Universidad Carlos III de Madrid - Escuela Politécnica Superior
Grado en Ingeniería Informática
Seguridad en dispositivos móviles

*Grupo de investigación: Computer Security Lab*

# Android botnets for multi-targeted attacks

Autores:
**Adán Cano Moreno. NIA: 100346105.**
**Jaime García González. NIA: 100346062.**

Grupo: **81.**

27 de marzo de 2019

# Índice general

**1. Resumen** **1**

**2. Introducción** **2**

**3. Trabajos previos** **4**

3.1. When the Droid became the Bot . . . . . 4

3.2. Andbot: Towards Advanced Mobile Botnets . . . . . 5

3.3. A K-Means Clustering Algorithm . . . . . 6

**4. Trabajos posteriores** **7**

4.1. Android botnet detection: An integrated source code mining approach . . . . . 7

4.2. Investigation into Google Play security mechanisms via experimental botnet . . . 8

**5. Revisión crítica del artículo** **9**

**6. Conclusiones** **10**

**7. Información complementaria** **12**

7.1. Información básica del autor . . . . . 12

7.2. Información básica de la revista . . . . . 12

7.3. Planificación . . . . . 12

# Capítulo 1

# Resumen

A continuación, se detallan las tareas realizadas por cada uno de los integrantes del grupo.

## Jaime

**Trabajos previos.** Lectura de trabajos previos obtenidos de la bibliografía del artículo. Redacción del Capítulo 3.

**Trabajos posteriores.** Búsqueda y lectura de trabajos posteriores. Redacción del Capítulo 4.

**Conclusiones.** Redacción del Capítulo 6.

**Planificación.** Realización del diagrama de Gantt de la sección 7.3 con la herramienta *pgfgantt*.

**Glosario.** Búsqueda de términos importantes del documento y redacción del Capítulo 7.3.

**Bibliografía.** Realización de la bibliografía con la herramienta BibLaTeX.

## Adán

# Capítulo 2

# Introducción

Un *botnet* se compone de dos partes: el cliente (que es a quien se ataca) y el servidor (que es quien realiza el ataque y quien recoge los datos). Al comienzo del artículo se muestra una breve introducción a los ataques a múltiples objetivos y la el por qué de realizarlo sobre dispositivos móviles. A continuación muestra una implementación del ataque donde se generan varias fases:

**Recoger información** : Como se puede recoger información de un dispositivo a través de la aplicaión *Android*, que es el cliente del *botnet*. En esta fase se muestra un ejemplo de una modificación del código de una aplicación para poder leer datos del dispositivo a través de la geolocalización.

**Almacenamiento y gestión de la información** : En esta fase se describe el procedimiento a seguir para almacenar la información que ha sido recogida a través del ataque. Para ello se puede usar tanto *PHP* como *MySQL* ya que ambos están instalados en la mayoría de los servidores de internet. En el artículo se indica que para almacenar la información de cada dispositivo lo mejor es crear una tabla por dispositivo cuya clave indentificativa sea el número *IMEI*, ya que es un número único e identificativo para cada dispositivo.

**Mostrar la información** : En este fase se describe brevemente como poder mostrar en un página la información que ha sido recogida. Para ello se necesita generar código dinámico a través de *Javascript* incrustado en código *PHP*. Para acceder a los datos almacenados se hace una consulta a la base de datos a través de el comando SELECT a la base de datos del dispositivo del que queramos recuperar sus datos. Todos los nombres de las tablas son almacenados en un array para poder hacer las consultas a la base de datos. Para mostar la información en una web, se incluye la cabecera de *HTML* de una página web para inicializar la *API* de *Google Maps* y a través de *PHP* podemos mostrar diferentes iconos por cada *botnet* en la página web y así poder diferenciar la información.

**Información de verificación** : En objetivo de esta fase es conocer el comportamiento de los usuarios que están siendo atacados por un *botnet*. Se desea conocer si un persona infectada viaja en autobús, en tren o simplemente camina o está parado. Para ello se debe determinar la velocidad de movimiento del infectado. Por lo tanto, el problema de este ataque es conseguir suficiente información como para poder determinar el comportamiento de la persona a la que se ataca. Lo primero que se debe hacer es crear una nueva tabla donde, por cada instante de tiempo, realizar una verificación de la información capturada de diferentes dispositivos. Esto nos va a facilitar el proceso de gestión de los datos de diferentes objetivos a la vez. Cuando queramos consultar la información de verificación

debemos indicar que queremos mostrar la información de manera ordenada en función del instante de tiempo para poder analizar la información de manera eficiente.

**Determinar puntos de encuentro entre dispositivos** : En esta fase se describe el proceso para poder determinar el momento en el cual dos dispositivos pueden estar en mismo lugar. Para ello se hace uso del algoritmo de *k-means*. En artículo se muestra un ejemplo de implementación del algoritmo donde lo primero que se hace es seleccionar la información de los dispositivos que se desean analizar y almacenarlo en un cluster. El algoritmo *k-means* puede ser aplicado de manera recursiva mientras los clusters no cumplan con las especificaciones deseadas. Si cumplen con los requisitos la primera vez basta con aplicarlo en una sola ocasión. El algoritmo es desarrollado en *PHP* y necesita implementar una función que compruebe que el cluster cumple con los requisitos especificados. Los clusters seleccionados son concatenados en un array único para poder determinar de una manera más sencilla los puntos de encuentro. En la implementación mostrada en el artículo, los centroides iniciales son calculados de manera aleatoria dentro del rango de los máximos y mínimos de longitud y latitud observados en los datos. Se calculan a través de este método para asegurar buenos resultados. Los demás centroides son calculados a través de una ecuación de distancias de puntos sobre una esfera a partir de sus longitudes y latitudes. Antes de calcular los nuevos centroides se transforman los puntos de longitud y latitud al eje cartesiano. En cada iteración del algoritmo se calcula el cluster a partir de los puntos donde cada distancia entre los puntos calculados es menor a la distancia máxima. El atacante decide cuando dos víctimas están cercanas en función de un radio de distancia. Para cada cluster se selecciona los posibles puntos de encuentro en función del tiempo y la distacia. Esta es la razón por la cual almacenamos la información de verificación en función del instante de tiempo. De esta manera podemos analizar las zonas donde hay posibilidades de que sean puntos de encuentro. Si no hay ningún resultado viable, significa que los datos capturados no nos permiten determinar un punto de encuentro.

# Capítulo 3

# Trabajos previos

## 3.1. When the Droid became the Bot

Uno de los grandes problemas de la seguridad en la actualidad ha sido el desarrollo de *botnets*. El *botmaster* despliega el código malicioso a diferentes *hosts* vulnerables. Cuando el sistema ha sido infectado, el dispositivo pasa a formar parte de la *botnet*, comienza a buscar nuevos dispositivos vulnerables y a realizar otras prácticas ilegales.

Las actividades delictivas más habituales que realizan las *botnets* son ataques de *spam* y ataques de denegación de servicio (DDoS). En los últimos años, el uso de *smartphones* ha crecido de manera exponencial, por ello, los criminales deciden infectar estos dispositivos para realizar los ataques expuestos anteriormente.

Con la aparición de la tercera generación de redes móviles (3G), los *smartphones* empiezan a conectarse a Internet, con las ventajas y desventajas que implica. Si el dispositivo resulta infectado, el *botmaster* tiene la capacidad de ejecutar diferentes acciones sobre sus esclavos, como pueden ser el envío de SMS, instalar y borrar aplicaciones, realizar llamadas ... Aprovechando esta funcionalidad, algunos de los ataques más habituales que ejecutan las *botnets* son los siguientes:

***Spyware.*** Este tipo de ataques se utiliza para recoger datos personales (nombres, direcciones, datos bancarios ...) del atacado. Esto se consigue con la lectura de mensajes, monitorizando la posición del dispositivo, analizando la actividad en Internet y escuchando conversaciones telefónicas.

**DDoS.** Ataques de denegación de servicio contra *call centers* o diferentes webs.

**Ataques al núcleo de la red.** Se realizan ataques de *flooding* a la red celular durante fechas importantes del año, como año nuevo, para colapsar el sistema. Se produce un cuello de botella en el HLR (*Home location register*), que es la base de datos que contiene cada teléfono móvil de la red GSM, hasta forzar su fallo.

En el entorno de pruebas, se han utilizado tres dispositivos *rooteados*, el *botmaster*, la víctima de spam y el dispositivo infectado. El objetivo de la investigación es detectar al *botmaster* analizando los SMS que recibe la víctima.

Se ha analizado el dispositivo afectado, mirando su tarjeta SIM, el almacenamiento interno y su tarjeta SD y no se han encontrado los mensajes de *spam* recibidos. Esto se debe a que el *malware* se ejecuta en las capas intermedias del dispositivo, en vez de en espacio de usuario, véase 3.1.

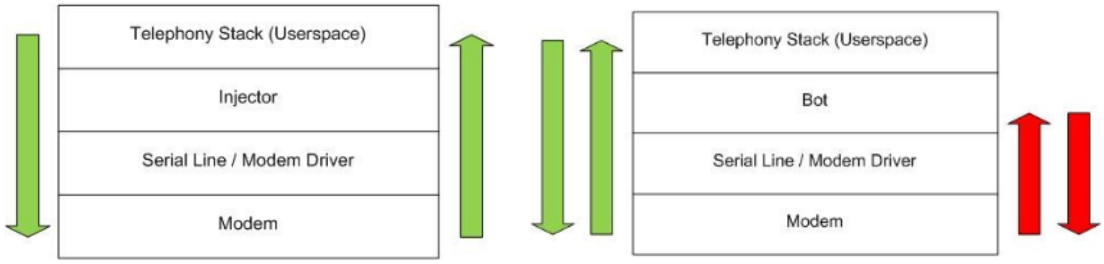



Figura 3.1: *Bot layer*

## 3.2. Andbot: Towards Advanced Mobile Botnets

Andbot es una *botnet* avanzada que se ejecuta en dispositivos *Android*. Sus características principales son las siguientes:

**Sigilosa.** Utiliza el protocolo HTTP, solo tiene acceso a Internet en segundo plano.

**Robusta.** Resistente a la mayoría de técnicas de defensa (sumidero DNS, inyección de comandos, lista negra de IPs ...).

**Barata.** Bajo coste, bajo uso de datos y bajo gasto energético.

**Soporte para comandos.** *Phishing* y filtrado, DDoS, robo de información, apagado y control de llamadas.

La parte más importante de una *botnet* es el C&C (*Command and controle design*). Para autenticar comandos, Andbot utiliza el algoritmo RSA. Para enviar comandos a los dispositivos infectados, el *botmaster* genera una imagen JPG en la que se incluye el comando encriptado y firmado, seguidamente se sube a un blog y se comprime la URL. Por otro lado, combina la nueva URL con la fecha de inicio y de expiración del comando. Finalmente, el dispositivo infectado descarga la imagen JPG, descifra el comando y lo ejecuta.

Las pruebas se han realizado en cuatros dispositivos *Android* y se ha incluido el código malicioso dentro de un conocido juego. La primera vez que se ejecuta el juego, Andbot comienza ha realizar sus funciones en segundo plano. Después de reiniciar, el juego se inicia de forma automática y el código se ejecuta siempre que el dispositivo se encuentre en reposo.

Para hacer frente a esta Andbot se recomienda utilizar las técnicas defensivas habituales contra *botnet*, como construir mecanismos de detección coordinados, realizar análisis en busca de comandos que utilicen las vulnerabilidades del sistema y ejecutar por primera vez el código en *sandboxes*.

## 3.3. A K-Means Clustering Algorithm

El algoritmo K-means [1] fue descrito por Hartigan en 1975. Este algoritmo tiene como objetivo dividir un conjunto $MxN$ en $K$ clusters. No se realiza la suma de los cuadrados de todas las particiones, excepto cuando $M$ y $N$ toman valores pequeños y $K = 2$. Para el caso general, se busca una solución óptima local que requiera mover un punto de un cluster a otro para evitar la suma de cuadrados.

Este algoritmo necesita una serie parámetros de entrada para poder ejecutar correctamente, son los siguientes:

1. Matriz de entrada de $M$ puntos por $N$ dimensiones.
2. Matriz inicial de $K$ centroides por $N$ dimensiones.

# Capítulo 4

# Trabajos posteriores

En los años siguientes al artículo original, se han realizado multitud de trabajos similares, ya que, el sistema operativo *Android* es cada vez más popular y tiene mayor cuota de mercado. A continuación, se resumen algunos de estos.

## 4.1. Android botnet detection: An integrated source code mining approach

Este artículo propone diferentes técnicas para detectar las *botnets* en *Android* [2]. Para ello, se aplica ingeniería inversa. Se utiliza la herramienta *dex2jar*, utilizada en el Módulo II, que convierte los ficheros descomprimidos .dex a .jar. Seguidamente, se utiliza la herramienta *JD-GUI* para poder analizar el código fuente del .jar.

Para averiguar si la aplicación contiene código malicioso, se utilizan diferentes modelos predictivos de *machine learning*. Se utilizan técnicas de minería de datos para analizar el código, se realizan estadísticas de las clases analizadas y se buscan palabras clave en el código (*lock*, *concurrent*, etc.). Algunos de los algoritmos que se utilizan en el estudio son:

- NaiveBayes.
- K-NN.
- J48.
- RandomForest.
- SMO.

Todos los algoritmos tienen un porcentaje de precisión superior al 80 %. Sin embargo, se puede comprobar que el algoritmo que mejor se comporta para detectar código malicioso es el algoritmo K-NN. Los algoritmos se deben entrenar con las nuevas técnicas que implementen las *botnets* para realizar modelos predictivos adecuados.

## 4.2. Investigation into Google Play security mechanisms via experimental botnet

En este artículo [3], se introduce código malicioso para que el dispositivo *Android* forme parte de una *botnet* y se comprueban los mecanismos de seguridad que utiliza *Google Play* para publicar aplicaciones.

Se introducen dos componentes en la aplicación, a priori inofensivos, pero que en realidad es el código malicioso de la *botnet*. Se piden los permisos típicos (*GPS*, Internet ...) al usuario cuando hace uso de la aplicación, pero, en realidad se utilizan para recopilar información.

El dispositivo infectado recibe un *JSON* del *C&C* en el que se incluyen datos meteorológicos y variables que son utilizadas para configurar y realizar ataques. Se utiliza el algoritmo *SHA-256* para comprobar que las variables de los componentes son iguales que las que recibe del *JSON*.

La aplicación se puede publicar en *Google Play* sin ningún problema, ya que, no se detecta código malicioso. Esto se debe a que la aplicación se conecta con el *C&C* cuando recibe el *JSON*, en la llamada *onCreate()*. Se protegen los componentes maliciosos con cifrado *SHA-256* para evitar que se manipulen.

# Capítulo 5

# Revisión crítica del artículo

La lectura de este artículo nos ha resultado bastante interesante y útil, ya que trata de un tema tan importante como la seguridad en nuestros dispositivos. Como se ha mencionado en el artículo, cada vez se hace un mayor uso de los *smartphones* y esto hace que los atacantes se sientan atraídos e intenten infectar nuestros dispositivos. Que en un artículo se mencione un procedimiento a través del cual se puede infectar un dispositivo y captar información del mismo hace que el lector se sienta identificado y preste atención al contenido del mismo. Además, pese a que incluye bastante porciones de código (lo cuál le hace ser aún más técnico), la mayor parte lectura se nos ha hecho amena y fácil de entender. La estructura del artículo es bastante buena donde, tras una breve introducción al mismo, en cada apartado se explica una de las fases del ataque. Esto hace que el lector no pierda en ningún momento la referencia de en que fase del ataque se encuentra.

Como se ha mencionado antes, la mayor parte de la lectura se ha hecho amena. Sin embargo, a la hora de explicar el proceso de determinar puntos de encuentro a través del algoritmo de k-means la lectura se ha hecho algo más pesada, incluso se ha tenido que releer unas cuantas veces para entender de manera general el desarrollo del algoritmo.

A pesar de que siempre que pensamos en ataques cibernéticos (en este caso de un botnet) pensemos que es para hacer el mal, en el artículo se deja bastante claro que el objetivo de este artículo no es enseñar a realizar un ataque botnet, si no de analizar el potencial de este sistema para combatir el crimen en el mundo. Por eso, mientras se leía el artículo, no ha dado la sensación de ser un manual para interesados en realizar ataques botnets, más bien una manera de describir como nuestro dispositivo puede ser infectado y el peligro que ello conlleva.

Además, a pesar de ser un artículo publicado hace 4 años, este tipo de ataques se siguen dando en la actualidad. Cada vez hay más teléfonos móviles funcionando en el mundo, por lo tanto hay más posibilidades de infectar dispositivos a través de este método y robar información del mismo. Es por ello, por lo que nos ha parecido un artículo interesante, el hecho de que a pesar del paso del tiempo (unos pocos años en materia de seguridad informática es un mundo) sigue siendo actual y aplicable en estos días.

# Capítulo 6

# Conclusiones

# Bibliografía

[1] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[2] B. Alothman and P. Rattadilok, "Android botnet detection: An integrated source code mining approach," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 111–115, Dec 2017.

[3] M. Oulehla, "Investigation into google play security mechanisms via experimental botnet," in *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 591–596, Dec 2015.

# Capítulo 7

# Información complementaria

## 7.1. Información básica del autor

El autor de este artículo es Valentin Hamon. En el momento en el que escribió el artículo era estudiante e investigador del laboratorio de criptologia operacional y virología en ESIEA (*École supérieure d'informatique, électronique, automatique*. Está graduado como Ingeniero de Seguridad Informática de Sistemas por la ESIEA. Actualmente trabaja en FAMOCO, empresa que ha ha desarrollado la primera solución transaccional Android del mundo para ayudar a las empresas a alcanzar sus objetivos de transformación digital.

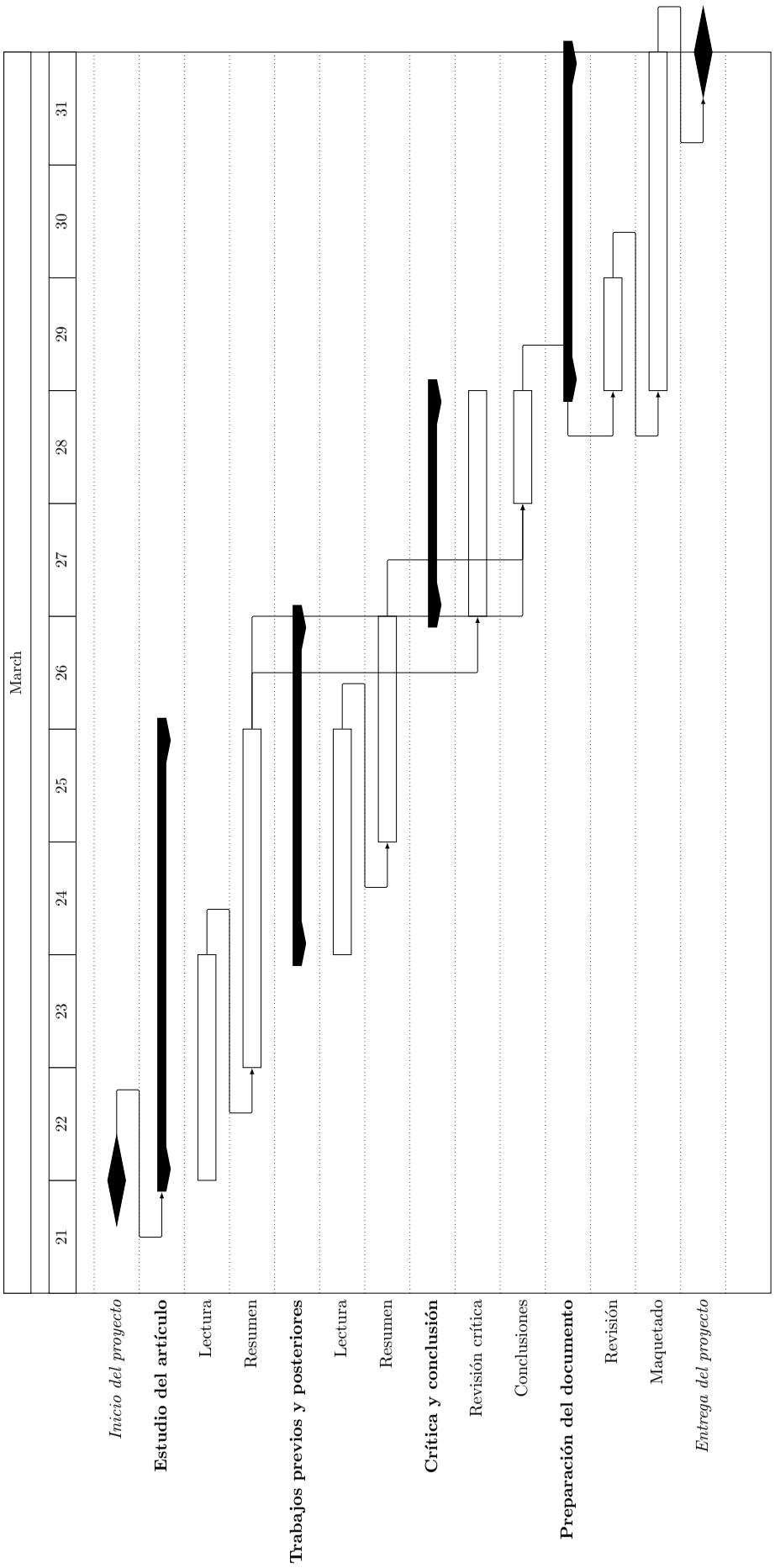
Además de este artículo, anteriormente publicó otro titulado: *Malicious URI Resolving in PDFs*, cuyo propósito es mostrar que el simple uso de una petición HTTP desde un PDF puede ser una buena herramienta para un atacante. Además, este documento trata sobre cómo puede ser relativamente fácil reutilizar algunas vulnerabilidades ajenas al documento.

## 7.2. Información básica de la revista

El artículo fue publicado en la revista *Springer-Verlag*. *Springer-Verlag* es la filial francesa de *Springer*, fundada en 1986. Su sede se encuentra en París. Los temas tratados en sus publicaciones son del ámbito de la medicina, las matemáticas, la estadística, la informática y la ingeniería. Además de la publicación de revistas de carácter científico, *Springer-Verlag* se dedica a la publicación de libros del ámbito de las ciencias y a comercializar la editorial Springer en Francia.

## 7.3. Planificación

En la siguiente página se muestra la planificación temporal del proyecto realizado, para ello, se ha utilizado un diagrama de Gantt.


March
21
22
23
24
25
26
27
28
29
30
31
Inicio del proyecto
Estudio del artículo
Lectura
Resumen
Trabajos previos y posteriores
Lectura
Resumen
Crítica y conclusión
Revisión crítica
Conclusiones
Preparación del documento
Revisión
Maquetado
Entrega del proyecto

# Glosario

**Android.** Sistema operativo para dispositivos móviles desarrollado por Google, basado en el kernel de Linux y de código abierto.

**API.** Application Programming Interface, conjunto de funciones que ofrece cierta biblioteca para ser utilizado por otro software como capa de abstracción.

**Botmaster.** Individuo o sistema responsable del control de bots.

**Botnet.** Conjunto de *bots* que se ejecutan de manera autónoma. El creador de la *botnet* puede controlar los dispositivos infectados de manera remota.

**C&C.** Command and Controle, software de control y comandos que utiliza el botmaster para controlar la red.

**DDoS.** Distributed Denial of Service, ataque de denegación de servicio que realiza múltiples peticiones hacia un único destino.

**GSM.** Global System for Mobile, sistema estándar y libre de telefonía digital.

**HLR.** Home Location Register, base de datos que contiene todos los dispositivos registrados en la red GSM.

**IMEI.** International Mobile Equipment Identity, código pregrabado en los móviles GSM que lo identifican unívocamente.

**JSON.** JavaScript Object Notation, es un formato de texto sencillo para el intercambio de datos.

**Machine learning.** Subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan.

**Phishing.** Proceso de ingeniería social que se utiliza para conseguir datos de usuarios de manera fraudulenta.

**Root.** Proceso que permite al usuario tener privilegios elevados (superusuario) para sobrepasar las limitaciones que impone el fabricante de hardware y software.

**RSA.** Sistema criptográfico de clave pública.

**Sandbox.** Aislamiento de procesos, es un mecanismo para ejecutar programas con seguridad y de manera separada.

**SHA-256.** Secure Hash Algorithm - 256, función hash de cifrado.

**SIM.** Subscriber Identity Module, almacena la clave de servicio del suscriptor usada para identificarse ante la red.