dan como resultado que el código controlado por el atacante se ejecute en el dispositivo móvil. Estas técnicas se suelen combinar con otras de otra categoría con el objetivo llevar el mayor número de acciones posibles, así como de robar y obtener el mayor número de información o realizar movimientos laterales.

Se emplea la técnica T1623 que consiste en utilizar los intérpretes de comandos nativos para realizar diferentes acciones y ejecutar el código que se desee. En este caso para implementar dicha técnica se utiliza la clase Runtime, la cual viene instanciada por defecto la aplicación y permite que la aplicación interactúe con el entorno de ejecución. Cabe destacar que la ejecución de comandos en Android está limitada si el dispositivo no está rooteado, de tal forma que se pueden ejecutar solo comandos a nivel de usuario. Se podría intentar aprovechar una vulnerabilidad de versiones anteriores del sistema Android para escalar privilegios.

También se utiliza T1603, que consiste la utilización de tareas programadas con el objetivo de ejecutar el malware de forma recurrente, en este caso para ejecutar el código malicioso de forma recurrente. El código malicioso consiste en consultar con mucha frecuencia el comando a ejecutar, ejecutarlo y enviar el resultado del mismo al servidor de comando y control.

6.2.3. Persistencia - TA0028

Para adquirir persistencia[32], la aplicación utiliza las técnicas de ejecución de tareas programadas etiquetada como T1623.001 y la utilización de broadcast receivers (T1603). La técnica de

ejecución de tareas programadas permite ejecutarse una determinada tarea cada cierto tiempo aun cerrando la aplicación y quitándola de la lista de recientes. La documentación de Android menciona que en algunos dispositivos incluso se siguen ejecutando aun reiniciando el móvil. Para asegurar la ejecución del código malicioso en la mayoría de dispositivos se utilizan varios broadcast receivers que se encarguen de crear la tarea programada una vez se reinicie del terminal.

6.2.4. Movimiento lateral - TA0033

Para realizar movimientos laterales[33] el botmaster llegado un momento podría mandar instrucciones a los bots para que realicen un envío masivo de SMS a todos los contactos de su organización, por ejemplo haciéndose pasar por su jefe. El SMS contendría un enlace para que se descarguen la aplicación y así se infectarían un mayor número de móviles en esa organización.

6.2.5. Recolección de datos - TA0035

Otra de las tácticas empleadas en la botnet es la recopilación de información[34]. Para ello se utilizan distintas técnicas. La primera es la de la geolocalización (Técnica T1430.001), el malware es capaz de geolocalizar de forma precisa el dispositivo utilizando los servicios de Google (GMS). Una vez geolocalizado por instrucción del botmaster, envía las coordenadas de latitud y longitud al servidor de C&C.

Otra técnica empleada es la utilización del servicio de accesibilidad (Accesibility Service)[35] que ofrece Android para facilitar la interacción con el dispositivo a personas que tienen problemas de visión u otros. En este caso se implementa un servicio de accesibilidad que captura la entrada de texto, también permite leer texto y permite capturar otros eventos como abrir una app entre otros muchos. Esto es útil para capturar incluso credenciales ya que sirve de Keylogger. También permite obtener fragmentos de conversaciones de WhatsApp o de otras redes sociales que utilice la víctima. Este servicio permite obtener en tiempo real cada acción que realiza la víctima. Esto abre un abanico muy amplio de posibilidades en el campo de la exfiltración de información.

Las trabas que podría tener esto es la necesidad de declarar explícitamente los permisos necesarios para acceder a la información mencionada, de tal forma que el usuario tenga conocimiento de a qué datos está accediendo la aplicación. Sin embargo muchos usuarios, para ahorrar tiempo, aceptan los permisos sin leer detenidamente cada permiso y sin reflexionar si tiene algún sentido el permiso que se va a conceder. Estos permisos se van mostrando cada vez de una forma más clara según se van liberando nuevas versiones de Android.

6.2.6. Comando y Control - TA00037

En una botnet la funcionalidad de comando y control[36] es fundamental. La principal técnica que utiliza la botnet para realizar las labores de comando y control sería el empleo Web Services externos que son aparentemente legítimos. Estos servicios se utilizan tanto para la lectura de los

comandos del botmaster, como para el almacenamiento de las respuestas a dichos comandos y otra información relativa al bot. En este caso se emplea un servicio de Google llamado Firebase, que ofrece una base de datos distribuida en tiempo real y está basada como se ha mencionado en un árbol de nodos JSON. Esto sirve para hacer creer al dispositivo infectado que está comunicando con un servicio legítimo para obtener o enviar información y así tratar de no llamar la atención ante los firewalls, IDS, IPS u otros sistemas de monitorización.

6.3. Funcionamiento de la botnet

6.3.1. Aplicación cliente

La aplicación diseñada para el sistema operativo Android, consta principalmente de 3 partes: La actividad principal, los Broadcast Receivers y la tarea programada. La aplicación en primer lugar nada más instalarse y abrirse se conectará con la base de datos real time de Firebase y creará un nuevo nodo hijo de Bots, que en adelante se llamará bot. Al crear este nodo, se le asignará automáticamente identificador único, que se denotará como `botID`. Este identificador se guardará en un fichero de configuración de la aplicación (almacenado en `/data/data/com.botapp)`. De tal forma que para la siguiente vez que se conecte la app con la base de datos de Firebase utilizará ese `botID` para leer el comando correspondiente.

Al crear un nuevo nodo bot en la base de datos, la app creará también varios hijos del mismo. El primer hijo será el comando para ejecutar. Este a su vez contiene 3 hijos. Estos son: el comando a ejecutar, si ha sido ejecutado o no y los parámetros del comando. El bot nada más crearse en la base datos asignará a estos valores por defecto. El comando que establecerá será NOP, un comando que indica que no debe realizar ninguna operación, ya que el bot no debe darse órdenes a sí mismo. Este comando lo establecerá como ejecutado, y no le establecerá ningún parámetro. El resto de hijos del nodo bot serán la marca. el modelo y el nombre del dispositivo. El número de serie físico no es posible obtenerlo salvo explotando alguna vulnerabilidad, ya que Android restringe a las aplicaciones habituales obtener dicho número.

Una vez creado el bot, registrará una tarea programada de tipo `OneTimeRequest`, que se ejecutará después de un determinado retardo que será menos de 1 minuto. Esta tarea solo se podrá ejecutar una vez. Sin embargo, justo antes de finalizar esta tarea única, registrará una nueva tarea, también única. Se podría utilizar una tarea de tipo periódica (PeriodicWorkRequest). Sin embargo, el intervalo mínimo de ejecución de la misma es de 900 segundos o 15 minutos[37]. En este caso interesa que se ejecute con más frecuencia, por lo que se utiliza el mecanismo explicado anteriormente.

La tarea programada[38] se ejecutará en segundo plano aun habiendo cerrado y descartado la aplicación de la lista de recientes y en algunos casos aun habiendo reiniciado el móvil, según la documentación para desarrolladores de Android. Esta tarea, se encargará de conectarse a la base

de datos de Firebase y comprobar el comando a ejecutar, y si este no ha sido ejecutado aún, y, en ese caso ejecutarlo. Una vez ejecutado el comando, el bot almacenará la respuesta en la misma base de datos de la que ha obtenido el comando. Esta respuesta, como es variable, se almacena en un String que contiene el objeto JSON serializado. También se almacenará en el nodo correspondiente la fecha en la que se ejecutó el comando y el comando ejecutado. De esta forma, el servidor, cuando consulte a Firebase las respuestas de los bots, sabrá de qué bot proviene y cuál es el comando que ha ejecutado.

Por otra parte, para asegurar la ejecución del malware ADB habiendo reiniciado el terminal, se utilizan broadcast receivers: `android.intent.action.BOOT_COMPLETED`, y `intent.action.REBOOT`. Para asegurar la compatibilidad con dispositivos de marca Xiaomi, se utiliza el segundo broadcast receiver. Cuando se desencadena el evento asociado a un Broadcast determinado, el sistema operativo Android informa del mismo a las aplicaciones que lo han registrado y estas ejecutan una acción determinada. En este caso, una vez recibido el evento de reinicio del sistema, la aplicación comenzará la ejecución de la tarea programada mencionada, que se encarga consultar el comando a ejecutar, ejecutarlo y enviar la respuesta al servidor C&C.

La actividad principal de la aplicación se encargará de realizar las funciones legítimas, que en este caso son las de mostrar el estado de salud de la batería y los distintos parámetros relacionados con la misma. Adicionalmente a esto, creará la instancia correspondiente para ejecutar la primera tarea programada. A partir de ahí, como se ha mencionado, cada tarea programada, al finalizarse registrará una nueva tarea programada única para evitar la ejecución de tareas programadas duplicadas.

6.3.2. Servidor C&C

El servidor se encargará tanto de recibir las respuestas a los comandos ejecutados por los bots, como de indicar a los bots el siguiente comando a ejecutar. Cuando se desee ejecutar un comando, el servidor escribirá en la base de datos de Firebase en el nodo del bot correspondiente el comando deseado, pondrá a 0 el bit de ejecutado y escribirá los parámetros del comando en el caso de que los haya. De esta forma cuando la tarea programada del bot consulte el siguiente comando a ejecutar leerá el comando y al estar el bit de ejecutado a 0 lo ejecutará. Una vez ejecutado, establecerá el bit de ejecutado a 1, y mandará la respuesta al servidor.

El servidor cuando reciba una respuesta, obtendrá el bot del que proviene la misma y el comando ejecutado de la URL y el cuerpo en JSON de la petición POST. Esta información la almacenará en una base de datos mySQL almacenada localmente en el servidor. Para ello insertará una tabla llamada responses un nuevo registro que contiene la información mencionada.

Además de la API REST, el servidor dispone de una interfaz web a la que se conectará el botmaster para dar las instrucciones correspondientes a los bots de la botnet, ver las respuestas a

los comandos ejecutados, recopilar información de los bots etc.

6.4. Funcionalidades implementadas

Las funcionalidades implementadas en la botnet son las siguientes.

6.4.1. Obtención de todos los contactos

La botnet ofrece la posibilidad de obtener todos los contactos de todos los bots o del bot que se desee. Además se puede buscar por nombre, apellido, o un número en concreto, así como ordenarlos según el campo que se desee. Esto se puede ver en la figura.

Para obtener los contactos la aplicación que se instala en el bot requiere el permiso de acceso a los contactos. Cuando se añade la tarea o instrucción de obtener un contacto se escribe `GET_CONTACTS` en el nodo commandToExecute y se marca el flag de ejecutado a 0. Cuando el bot consulta Firebase y decodifica este comando ejecuta la función `getContacts()` que devuelve un objeto de tipo `JSONArray`. Finalmente, una vez obtenidos los contactos la aplicación marca el flag de ejecutado a 1 y almacena en Firebase un JSON que contiene los contactos y que es semejante al de la figura

6.4.2. Geolocalización

Otra funcionalidad es la de geolocalización tanto en primer plano como en segundo plano. Para ello requiere los permisos de geolocalización tanto el primer plano como en segundo plano. También utiliza los permisos de obtención de la geolocalización precisa (`ACCESS_FINE_LOCATION`. Para obtener la localización se utilizan los Google Management Service o (GMS). Cuando se le da la orden de geolocalizar al bot almacena en el nodo Responses la respuesta con la fecha y un JSON que contiene la latitud y la longitud, que sería como el de la figura. El servidor C&C permite visualizar la ubicación de forma intuitiva mediante un mapa interactivo en el cual vendrá marcada la ubicación precisa del bot y la hora a la que se ha obtenido dicha ubicación

6.4.3. Ejecución remota de comandos

La botnet permite la ejecución remota de comandos en los bots. Cabe recalcar que estos comandos se ejecutarán en modo no privilegiado, salvo que el dispositivo estuviera rooteado y se ejecutara el comando `su` antes. Como se ejecutan en modo no privilegiado, esta ejecución está limitada. Además hay numerosas carpetas a las que no se podrá acceder. Si se intentar acceder listando su contenido mediante un comando `ls` por ejemplo la respuesta será una respuesta vacía ya que la salida recogida de los comandos es la salida estándar y no se recoge información sobre la salida estándar de errores o `stderr`. Cuando se envía un comando se añade una nueva tarea cuya instrucción es `EXECUTE_COMMAND`, y se le pasa por parámetro el comando a ejecutar. Cuando el bot

recibe esta instrucción, este obtiene el parámetro que representa el comando a ejecutar, lo ejecuta gracias al `Runtime` que ofrece Android (`ART`)[39]. Este es el entorno de ejecución de aplicaciones, que reemplaza a la anterior máquina virtual de Android llamada Dalvik. Una vez ejecutado el comando se obtiene la respuesta del sistema y se almacena la misma en un objeto JSON como el de la figura.

6.4.4. Envío de SMS

La botnet permite el envío de SMS cuyo origen puede ser un bot o pueden ser todos los bots de la botnet. Para enviar dicho SMS se añade una tarea que contendrá el número al que se desea enviar el mismo, y el mensaje a enviar. Esto permite por ejemplo lanzar campañas de Phishing, de SPAM o de ingeniería social. Para ello, se solicita el permiso de enviar SMS y se refleja este en el manifiest. A la hora de realizar el envío de SMS se crea una instancia de la clase `SmsManager` y envía mediante el método `sendTextMessage`, que ofrece la misma clase. Para informar al servidor C&C que se ha enviado correctamente, una vez realizado el envío almacena un JSON en el nodo Responses de Firebase que pertenece al nodo bot que corresponda. Este JSON contiene información relativa a que se ha enviado correctamente, el destinatario y el contenido del mensaje.

6.4.5. Realización de peticiones HTTP

Otra de las funcionalidades es la realización de peticiones HTTP [40] de forma que el bot actúe como una especie de proxy. Por ejemplo una utilidad de esto es si se desea enviar solicitudes malintencionadas, como solicitudes para realizar inyección de comandos SQL (SQLi) o cross-site-scripting (XSS). Al enviar estas solicitudes desde los bots resulta complicado localizar al atacante ya que la IP reflejada será la de los bots, que son los que realizan la solicitud HTTP.

6.4.6. Exfiltración de información

Gracias al servicio de accesibilidad[35] que ofrece el sistema para facilitar la interacción con el terminal a personas con problemas de visión por ejemplo, se puede obtener una gran cantidad de información. La botnet, si se le concede el permiso de accesibilidad es capaz de obtener las pulsaciones de las teclas (keylogger), los clicks a determinados ítems de las aplicaciones, fragmentos de conversaciones de WhatsApp, Instagram u otras redes sociales. De esta forma se pueden monitorizar de forma exhaustiva las acciones que realiza el usuario en el terminal. Como hace de Keylogger resulta relativamente sencillo averiguar las credenciales ya que se obtiene cada texto escrito. Por ejemplo si la víctima abre la app de su banca online e introduce su contraseña para iniciar sesión, se verá reflejado que el usuario ha abierto la app del banco y ha introducido una contraseña. De esta forma se pueden obtener credenciales bancarias y credenciales de todo tipo. Esto abre un abanico muy amplio de posibilidades en cuanto a exfiltración de información respecta. Si se averiguan las credenciales de cuentas como las de google que tiene vinculadas al terminal se puede obtener una cantidad inmensa de información. Si el usuario concede este permiso este servicio correrá de forma persistente sin que el usuario tenga más conocimiento de ello ya que

estará en segundo plano monitorizando y almacenando en Firebase todas la acciones que realice e información que recopile. A continuación en la figura se muestra un ejemplo breve de lo que puede extraer en tan solo segundos de interacción del usuario con el terminal.

6.4.7. Persistencia

La botnet también tiene la funcionalidad de persistencia, es decir aunque el usuario reinicie el terminal, el malware seguirá corriendo en segundo plano obteniendo información y ejecutando las instrucciones que le mande el botmaster. Todo esto sin que el usuario tenga ninguna constancia de ello. Tampoco los mecanismos de protección de Android detectan específicamente que esta aplicación sea maliciosa ya que la firma de la misma aún no ha sido reconocida y aparentemente realiza una función legítima que es la de supervisar el estado de salud de la batería del terminal.

Para asegurar la persistencia se utilizan diferentes broadcasts receivers[41] en conjunto con las tareas programadas y el servicio de accesibilidad. Los broadcasts receivers utilizados son: `REBOOT`, `BOOT_COMPLETED`, `QUICKBOOT_POWERON`. Se utilizan varios broadcasts receivers ya que algunos terminales no soportan el de `BOOT_COMPLETED`. Por ejemplo hay algunos smartphones de marca Xiaomi[42] a los que se les debe declarar el de `REBOOT`. La documentación de Android menciona que las tareas programadas se pueden ejecutar incluso después de un reinicio, sin embargo se ha podido comprobar que esto no se cumple en todos los smartphones, por lo que se utilizan los broadcast receivers para asegurarse. Cuando el dispositivo se reinicia el sistema envía a las apps que tienen declarados estos broadcast receivers un mensaje de Broadcast, y estas ejecutarán unas determinadas acciones como realizar copias de seguridad, comprobaciones etc. En este caso se lanzará la tarea programada que se encarga de comprobar si hay un comando a ejecutar y si lo hay lo ejecuta. Otro aspecto para destacar es que el servicio de accesibilidad una vez activado de por sí es persistente, por lo que al reiniciar puede seguir monitorizando las acciones del usuario y almacenando información en Firebase para que el servidor C&C pueda consultarla. De esta forma para liberar este malware el usuario deberá específicamente desinstalar completamente dicha aplicación.

6.4.8. Interfaz UI de la aplicación del bot

Para evitar levantar sospechas la aplicación constará de una interfaz gráfica, la cual se puede ver en la figura 6.2. Esta parte de la aplicación realiza acciones legítimas para evitar que el usuario sospeche. Estas son consultar los parámetros de la batería y en función de los mismos indicar el estado de la batería. La batería, es el componente del móvil que menos tarda en fallar. En consecuencia a esto, surge en los usuarios la necesidad de consultar el estado de la batería para saber si deben reemplazarla o si deben tomar medidas para intentar en la medida de lo posible que se acabe estropeando Ante esta necesidad la aplicación dispone de esta interfaz con datos reales sobre diferentes parámetros de la batería y el estado de la misma. Sin embargo una vez instalada y abierta por primera vez esta se convierte en persistente y empieza a ejecutar los comandos enviados por el botmaster, así como recopilar y exfiltrar información.

Figura 6.2: Interfaz funcional de la aplicación que se ejecuta en el bot

6.4.9. Aplicación web para interfaz UI del servidor

Para poder administrar la botnet de una manera intuitiva se desarrolla en el servidor C&C una aplicación web. Para ello se utiliza el framework de Flask. Este framework facilita la codificación de aplicaciones web en Python. Esta aplicación está basada en el modelo-vista-controlador (MVC). El modelo se encarga de interactuar con la base de datos de Firebase. El controlador implementa la lógica del negocio e interactúa con la vista y el modelo. Las vistas están desarrolladas en HTML5, CSS3 y Javascript. Para las vistas o el front-end se emplean las librerías de Bootstrap 5[21] y jQuery[22]. Además para representar las tablas se emplea la librería Datatables[23] y para mostrar un mapa intuitivo se utiliza la librería OpenLayers[24]. Las funcionalidades que se implementan en la aplicación web son las siguientes:

- Consultar los bots existentes en la botnet. En la página principal o en la sección de bots se puede obtener un resumen en forma de tabla que muestra de forma resumida los datos de cada bot que forma la botnet. Estos datos son el modelo, la marca, el identificador del mismo, el comando a ejecutar y si el estado de ejecución del comando. Si se pincha en un

bot se puede obtener información más detallada del mismo, así como el historial de acciones, el cual se detallará en los párrafos posteriores.

- Añadir nuevas tareas a un bot o a todos los bots (enviar instrucciones). Las tareas o instrucciones que se le pueden dar al bot son las siguientes:
  - Obtención de todos los contactos
  - Enviar SMS
  - Realizar una petición HTTP remota
  - Geolocalizar dispositivo
  - Ejecución remota de comandos
- Obtención del historial de acciones de los bots: Conversaciones de WhatsApp u otras redes sociales, textos escritos por la víctima, acciones realizadas por la misma etc. Cuando se pincha en un bot, se visualiza una pantalla que muestra información relativa al bot y también el historial de acciones. El historial de acciones se representa en forma de tabla y contiene la fecha, hora con una precisión de milisegundos, en la que se realizó dicha acción, el tipo de acción (Escribir, click, foco etc.) y la descripción de la acción. Estas acciones, como se ha mencionado, pueden contener información sobre credenciales escritas o cualquier otro texto que haya escrito el usuario, así como fragmentos de conversaciones de WhatsApp, Instagram u otras redes sociales, acciones que realice el usuario (abrir una app, pinchar en una parte concreta de una app, buscar, desinstala app etc.). Es importante que haya precisión con los tiempos a la hora de reconstruir las acciones de los usuarios, sobre todo acciones que se realicen rápidamente como escribir texto.
- Consultar el log de respuestas a los comandos de los bots. Si se consulta la sección de respuestas se puede visualizar una tabla que contiene información del historial de respuestas ordenados por fecha de forma descendente por defecto. La información relativa a las respuestas que muestra la tabla consiste en el identificador de la respuesta, el identificador del bot asociado a esa respuesta, la fecha a la que respondió el bot al comando y el comando respondido.
- Consultar de forma detallada una respuesta a un comando. Cuando se pincha en una respuesta se muestra una interfaz que muestra de forma detallada la misma. Esta interfaz varía en función del tipo de comando al que hace referencia la respuesta.
  - Respuesta a la obtención de los contactos. Se muestra una tabla en la que se pueden visualizar los contactos ordenados por orden alfabético y que permite realizar una búsqueda por nombre, apellido o número de teléfono del contacto.
  - Respuesta a la geolocalización. La aplicación representa un mapa interactivo en el cuál vienen marcado el lugar donde se localizaba el bot en el momento de ejecutar el comando. Además se muestra la fecha y la hora exacta a la que se obtuvo la localización del mismo.

- Respuesta a la petición HTTP remota. La aplicación permite visualizar de forma gráfica y consultar el código de respuesta del servidor HTTP al que se está haciendo la consulta utilizando como intermediario el bot. También se muestra la fecha y hora en la que se obtuvo la respuesta del servidor.
- Respuesta a la ejecución remota de comandos. Se muestra la respuesta al comando en texto plano con unos colores y una tipografía semejante a la de una terminal.

En el Anexo, en la sección de manual de usuario se explica en mayor detalle las funcionalidades que ofrece la aplicación web del servidor de Comando y Control.

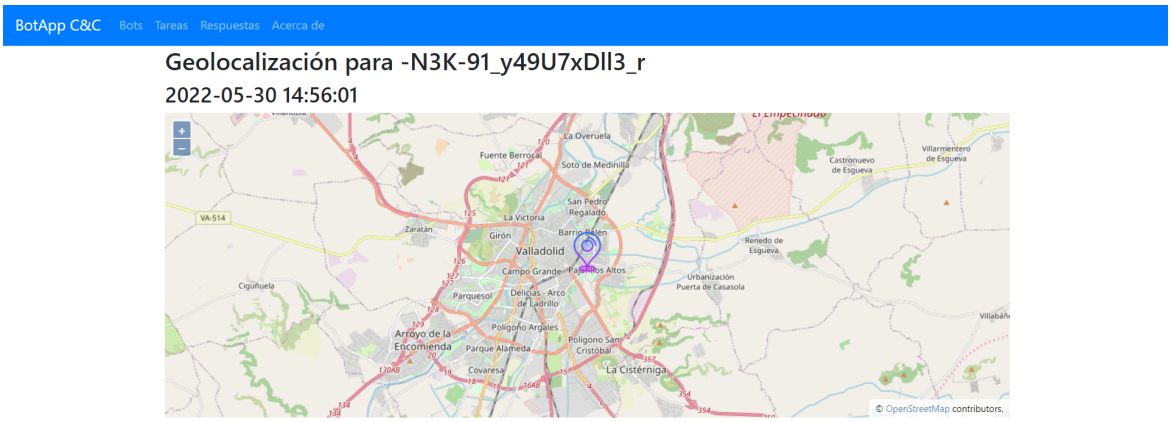

Figura 6.3: Servidor C&C - Interfaz GUI

6.5. Despliegue de la botnet

En primer lugar se debe obtener el código fuente de la botnet, tanto el de los bots como el del servidor de comando y control. Para ello, se clona el repositorio donde está alojado mediante el comando:

```
#git clone https://gitlab.inf.uva.es/carmaci/tfg-botnet.git
```

Una vez clonado el directorio entra en el directorio `tfg-botnet` y se lista su contenido.

```
#cd tfg-botnet
#ls -la
total 28
drwxr-xr-x 5 root root 4096 Jun 10 14:24 .
drwxr-xr-x 3 root root 4096 Jun 10 14:23 ..
drwxr-xr-x 8 root root 4096 Jun 10 14:25 .git
drwxr-xr-x 3 root root 4096 Jun 10 14:24 Client
```

```
-rw-r--r-- 1 root root 6289 Jun 10 14:23 README.md
drwxr-xr-x 5 root root 4096 Jun 10 14:24 Server
```

Al listar el contenido del directorio se obtendrán 3 carpetas: `.git`, el cliente y el servidor, y el fichero `README.md`.

6.5.1. Despliegue del bot

En el directorio `Client/` se encuentra el código fuente de la aplicación del bot que contiene el malware. Para instalar la aplicación en el bot se debe compilar la aplicación y empaquetarla en un fichero `APK`. Esto se puede realizar de varias formas.

Se puede realizar a través del IDE de `AndroidStudio`, el cual se puede descargar en el enlace `https://developer.android.com/studio`. Una vez descargado e instalado se puede importar el proyecto pinchando en `File >Open` y seleccionando la carpeta `Client`, que es la que contiene el proyecto. Una vez importado correctamente, se genera el APK pinchando en `Build >Build Bundle(s)/ APK(s) >Build APK(s)`.

Otra alternativa de compilación es irse al subdirectorio `Botcalc` de `Client` y ejecutar el script bat `gradlew.bat` en el caso de Windows o el ejecutable `gradlew` en el caso de Linux. Para indicarle al script/ejecutable que debe compilar la aplicación y empaquetarla en un fichero APK se ejecuta de la siguiente manera:

```
#gradlew assembleDebug
```

Al ejecutar este comando se generará un fichero APK que se podrá visualizar accediendo a la siguiente ruta:

```
#cd Client/Botcalc/app/build/outputs/apk/
```

Una vez generado el APK, solo quedaría pasar el mismo a la víctima para que se lo instale. Algo a recalcar, es que método de infección es local y no remoto, es decir, es necesario que se instale la aplicación en el terminal de forma física. Sin embargo, hay diferentes formas de instalar la aplicación sin tocar el terminal. Una de ellas es solamente conectándolo por USB. Por ejemplo, si se dispone de un falso cargador que sea un rubber-ducky se podría instalar la aplicación mediante comandos `adb`. Un rubber-ducky, es dispositivo que simula ser un teclado y con el que se puede tener el control de forma física del móvil. También si se dispone de un ordenador que esté autorizado para ejecutar comandos `ADB` en el móvil se podría instalar. Para instalar la aplicación mediante ADB (Android Debug Bridge) se ejecutaría el siguiente comando.

```
#adb install <fichero apk>
```

6.5.2. Despliegue del servidor C&C

Si se accede al directorio `Server/` se podrá apreciar el código fuente del servidor. Para desplegarlo hay varias opciones. Una es desplegarlo en un contendor Docker y otra es desplegarlo sin contenedor utilizando `Python` y `Pip`.

Despliegue en un contenedor Docker

Para desplegarlo como un contenedor debe de estar instalado Docker[28]. Una vez instalado basta con generar la imagen y lanzarla sin preocuparse de instalar dependencias. Para generar la imagen con el tag de `py:botnet-server`, una vez dentro del directorio `Server/` se ejecutaría:

```
#docker build -t py:botnet-server .
```

Una vez generada se lanzaría la misma en el puerto xxxx mediante el comando:

```
#docker run -p 5000:xxxx py:botnet-server
```

Despliegue sin contenedor

Para realizar el despliegue sin el contenedor, en el caso de que no se disponga de Python y de Pip, se deben instalar. Para ello se ejecutaría (en Ubuntu o semejantes):

```
#sudo apt-get update
#sudo apt install python3-pip
```

Una vez instalado Python y Pip se instalarían las librerías y requerimientos necesarios mediante:

```
#pip -r install requirements.txt
```

Posteriormente se lanzaría el servidor:

```
#python3 server.py
```

Lanzar el servidor de forma persistente

Para lanzar el servidor de forma persistente, si no se ha desplegado como contenedor, hay varias alternativas. En este caso, se propone crear un nuevo servicio utilizando Systemd [43] para el servidor C&C. Para ello se debe abrir un bloc de notas y pegar el siguiente contenido y guardarlo como `botnet_server.service`.

```
[Unit]
Description=BotnetCyC
After=botnet_server.target
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=always
RestartSec=1
User=<usuario>
ExecStart=/usr/bin/env python3 /ruta/a/tfg-botnet/Server/server.py

[Install]
WantedBy=multi-user.target
```

Una vez guardado, se copiaría al directorio `/etc/systemd/system/` mediante el comando:

```
#sudo cp botnet_server.service /etc/systemd/system/
```

Para iniciarlo y conseguir que se lance al arranque del sistema, se ejecutaría lo siguiente:

```
#systemctl enable botnet_server
#systemctl start botnet_server
```

Capítulo 7

Pruebas

En el presenta apartado se expondrán las diferentes pruebas realizadas, con el objetivo de comprobar el correcto funcionamiento de la botnet. En resumen, se probará la instalación del malware en distintos bots (acceso inicial), el envío de todos los tipos de instrucciones tanto a cada bot individualmente como a todos los bots en conjunto. Se comprobará también los casos de cuando un bot no está disponible por cualquier circunstancia. Además, se pasará la aplicación maliciosa que tiene instalada el bot por distintos antivirus. También se probarán los casos en los que los usuarios no autoricen o autoricen parcialmente los permisos requeridos.

P01	**Acceso inicial**
Descripción	Se descarga, instala y ejecuta por primera vez la app en la máquina virtual Android que está encuadrada dentro del entorno del laboratorio
Resultado esperado	Se crea un nuevo nodo bot que se verá reflejado tanto el Firebase como en la interfaz web en la sección de bot. Este bot aparecerá un comando inicial llamado NOP (No operación) y que ya será ejecutado.
Resultado real	Se ve reflejado el nuevo nodo bot en Firebase y se puede apreciar en la interfaz de la aplicación web

Cuadro 7.1: Acceso Inicial

P02	**Monitorización de las acciones**
Descripción	El usuario autoriza los permisos de accesibilidad y se van monitorizando sus acciones y extrayendo información. En este caso las acciones son: abrir WhatsApp, abrir una conversación y escribir el mensaje de Holaa. Luego abrir el navegador, acceder al WebMail de la uva y escribir unas credenciales ficticias.
Resultado esperado	En la aplicación web, al pinchar sobre el bot se van mostrando los diferentes eventos que realiza el usuario. Tanto los eventos de click en los que abre la app de WhatsApp o la del navegador y accede al WebMail de la Universidad de Valladolid, como los eventos de escritura de texto. En estos eventos se ven reflejado todo texto que escriba de forma ordenada cronológicamente.
Resultado real	En la aplicación web, al pinchar sobre el bot se van mostrando los diferentes eventos que realiza el usuario. Se muestra la información mencionada tras realizar dichas acciones en el móvil. En estos eventos se ven reflejado todo texto que escriba de forma ordenada cronológicamente

Cuadro 7.2: Monitorización de las acciones

P03	**Acceder a la página de tareas**
Descripción	Se accede a la sección de la aplicación web de tareas
Resultado esperado	Se muestra una tabla de todas las tareas, una por cada bot. En esta tabla se deben visualizar si está ESPERANDO a ejecutar esa tarea o si ha sido ejecutada (OK). También debe aparecer el bot que tiene asignado esa tarea y los parámetros si los tiene.
Resultado real	Se muestra una tabla de todas las tareas, una por cada bot. En esta tabla se deben visualizar si está ESPERANDO a ejecutar esa tarea o si ha sido ejecutada (OK). También debe aparecer el bot que tiene asignado esa tarea y los parámetros si los tiene.

Cuadro 7.3: Acceder a la página de tareas

P04	**Añadir una nueva tarea para un bot encendido**
Descripción	Se añade una nueva tarea para un bot existente de la botnet que esté encendido
Resultado esperado	En cuanto sea añadida se debe visualizar tanto en Firebase en el nodo `commandToExecute`, como en la aplicación web la nueva tarea o comando añadido que reemplazará al anterior, sus parámetros, y el estado que será ESPERANDO hasta que la ejecute el bot. En un tiempo de unos 20 segundos como máximo, si el bot está encendido, la ejecutará y el estado de la tarea cambiará a OK indicando que se ha ejecutado
Resultado real	Se añade la tare

Cuadro 7.4: Añadir nueva tarea para un bot encendido

P05	**Añadir una tarea en un bot apagado**
Descripción	Se añade una nueva tarea o comando en un bot que está apagado
Resultado esperado	Ocurre lo mismo que en la prueba anterior, a excepción de que la tarea no estará en estado `ESPERANDO` hasta un máximo de 20 segundos, sino que seguirá esperando hasta que el bot se encienda.
Resultado real	Ocurre lo mismo que en la prueba anterior, a excepción de que la tarea no estará en estado `ESPERANDO` hasta un máximo de 20 segundos, sino que seguirá esperando hasta que el bot se encienda.

Cuadro 7.5: Añadir una nueva tarea en un bot apagado

P06	**Añadir una tarea con parámetros requeridos vacíos**
Descripción	Se intenta añadir una tarea que requiere parámetros (Por ejemplo, ejecutar un comando que requiere el comando a ejecutar), y estos no se rellenan.
Resultado esperado	Marcar en rojo los campos que no se han rellenado y evitar la creación de la nueva tarea.
Resultado real	Se marcan en rojo los campos que no se han rellenado y se evita la creación de la nueva tarea.

Cuadro 7.6: Añadir nueva tarea con los parámetros requeridos vacíos

P07	**Prueba de persistencia**
Descripción	Se reinicia el bot y una vez reiniciado se le ordena que realice una tarea.
Resultado esperado	Se deberá de ver reflejada la respuesta a dicha tarea en un máximo de 1 minuto después de que haya terminado el proceso de encendido.
Resultado real	El bot una vez reiniciado responde correctamente a las tareas que se le encomienda realizar.

Cuadro 7.7: Prueba de persistencia

P08	**Acceder a un bot o a una respuesta que no existe**
Descripción	Se intenta acceder a un bot o una respuesta que no exista mediante la ruta `/responses/<id_bot>/id_respuesta` o `/bots/<id_bot>`. A estas rutas se les pasa un `id_bot` o un `id_respuesta` que no exista
Resultado esperado	El servidor debería mostrar una web que indique que no existe el bot o la respuesta indicada.
Resultado real	El servidor muestra una web que contiene una mensaje que informa de que el bot o la respuesta indicados no existen.

Cuadro 7.8: Acceder a un bot o a una respuesta que no existe

P09	**Geolocalizar dispositivo**
Descripción	Se le indica al bot que debe de obtener su geolocalización para que la pueda visualizar el botmaster.
Resultado esperado	El bot ejecuta la tarea y al cabo de unos segundos al refrescar la sección de respuestas de la web se debe visualizar una nueva respuesta. Al pinchar en ella debe salir un mapa interactivo en el que se indique la localización del bot. En las máquinas virtuales no se debería poder obtener la localización real ya que no disponen de chip GPS.
Resultado real	El bot ejecuta la tarea encomendada y se visualiza en la sección de respuestas la nueva respuesta. Al pinchar en ella se visualiza un mapa interactivo con la localización del bot. En las máquinas virtuales no se debería poder obtener la localización real ya que no disponen de chip GPS.

Cuadro 7.9: Acceder a un bot o a una respuesta que no existe

P10	**Obtener todos los contactos del dispositivo**
Descripción	Se le ordena a un bot que debe obtener todos sus contactos.
Resultado esperado	El bot debe obtener todos sus contactos (nombre, apellidos y números) y en la sección de respuestas de la web servidor C&C se debería de apreciar una nueva respuesta. Si se pincha en ella se debe visualizar una tabla con los contactos ordenados alfabéticamente.
Resultado real	El bot obtiene todos sus contactos y en la sección de respuestas se aprecia una nueva. Al pinchar en la misma se muestra otra web cuyo contenido es una tabla con todos los contactos del terminal.

Cuadro 7.10: Acceder a un bot o a una respuesta que no existe

P11	**Ejecutar un comando en el terminal**
Descripción	Se le ordena a uno de los bots la ejecución de un comando remoto. Se prueban `ls`, `pwd`, `ifconfig`, `ping -c 1 google.es`, `curl google.es`, `ps`, `su`, `ls /data/data`
Resultado esperado	El bot debe ejecutar correctamente los comandos que no conllevan tener permisos de root. Al ejecutar los comandos que conllevan permisos de root no se debe obtener ninguna respuesta por la salida estándar ya que se mostrará el error de acceso denegado por la salida estándar de errores. La respuesta a los mismos se podrá obtener de la misma forma que en las dos pruebas anteriores.
Resultado real	El bot ejecuta correctamente los comandos que no conllevan tener permisos de root. Al ejecutar los comandos que conllevan permisos de root no se obtendrá ninguna respuesta por la salida estándar ya que se mostrará el error de acceso denegado por la salida estándar de errores. La respuesta a los mismos se podrá obtener de la misma forma que en las dos pruebas anteriores.

Cuadro 7.11: Acceder a un bot o a una respuesta que no existe

P12	**Enviar un SMS con un enlace a un número**
Descripción	Se le indica al bot que debe ejecutar la instrucción de envío de un SMS cuyo contenido se especifica a un número en concreto
Resultado esperado	El bot debe enviar el SMS al destinatario indicado (si es de España sin prefijo internacional) con el contenido especificado. Se deberá visualizar en la respuesta que el bot ha realizado el envío correctamente. Este tipo de comando no funcionará en una máquina virtual ya que no dispone de una tarjeta SIM física.
Resultado real	El bot envía el SMS al destinatario indicado (si es de España sin prefijo internacional) con el contenido especificado. En la respuesta del bot se visualizará que este ha realizado el envío correctamente. Este tipo de comando no funciona en una máquina virtual ya que no dispone de una tarjeta SIM física.

Cuadro 7.12: Acceder a un bot o a una respuesta que no existe

P13	**Realizar petición HTTP desde el dispositivo víctima**
Descripción	Se realiza una petición HTTP GET a una URL especificada sin indicar Cookies. Esta petición se realiza a un servidor web de ejemplo en el que queda registrado en los logs la IP de quien accede a algún recurso del mismo
Resultado esperado	El bot debe realizar la petición HTTP y en la aplicación web del servidor C&C se podrá visualizar la representación de la web que proporciona como respuesta el servidor de prueba. También se debería ver el código que viene incluido en la misma respuesta del servidor. Por otra parte en el servidor de prueba se apreciaría que la IP del dispositivo que ha accedido es la del bot y no la del servidor C&C.
Resultado real	El bot realiza la petición HTTP y en la aplicación web del servidor C&C se podrá visualizar la representación de la web que proporciona como respuesta el servidor de prueba. También se podrá ver el código que viene incluido en la misma respuesta del servidor. Por otra parte en el servidor de prueba se aprecia que la IP del dispositivo que ha accedido es la del bot y no la del servidor C&C.

Cuadro 7.13: Acceder a un bot o a una respuesta que no existe

P14	**Añadir una nueva tarea o comando del que el bot no disponga los suficientes permisos**
Descripción	Se le ordena a un bot una instrucción para la cual el malware no dispone de los permisos suficientes ya que el usuario del dispositivo móvil no se los ha otorgado.
Resultado esperado	La tarea se quedará en estado esperando de forma indefinida hasta que el usuario otorgue los permisos correspondientes o se le indicara al bot otra tarea a realizar.
Resultado real	La tarea se queda en estado esperando de forma indefinida hasta que el usuario otorgue los permisos correspondientes o se le indique al bot otra tarea a realizar.

Cuadro 7.14: Acceder a un bot o a una respuesta que no existe

P15	**Ejecución de las instrucciones en todos los bots**
Descripción	Se ejecutan todos los comandos disponibles en todos los bots seleccionando la opción de todos los bots en el desplegable en el que se indica el bot en el que se desea ejecutar la instrucción.
Resultado esperado	Ejecución de la tarea en todos los bots encendidos sin necesidad que la app esté abierta
Resultado real	Los distintos comandos según se van enviando se van ejecutando en todos los bots encendidos. En los que están apagados o no disponibles salen en estado de `ESPERANDO...` hasta que estos estén disponibles.

Cuadro 7.15: Prueba de detección de antivirus

P16	**Prueba de detección del malware por parte de los antivirus**
Descripción	Se sube el fichero APK a plataformas que pasan diversos antivirus como Virus total.
Resultado esperado	Detección por parte de al menos 40 % de los antivirus
Resultado real	Solamente 1 de 61 antivirus ha detectado la aplicación como un malware. El antivirus que lo ha detectado ha sido Avast Mobile. También al probarlo con el antivirus de Xiaomi lo detecta como un Troyano.

Cuadro 7.16: Prueba de detección de antivirus

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

Para concluir se exponen los que objetivos han sido completados fructíferamente:

- Se ha implementado una botnet en la que los bots se comunican anónimamente con el servidor C&C mediante la utilización de los servicios de almacenamiento en la nube que ofrece Firebase.
- Se ha investigado y puesto en práctica las distintas técnicas utilizadas por este tipo de malware. Muchas de ellas vienen expuestas en la Matriz ATT&CK del MITRE. Al conocer este tipo de técnicas, se pueden establecer distintos patrones que utilizan este tipo de malware con el fin de mejorar la detección de los mismos por parte de los antivirus de los usuarios.
- También se ha configurado un entorno de laboratorio concreto específico para este proyecto, con máquinas virtuales Android.
- Además se ha estudiado detenidamente la arquitectura de seguridad de Android y los distintos mecanismos que utiliza el mismo sistema para salvaguardar la integridad, disponibilidad y confidencialidad de la información que está almacenada en el terminal.
- Se ha desarrollado una aplicación web en Python utilizando un framework que sirve para administrar la botnet de forma intuitiva.

A nivel personal, considero que he cumplido con los siguientes objetivos:

- He comprendido en profundidad el funcionamiento de una botnet.
- He desarrollado los conocimientos sobre malware persistente, en especial en sistemas móviles.
- He potenciado los conocimientos existentes sobre programación de aplicaciones Android. He adquirido nuevos conocimientos avanzados de desarrollo de apps para Android. Esto incluye aprendizaje sobre broadcast receivers, tareas programadas, servicios de accesibilidad...
- He mejorado mis habilidades con el lenguaje Python y he aprendido un nuevo framework.

- Ha crecido mi interés sobre los malware persistentes, las botnets, vulnerabilidades en sistemas móviles y sobre la ciberseguridad en general.

8.2. Trabajo futuro

Se plantean las siguientes líneas de trabajo para un futuro:

- Mejorar las técnicas para evadir los antivirus y también evitar exponer partes concretas del código si alguien realiza ingeniería inversa. Ciertos valores con la conexión de con Firebase se pueden cifrar con el objetivo de evitar que alguien que realice ingeniería inversa los pueda detectar. Pese a que muchos smartphones no detectan este malware, los que tienen instaladas versiones más recientes, especialmente los que llevan incorporados antivirus más sofisticados, detectan la aplicación como un Troyano o como un Spyware,

- Explotación de vulnerabilidades para escalar privilegios. El entorno de Android está restringido debido a que el usuario no tiene acceso a un usuario privilegiado salvo que le realice específicamente el proceso de «rooteo». En consecuencia, el malware no puede acceder a otros aspectos más avanzados del sistema ni ejecutar comandos privilegiados. Se propone incluir al malware ciertos exploits que se aprovechen de alguna vulnerabilidad existente en versiones de Android antiguas que permite la escalada de privilegios.

- Añadir más comandos al bot. Se pueden añadir gran multitud de comandos y por tanto enriquecer la funcionalidad de la botnet. Por ejemplo: realizar llamadas, leer SMS, hacer fotografías, grabar audio etc.

- Añadir más capas de anonimización a la botnet. En la botnet implementado hay una capa de anonimización, que es Firebase, que hace de intermediario entre el servidor C&C y los bots, para que estos no se comuniquen directamente con el servidor C&C. Se pueden añadir más capas de anonimización utilizando proxies.

- Convertir la botnet de centralizada a una botnet P2P. Las botnets P2P han demostrado ser de las más peligrosas debido a que no hay forma de saber quién es el servidor de comando y control. Basta con introducir el comando en uno de los bots, y este se propaga por toda la botnet. Si se desea llevar la anonimización a otro nivel se podría transformar la botnet en una botnet P2P. Para ello, los bots llevarían incorporada la funcionalidad de servidor C&C.

Capítulo 9

Anexos

9.1. Estructura del código

El código se divide en dos partes: cliente y servidor. La parte del cliente consiste en la aplicación móvil que contiene el malware, la cual se instalará en los distintos dispositivos móviles de las víctimas. La parte del servidor se encarga de realizar las labores de comando y control. Esta parte contiene la aplicación web que utilizará el botmaster para interactuar con la botnet.

En la figura 9.1 se puede apreciar la estructura del código del servidor. Esta es más sencilla de que la de la app cliente que contiene el malware como se podrá ver posteriormente. El servidor consta en un primer nivel de un fichero `DockerFile` que es el encargado de configurar la imagen docker por si se desea «dockerizar». También se puede ver el fichero `requirements.txt`, el cual contiene las librerías Python necesarias para que funcione el servidor correctamente. También se ve el fichero `server.py`, el cual contiene el código necesario para desplegar el backend de la aplicación web. Se podría decir que es el controlador de la aplicación web. Por otra parte, se puede apreciar el directorio `model` con 3 ficheros: `bot.py`, `command.py` y `response.py`. Esta parte del código es el modelo, por lo que es la encargada de interactuar con la base de datos, en este caso con Firebase. De esta forma se logra una abstracción en la que el controlador (`server.py`) desconoce la forma de acceso a la base de datos, ya que de esto se encarga el modelo. Finalmente, se pueden visualizar 2 carpetas que son `static` y `templates`. La primera, contiene los ficheros estáticos de la web, como imágenes, CSS y otros recursos multimedia. La segunda, contiene las plantillas html. Todas las plantillas heredan de la plantilla `base.html`. El contenido de esta es la cabecera de la web y el «footer» que tienen en común todas las plantillas. El controlador (`server.py`), en función de una acción u otra, devolverá una plantilla diferente.

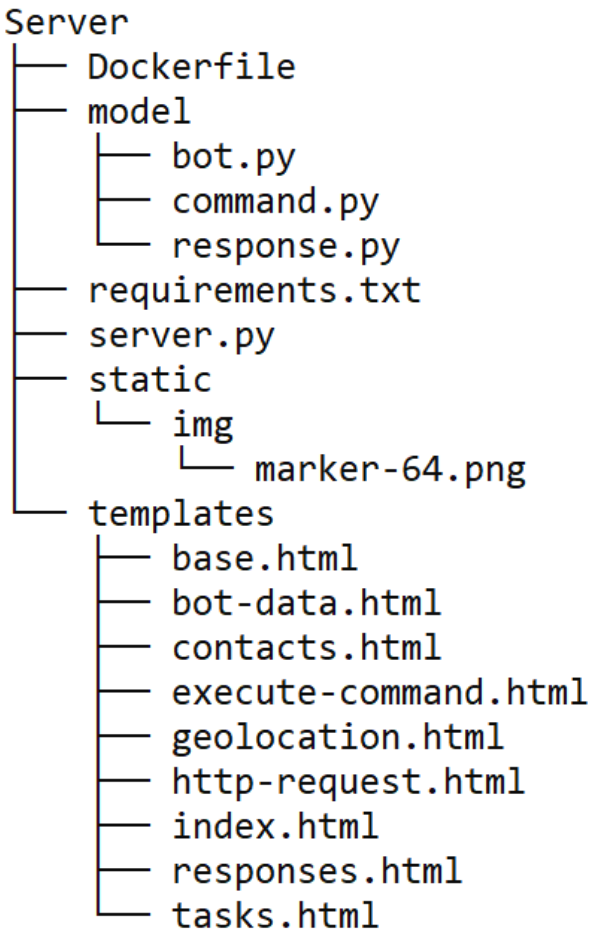

Figura 9.1: Estructura del código - Servidor

La figura 9.2 muestra los 4 niveles superiores de directorios y ficheros de la aplicación cliente que contiene el malware. El directorio `build` contiene los ficheros resultantes, intermedios y temporales que se generan durante el proceso de compilación de la aplicación. Estos se almacenan en las subcarpetas `generated`, `intermediates`, `outputs` y `tmp`. Otro fichero a recalcar es el `./app/build.gradle` a nivel de aplicación. Este define varias variables de configuración para los procesos compilación y empaquetado de la aplicación. Por ejemplo, configura los tipos de compilación, que definen determinadas propiedades que Gradle usa cuando compila y empaqueta la aplicación, o las dependencias del proyecto, que pueden estar ubicadas tanto sistema de archivos local como en repositorios remotos. También permite configurar la firma del APK, variantes de compilación, variantes de productos, entre otras. También se puede recalcar el fichero `build.gradle` de nivel superior el cual permite añadir las opciones de configuración comunes a todos los subproyectos y módulos. Otro fichero relevante es el fichero `google-services.json`, el cual contiene los tokens y demás parámetros necesarios para conectarse de forma segura a la base de datos de Firebase.

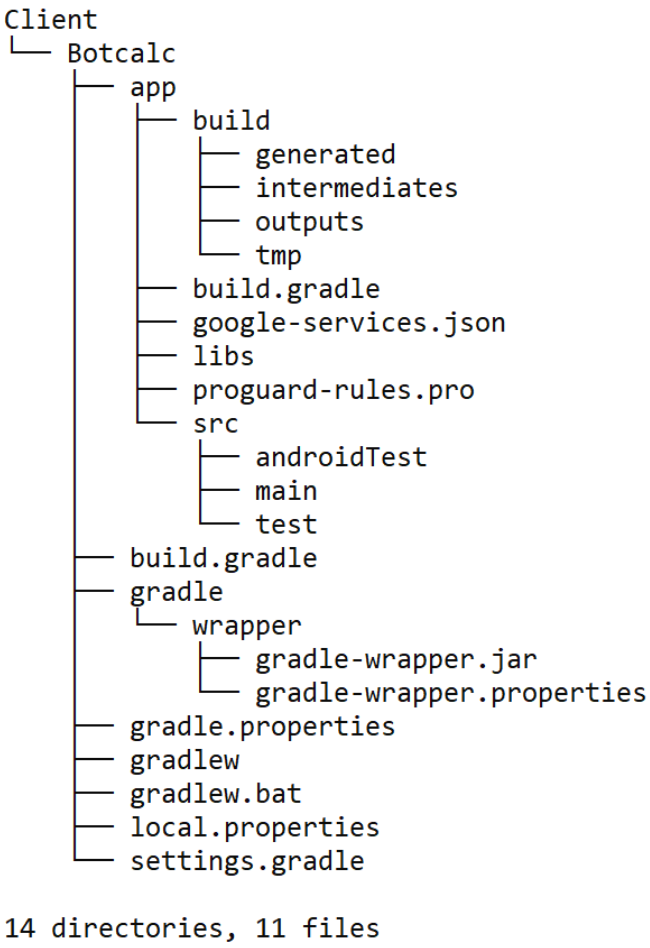

Figura 9.2: Estructura del código - Cliente nivel superior

Por otra parte, el directorio que contiene los ficheros de código es el directorio `src`. En este directorio, están alojados tanto la lógica como la interfaz de la aplicación. Este de divide en 3 subdirectorios 2 de para pruebas (`test` y `androidTest`), y el directorio `main`. A su vez, el directorio `main`, como se aprecia en la figura 9.3, contiene el fichero `AndroidManifiest.xml` y se divide en 2 subdirectorio, llamados `java` y `res`. El fichero `AndroidManifiest.xml`, contiene la declaración de los permisos que requiere la aplicación para funcionar, los componentes de la misma (las actividades, los servicios y los broadcasts receivers, entre otros), el nombre del paquete de la aplicación y las funciones de hardware y software que requiere la aplicación. A su vez, el directorio `java` contiene la lógica de la aplicación y el directorio `res` contiene los recursos necesarios para modelar la interfaz o la parte font-end de la aplicación.

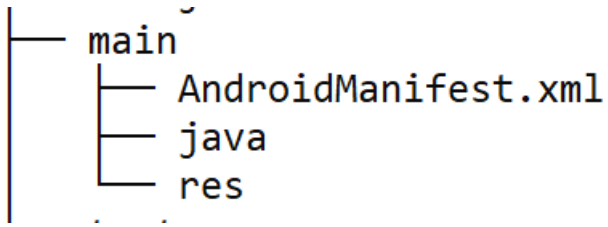

Figura 9.3: Estructura del código - Cliente directorio main

En la figura 9.4, se detalla el subdirectorio `java`. Este, se divide en varios subdirectorios:

- `activities`. Contiene todas las actividades de la aplicación. En esta app solamente hay una actividad, que es la principal (`MainActivity`)

- `domain`. Este incluye las clases resultantes del modelo del dominio. Estas se instanciarán para crear diferentes objetos. Las clases que incluye son: `Bot.java`, `Command.java` y `Response.java`.

- `model`. Contiene las clases del modelo, es decir las que interactuarán con la base de datos de Firebase. Estas son: `Bot.java` y `Response.java`. Aunque se llamen igual son clases totalmente distintas y están ubicadas en paquetes diferentes. Esto se puede ver en el diagrama de paquetes de la app.

- `receivers`. Contiene el broadcast receiver que ejecuta la acción de crear la tarea programada cuando el smarphone se reinicia o se enciende. Este se denomina `RebootBroadcastReceiver`.

- `globals`. Este paquete contiene una clase estática que contiene diferentes constantes que se utilizan en distintas partes del código de la aplicación.

- `helpers`. Este paquete incluye una interfaz que modela el callback que utilizan la clases del modelo para indicar que el dato obtenido está listo, ya que la obtención de datos en Firebase en Android es asíncrona. Esta interfaz recibe el nombre de `GetValueEventListener.java`.

- `services`. Incluye el servicio de accesibilidad personalizado que monitoriza las acciones que realiza el usuario sobre el dispositivo. Este se denomina `KeyloggerService.java`, aunque no solamente incluye un Keylogger.

- `workers`. Este contiene el Worker que trabaja en segundo plano para consultar el siguiente comando a ejecutar y ejecutarlo. Es el que modela la lógica de las labores de comando y control que se ejercen sobre el bot.

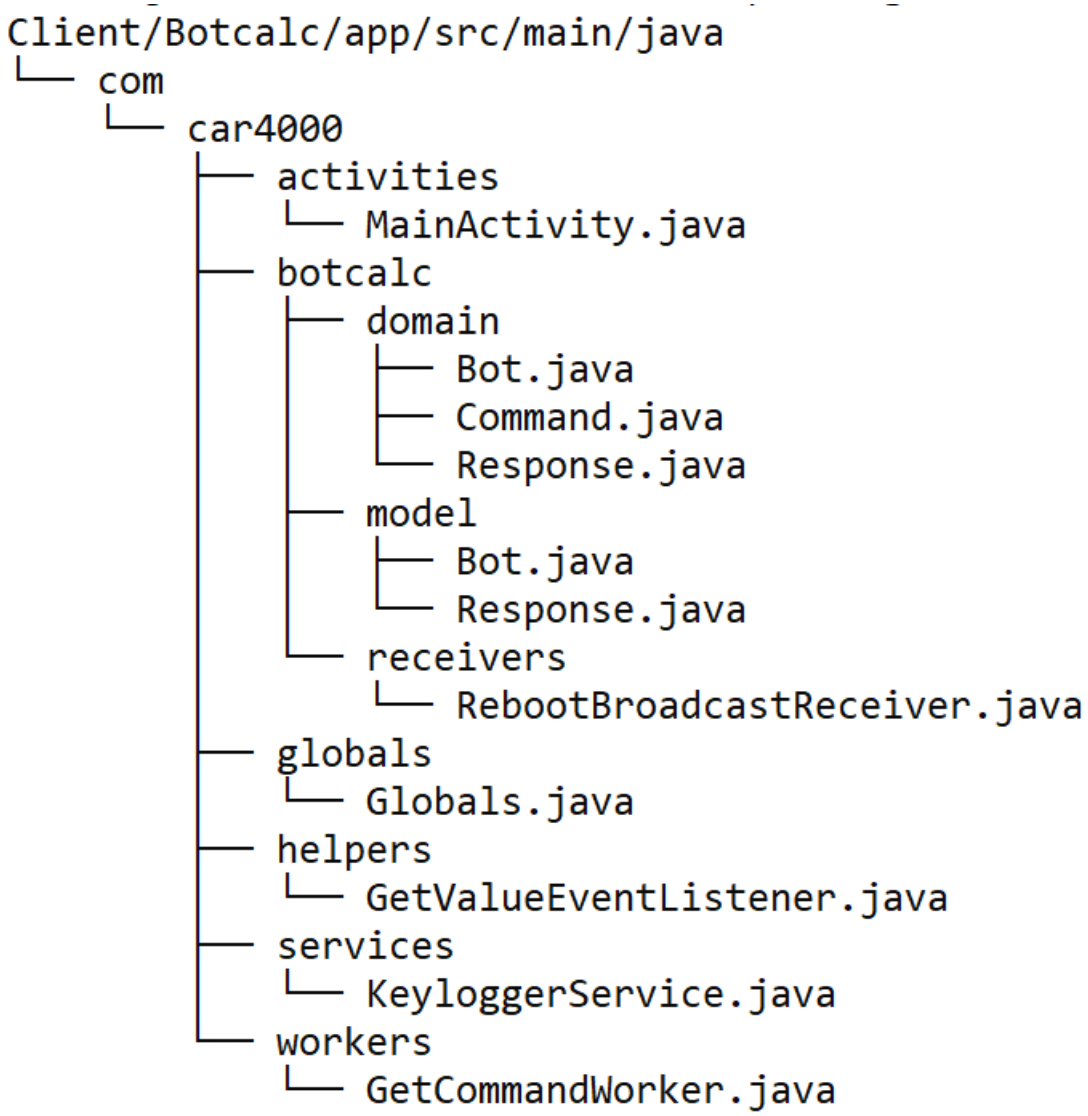

Figura 9.4: Estructura del código - Cliente Java

Finalmente, en la figura 9.5, se puede observar el contenido de la carpeta res. De este se puede recalcar subdirectorio `layout`, que contiene el diseño de las interfaces de las actividades o «layouts». En este caso, como solamente hay una actividad, este solamente contendrá un fichero layout denominado `activity_main.xml`. Otro subdirectorio de `res/` a recalcar es el subdirectorio `values`, que contiene diferentes constantes que almacenan colores (`colors.xml`), ciertas cadenas determinadas utilizadas por la app (`strings.xml`) o los temas (`themes.xml`). Los directorios que comienzan por `mipmap-` contienen el icono de la aplicación en diferentes resoluciones. Y los directorios `drawable` contienen imágenes utilizadas por la aplicación. En este caso la app no utiliza ninguna imagen más que el fondo del icono de la app.

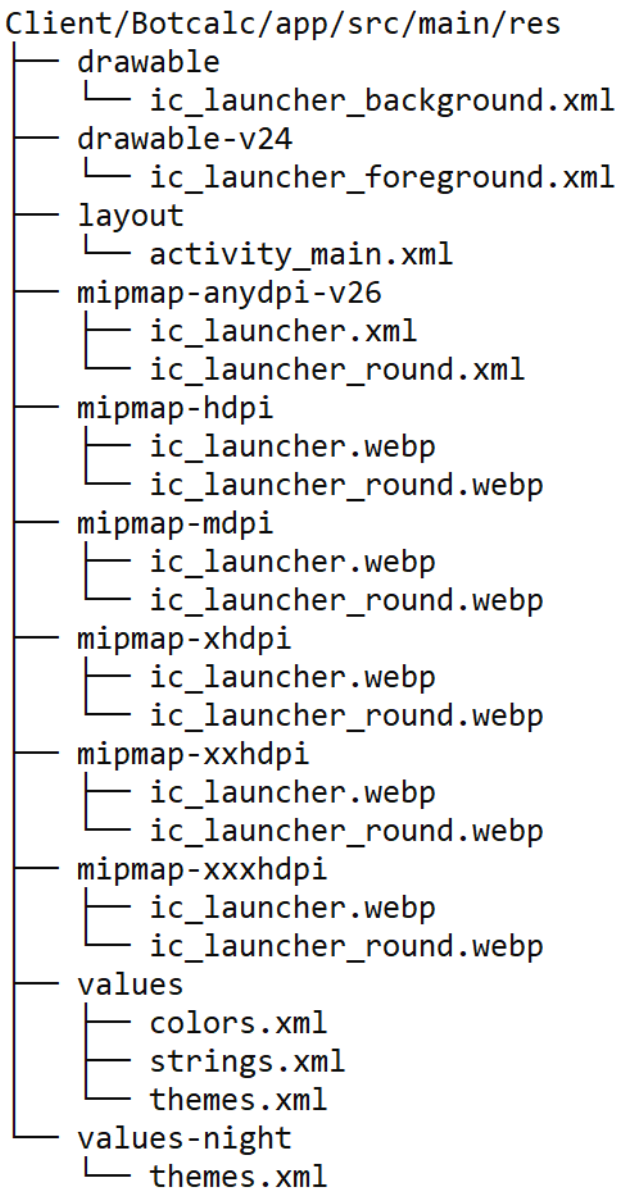

Figura 9.5: Estructura del código - Cliente Res

9.2. Manual de usuario

9.2.1. Sección principal

El presente manual está desarrollado para la aplicación web del servidor C&C. En la sección principal (figura 9.6) de la web se podrán ver los bots que componen la botnet, independientemente de si están encendidos o apagados. Estos bots han instalado en algún momento la aplicación que contiene el malware y la han ejecutado. Estos bots se visualizan en formato de tabla. Cada fila representa a un bot distinto. De cada bot se muestra su identificador único, su marca, modelo, el nombre, el comando o tarea a ejecutar en ese momento y si ha sido ejecutado o no (0 - No ejecutado y 1 - Ejecutado). El nombre del bot es el nombre que tiene asignado ese dispositivo, en este caso el nombre es la versión del firmware que tiene instalado en los dispositivos físicos y en

los virtuales es PI. La aplicación permite la búsqueda de los distintos campos en la misma tabla así como el ordenamiento por el campo que se desee.

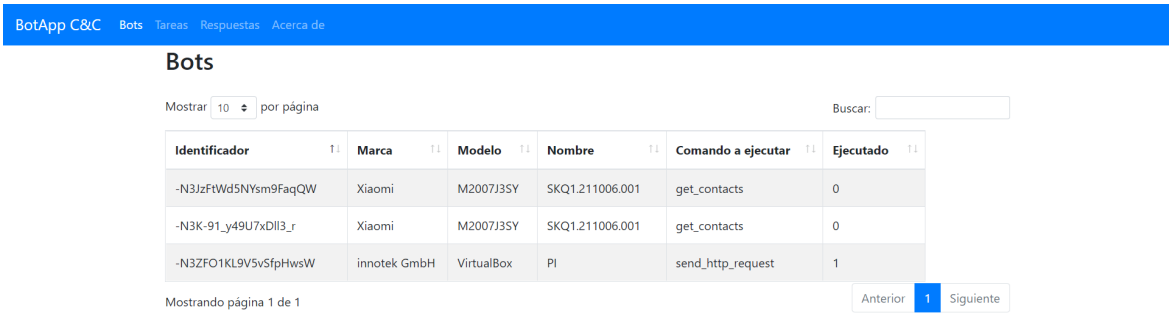

Figura 9.6: Servidor C&C - Sección principal

9.2.2. Sección de tareas

Si se pincha en la sección de Tareas se mostrará una web como la de la figura 9.7. Esta permite la visualización de las tareas de forma ordenada y en formato de tabla. Cada fila de la tabla representa una tarea y de esta se muestra el comando a ejecutar, los parámetros si los tuviera, en el caso de que no los tuviera se indica mediante los literales de `null` o `no`. También se muestra el estado de la misma. Este puede ser `ESPERANDO...` u `OK`. El estado `ESPERANDO...` denota que la tarea aún no se ha ejecutado y que se está esperando a que se ejecute. El estado de `OK`, significa que la tarea ha sido ejecutada correctamente y ya se puede obtener la respuesta de la misma.

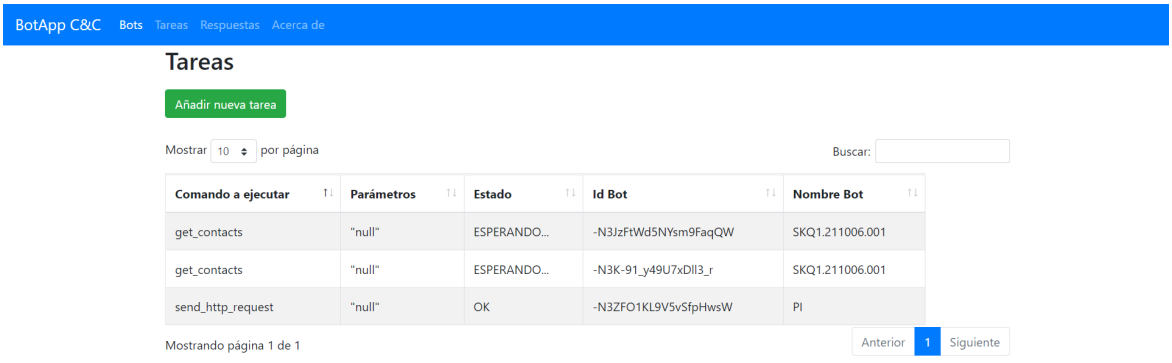

Figura 9.7: Servidor C&C - Sección de las tareas

Por otra parte, en esta sección, se pueden añadir nuevas tareas o comandos para que los bots los ejecuten. Esto se realiza pinchando en el botón de «añadir nueva tarea». Si se hace click en él se desplegará un modal o un pop-up como el de la figura. En este se podrá seleccionar el tipo de tarea que se desea ejecutar. En la figura se puede apreciar la lista de las tareas que están disponibles para seleccionar. y el bot en el cual se ejecutará (figura 9.8). Para seleccionar la tarea se despliega la lista con el de comando a ejecutar y se pincha en la tarea deseada. En función de la tarea seleccionada saldrán diferentes campos adicionales (un ejemplo de ello se puede ver en la figura 9.9). Estos campos son parámetros que se le deben pasan la tarea para que se ejecute correctamente. Las tareas disponibles son las siguientes:

- `OBTENER CONTACTOS`. Esta tarea sirve para obtener todos los contactos del terminal. No es necesario rellenar campos adicionales.

- `GEOLOCALIZAR`. Este comando permite obtener la ubicación exacta del bot. Tampoco es necesario rellenar campos adicionales.

- `ENVIAR SMS`. Mediante este comando se podrá ordenar al bot que envíe SMS a un destinatario en concreto. Los campos adicionales que deben rellenar son: el número de teléfono (con prefijo internacional) y el texto que se desea enviar. Esto se puede apreciar en la figura .

- `EJECUTAR COMANDO`. Si se selecciona y se añade esta tarea se le ordenará al bot que ejecute un comando de linux. Solamente se podrán ejecutar en móviles «no rooteados» los comandos que no requieran privilegios de `root`. Para ejecutar esta tarea, se debe rellenar el campo denominado como «Comando».

- `REALIZAR PETICIÓN HTTP`. Si se le ordena al bot realizar esta tarea, este realizará una petición HTTP a la URL que se le indique. Por defecto esta será de tipo GET. También se le podrán pasar cookies de forma adicional separadas por puntos y comas de la siguiente forma: `nombreCookie1=cookie1;nombreCookie2=cookie2 ...`

También se podrá seleccionar el bot en el que se desea que se ejecute la tarea. Si se selecciona la opción de «Todos los bots», el comando se ejecutará en todos los bots que componen la botnet.

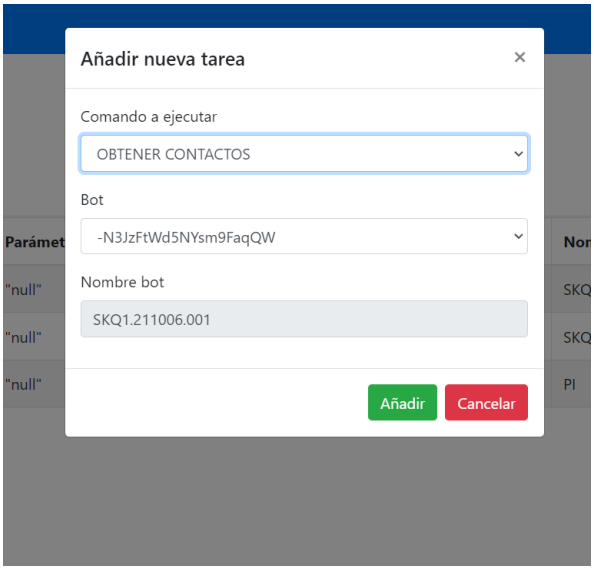

Figura 9.8: Servidor C&C - Añadir nueva tarea

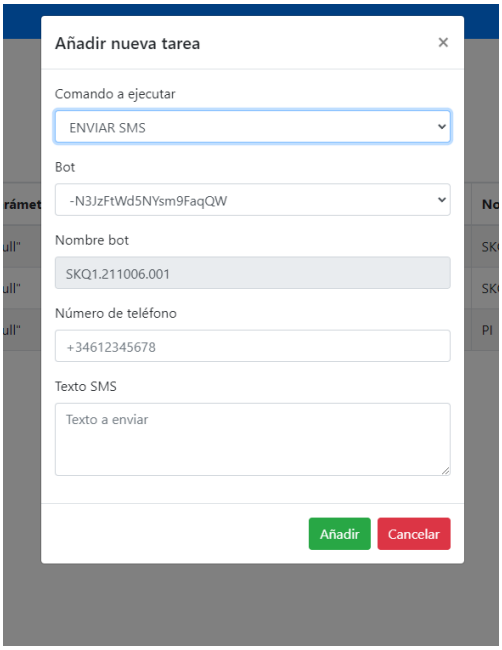

Figura 9.9: Servidor C&C - Ejemplo tarea de envío de SMS

Si se pulsa en añadir se quedará cargando hasta que se añada la tarea correctamente. Cuando esto ocurra, saldrá un pop-up como el de la figura, indicando que la tarea se ha añadido correctamente, y a continuación, se recargará automáticamente de forma que ya aparezca la nueva tarea reflejada. Algo a recalcar es que esta tarea reemplazará a la anterior independientemente de si se ha ejecutado o no. Solamente un bot puede tener una tarea pendiente a la vez.

9.2.3. Sección de respuestas

En esta sección, como se puede ver en la figura 9.10, se podrán visualizar todas las respuestas en forma de tabla. Las mismas vienen ordenadas por defecto por fecha de más reciente a menos

reciente. Sin embargo, pinchando sobre la cabecera de cada campo se podrán ordenar como se deseen. También se puede buscar en la tabla. De cada respuesta se muestra su identificador único («ID Respuesta»), el identificador del bot que la realizó («ID Bot»), el comando al que viene asociado la misma y la fecha en la que realizó la misma en formato datetime.

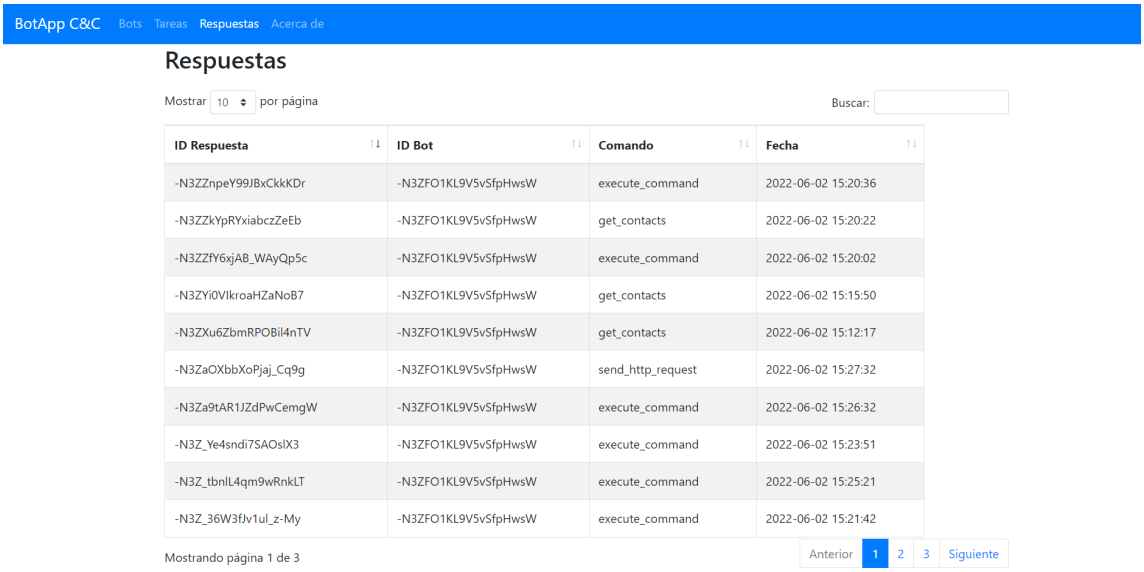

Figura 9.10: Servidor C&C - Sección de respuestas

Si se pincha en una respuesta en concreto se podrá visualizar el detalle de la misma, es decir, el contenido de la respuesta en sí. En función del comando al que venga asociado se representará de una forma o de otra. A continuación se exponen las diferentes formas en las que se representa.

La respuesta a un comando de geolocalización, como se puede ver en la figura 9.11, se representa mediante un mapa interactivo en el cuál vendrá marcado en morado el lugar en el que se localizaba el bot. También sale la hora en la que se le ubicó, que es la misma en la que respondió al comando indicando su localización.

Los contactos obtenidos del terminal se representan en forma de tabla, tal y como se puede apreciar en la figura 9.12. De cada contacto se muestra su nombre, apellidos, su primer número y su segundo número si lo tiene. Por defecto viene ordenados de forma alfabética por nombre. Si se desea otro tipo de ordenamiento se puede pinchar en el campo sobre el que se quiere ordenar.

Las respuestas a la ejecución de un comando remoto se representan como en la figura 9.13. Se muestra la respuesta de la salida estándar del comando en color verde sobre un fondo negro.

Finalmente, se representa la web que devuelve la petición a un servidor cuando se ejecute el comando `ENVIAR PETICIÓN HTTP`. Esto se puede apreciar en la figura 9.14.

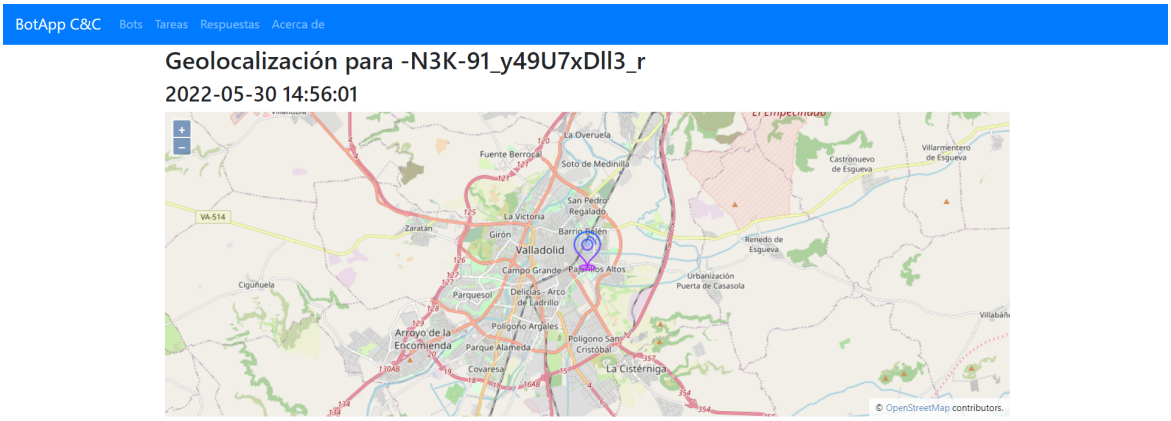

Figura 9.11: Servidor C&C - Comando de geolocalización

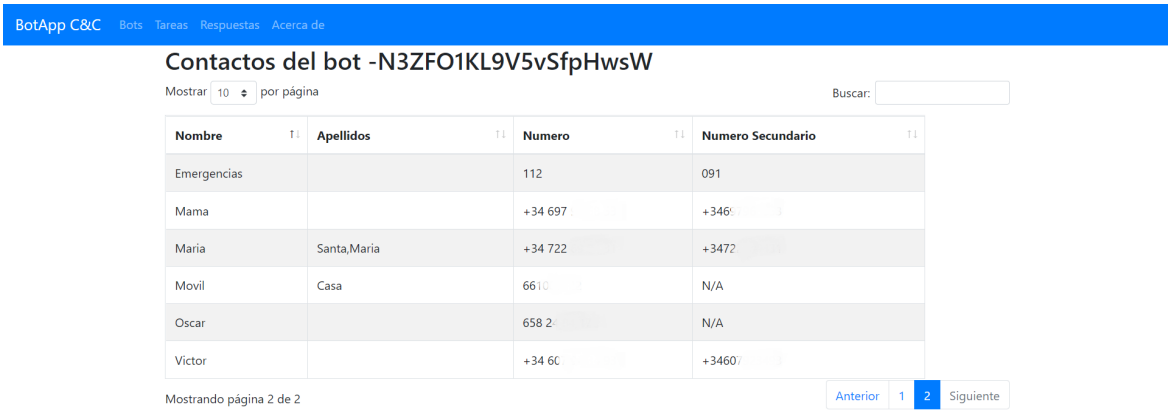

Figura 9.12: Servidor C&C - Contactos

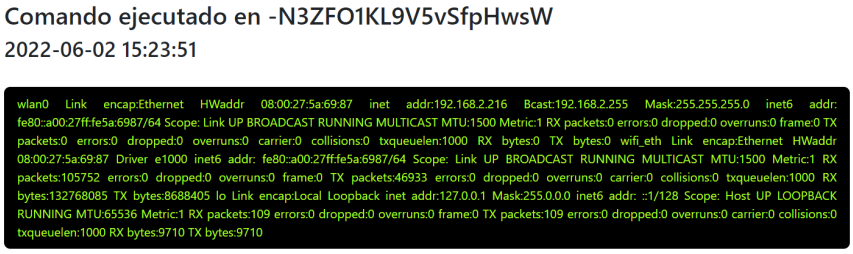

Figura 9.13: Servidor C&C - Ejecución de un comando remoto

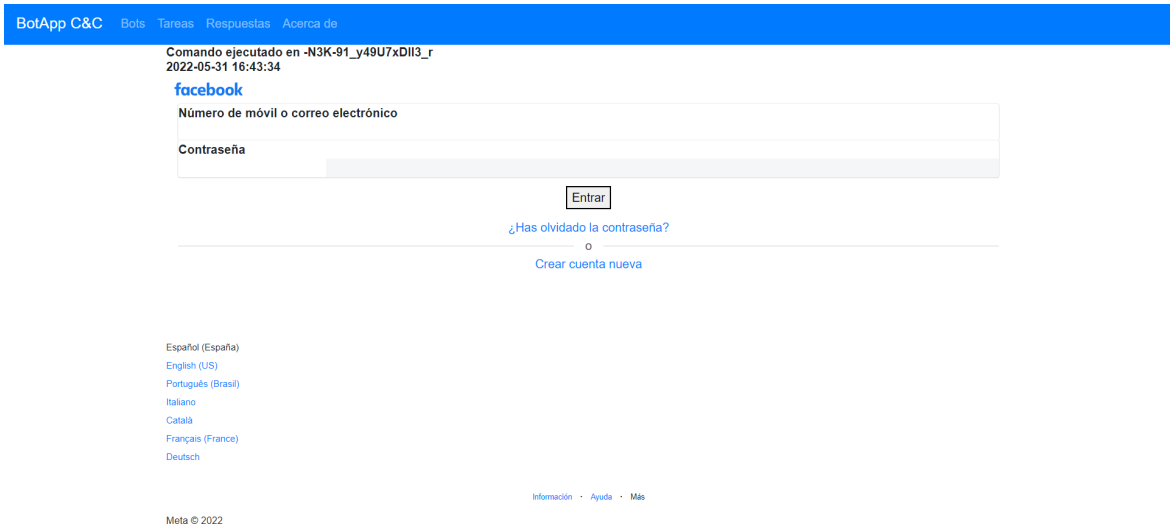

Figura 9.14: Servidor C&C - Respuesta HTTP de la petición enviada por el bot

9.2.4. Historial de acciones del bot

Para obtener el historial de acciones de un bot bastaría con irse a la sección principal, donde salen todos los bots, y pinchar en un bot en concreto. Al pinchar en el mismo se mostrará su historial de acciones. Este se representa en forma de tabla como las tablas anteriormente expuestas. Las figuras 9.15, 9.16 y 9.17 muestran varias acciones realizadas por el usuario víctima. También incluyen información confidencial como una contraseña del correo electrónico y fragmentos de conversaciones de WhatsApp. En la misma tabla se puede Se puede buscar, ordenar de la misma forma las distintas acciones y otro tipo de información. Las acciones y la información recopilada vienen ordenadas cronológicamente de más recientes a menos recientes, con una precisión de milisegundos. Esto se puede apreciar en la figura. De cada acción se muestra su la fecha en formato datetime, el tipo de evento y la información detallada de la misma. El tipo de evento puede ser:

- `CLICKED`. Cuando el usuario realiza un click en alguna parte de la pantalla que no sea el teclado. De este tipo de evento se muestra el contexto de la aplicación a la que se hizo click (por ejemplo: `com.whatsapp`, y el detalle de lo que se ha hecho click

- `TEXT`. Este tipo de evento se desencadena cuando el usuario escribe algo en el teclado virtual del móvil. Se muestra entre corchetes lo que lleva escrito en el buffer del teclado.

- `FOCUSED`. Este tipo de evento se desencadena cuando cambia el foco de la aplicación. Por ejemplo, en un formulario se cambia del campo nombre al campo apellidos.

Figura 9.15: Servidor C&C - Historial de acciones - Conversación de WhatsApp

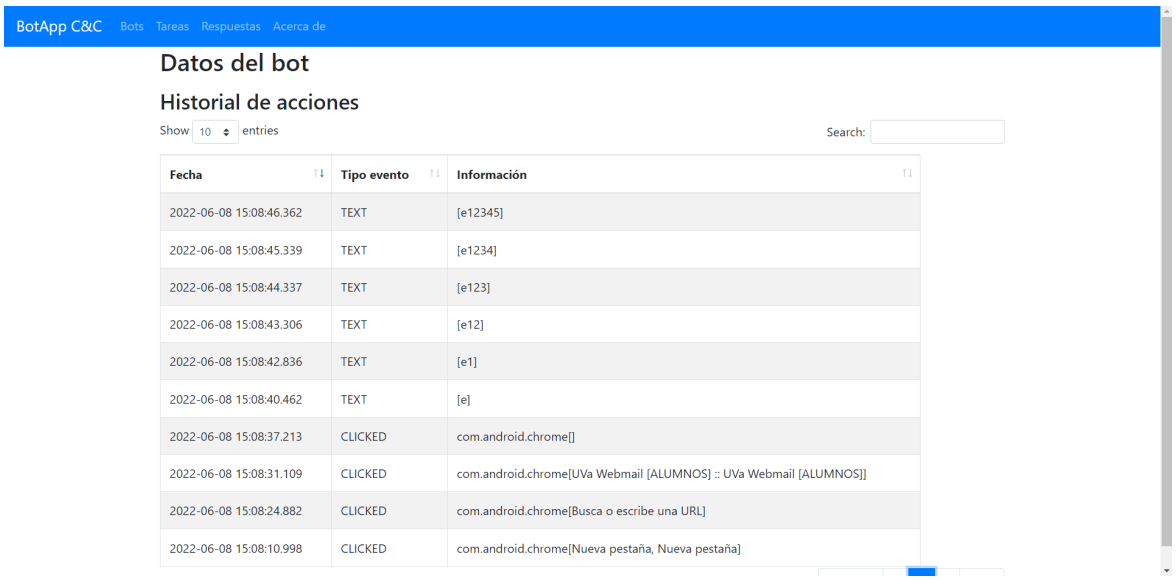

Figura 9.16: Servidor C&C - Historial de acciones - Robo de credencial de mail (1)

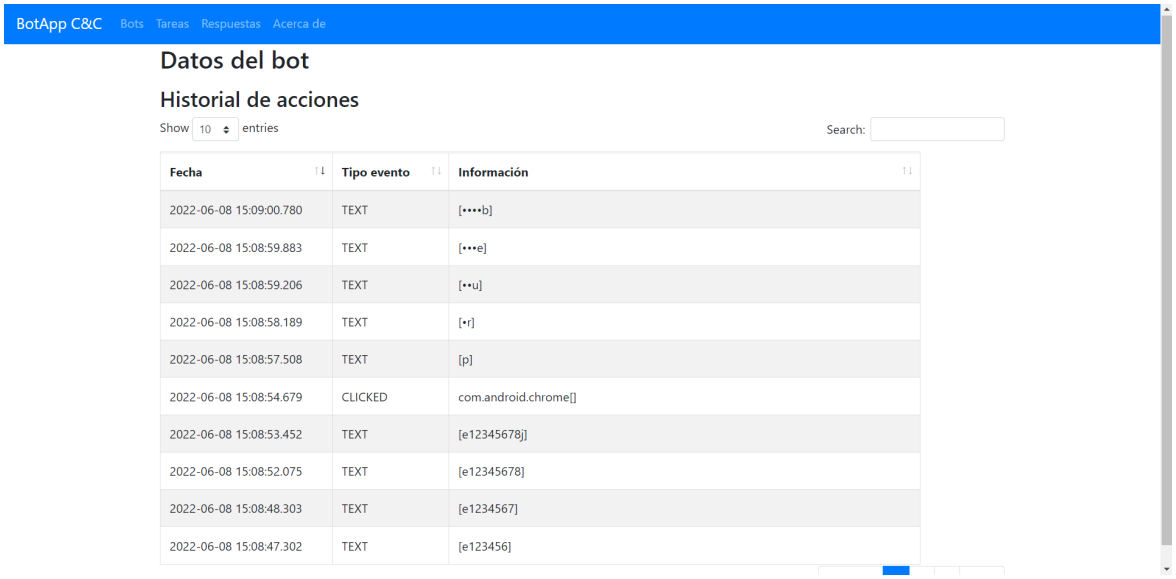

Figura 9.17: Servidor C&C - Historial de acciones - Robo de credencial de mail (2)

9.3. Código y Licencia

El código está almacenado en el repositorio https://gitlab.inf.uva.es/carmaci/tfg-botnet.git A este proyecto se le ha dotado de una licencia de tipo MIT. Este tipo de licencia es una licencia libre y permisiva. Esto significa que se imponen muy pocas limitaciones en cuanto a la reutilización del software se refiere. De esta forma con esta licencia se obtiene software libre.

Bibliografía

[1] CCN-CERT. *Informe anual sobre ciberamenazas y tendencias 2021*. URL: https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/6338-ccn-cert-ia-13-21-ciberamenazas-y-tendencias-edicion-2021-1/file.html (visitado 20-02-2022).

[2] INCIBE. *Botnet. Qué es una botnet*. URL: https://www.incibe.es/aprendeciberseguridad/botnet (visitado 28-02-2022).

[3] ENISA (European Union Agency for Network e Information Security). *Botnets*. URL: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/botnets (visitado 22-02-2022).

[4] INCIBE. *Cómo proteger tu empresa ante las botnets*. URL: https://www.incibe.es/protege-tu-empresa/blog/botnet-y-saber-si-tu-empresa-forma-parte-ella (visitado 20-02-2022).

[5] White Ops. *9 of History's Notable Botnet Attacks*. URL: https://www.humansecurity.com/blog/9-of-the-most-notable-botnets (visitado 23-02-2022).

[6] Us Norton Inc. *What is a botnet*. URL: https://us.norton.com/internetsecurity-malware-what-is-a-botnet.html (visitado 21-02-2022).

[7] Google. *Arquitectura de Android*. URL: https://source.android.com/devices/architecture (visitado 16-03-2022).

[8] Google. *Android Security Enterprise Whitepaper*. URL: https://source.android.com/security/reports/Google%5C_Android%5C_Enterprise%5C_Security%5C_Whitepaper%5C_2020.pdf (visitado 16-03-2022).

[9] Google. *Características de seguridad de Android*. URL: https://source.android.com/security/features (visitado 16-03-2022).

[10] CCN-CERT. *Informe de buenas prácticas*. URL: https://www.ccn-cert.cni.es/informes/informes-de-buenas-practicas-bp/1757-ccn-cert-bp-03-dispositivos-moviles/file.html (visitado 20-03-2022).

[11] Google. *Boletín de Seguridad de Febrero de 2022*. URL: https://source.android.com/security/bulletin/2022-02-01 (visitado 20-03-2022).

[12] Blake Strom. *ATT&CK 101*. URL: https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/attck-101 (visitado 28-02-2022).

[13] MITRE Corp. *Matriz ATT&CK MITRE*. URL: https://attack.mitre.org/ (visitado 20-02-2022).

[14] Bob Hughes y Cotterell Mike. *Software Project Management*. McGraw Hill / Europe, Middle East & Africa, 2009. ISBN: 978-0077122799.

[15] Jobted. *Salarios medios programadores en España 2022*. URL: https://www.jobted.es/salario/programador#:~:text=Un%5C%20Programador%5C%20junior%5C%2C%5C%20con%5C%20menos,19.700%5C%20%5C%E2%5C%82%5C%AC%5C%20brutos%5C%20por%5C%20a%5C%C3%5C%B1o. (visitado 01-03-2022).

[16] Agencia Tributaria. *Tabla de coeficientes de amortización lineal*. URL: https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/requisitos-generales/coeficientes-amortizacion-lineal.html (visitado 01-03-2022).

[17] LaTeX Project ORG. *LaTeX documentation*. URL: https://www.latex-project.org/help/documentation/ (visitado 25-04-2022).

[18] Astah. *Software de modelado UML Astah*. URL: https://astah.net/ (visitado 20-04-2022).

[19] Diagrams.net. *Software de dibujo de diagramas*. URL: https://www.diagrams.net/ (visitado 25-04-2022).

[20] Python Org. *Documentación de Python3*. URL: https://docs.python.org/3/ (visitado 10-04-2013).

[21] Twitter. *Documentación de Boostrap*. URL: https://getbootstrap.com/docs/ (visitado 10-04-2022).

[22] jQuery. *jQuery AJAX*. URL: https://api.jquery.com/jquery.ajax/ (visitado 19-04-2022).

[23] Datatables.net. *Manual de la librería datatables*. URL: https://datatables.net/manual/ (visitado 05-04-2022).

[24] Open Layers Org. *Documentación de la libería de mapas JS Openlayers*. URL: https://openlayers.org/en/latest/doc/tutorials/ (visitado 28-04-2022).

[25] Google. *Documentación Firebase*. URL: https://firebase.google.com/docs (visitado 29-03-2022).

[26] Oracle. *VirtualBox*. URL: https://www.virtualbox.org/wiki/Documentation (visitado 10-03-2022).

[27] Android-x86 ORG. *Proyecto de Android para arquitecturas x86*. URL: https://www.android-x86.org/ (visitado 10-03-2022).

[28] Docker Inc. *Documentación de Docker*. URL: https://docs.docker.com/ (visitado 29-05-2022).

[29] Git. *Git SCM*. URL: https://git-scm.com/ (visitado 10-05-2022).

[30] MITRE Corp. *MITRE ATT&CK Táctica TA0028 - Acceso Inicial*. URL: https://attack.mitre.org/tactics/TA0027/ (visitado 25-03-2022).

[31] MITRE Corp. *MITRE ATT&CK Táctica TA0041 - Ejecución*. URL: https://attack.mitre.org/tactics/TA0041/ (visitado 26-03-2022).

[32] MITRE Corp. *MITRE ATT&CK Táctica TA0028 - Persistencia*. URL: https://attack.mitre.org/tactics/TA0028/ (visitado 25-03-2022).

[33] MITRE Corp. *MITRE ATT&CK Táctica TA0033 - Movimiento lateral*. URL: https://attack.mitre.org/tactics/TA0033/ (visitado 25-03-2022).

[34] MITRE Corp. *MITRE ATT&CK Táctica TA0035 - Recolección*. URL: https://attack.mitre.org/tactics/TA0035/ (visitado 10-05-2022).

[35] Google. *Servicio de accesibilidad personalizado para Android*. URL: https://developer.android.com/guide/topics/ui/accessibility/service (visitado 10-05-2022).

[36] MITRE Corp. *MITRE ATT&CK Táctica TA0037 - Comando y Control*. URL: https://attack.mitre.org/tactics/TA0037/ (visitado 08-03-2022).

[37] *How to reduce time of PeriodicWorkManager in WorkManager*. URL: https://stackoverflow.com/questions/55135999/how-to-reduce-time-of-periodicworkmanager-in-workmanager (visitado 24-03-2022).

[38] Google. *Tareas programadas con Workmanager*. URL: https://developer.android.com/topic/libraries/architecture/workmanager?hl=es-419 (visitado 22-03-2022).

[39] Java2s. *How to execute a shell command Android*. URL: http://www.java2s.com/example/android/android-os/execute-shell-command-and-get-result.html (visitado 12-04-2022).

[40] Google. *Send a HTTP request with Volley library*. URL: https://google.github.io/volley/ (visitado 17-04-2022).

[41] Google. *Broadcast Receivers*. URL: https://developer.android.com/guide/components/broadcasts?hl=es-419 (visitado 25-03-2022).

[42] *Broadcast receiver for reboot in Xiaomi phones*. URL: https://stackoverflow.com/questions/29627856/reboot-receiver-is-not-working-in-xiaomi-phones (visitado 27-03-2022).

[43] Rahul. *How to setup autorun a python script using Systemd*. URL: https://tecadmin.net/setup-autorun-python-script-using-systemd/ (visitado 28-05-2022).

[44] Digital Ocean. *Cómo crear una aplicación Web usando Flask en Python 3*. URL: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3-es (visitado 01-04-2022).

[45] Google. *Como depurar las tareas programadas*. URL: https://developer.android.com/studio/inspect/task (visitado 10-04-2022).

[46] Carlos Macías Valdivieso. *Trabajo de la asignatura de Sistemas Móviles: Seguridad, autenticación, y privacidad en dispositivos móviles*. URL: https://campusvirtual.uva.es/pluginfile.php/3158202/assignsubmission_file/submission_files/2165730/SeguridadMovilesMaciasValdivieso.pdf?forcedownload=1 (visitado 05-03-2022).

[47] thisbejim. *Librería de Python para la API de Firebase*. URL: https://github.com/thisbejim/Pyrebase (visitado 10-04-2022).

[48] n37sn4k3. *An example of Keylogger with Accesibility Service*. URL: https://github.com/n37sn4k3/Android-Accessibility-Keylogger (visitado 25-05-2022).