¿Computación Voluntaria o Involuntaria? Análisis y comparación de los recursos de altas prestaciones entre un sistema malicioso y uno estándar

Universidad Nacional de La Plata

María José Erquiaga
Director: Sebastián García
Codirector: Adrián Pousa

18 de diciembre de 2020

Resumen

Una botnet es un grupo de computadoras controladas remotamente y de forma simultánea. Las botnets suelen ser ilegales dado que la incorporación de computadoras (bots) se realiza sin autorización mediante un ataque a esas computadoras. En esta tesis consideramos por primera vez a las botnets como sistemas de altas prestaciones (HPC por el término en inglés *High Performance Computing*).

La computación de altas prestaciones tiene como objetivo el agregado de potencia computacional para obtener un mejor desempeño y resolver problemas que requieren gran potencia de cálculo. Para lograr este objetivo, HPC utiliza diversos métodos y tecnologías tanto a nivel de software, como de hardware e incluso de administración de recursos. Los sistemas HPC han sido estudiados profusamente y son en general bien entendidos. Por su parte las botnets son también en general bien entendidas en el ámbito de la seguridad informática, donde mayormente se estudian sus ataques, sus amenazas y como detectarlas.

El objetivo de esta tesis es estudiar las botnets como sistemas HPC. Para este propósito se extraen las características de altas prestaciones en una botnet en particular llamada Geost y se las compara con un sistema de altas prestaciones estándar conocido como es SETI@home. Esta comparación servirá pare entender mejor la potencia de las botnets en relación a los sistemas legales y tradicionales de HPC.

Con el objetivo de realizar esta comparación, se seleccionaron cinco características de la computación de altas prestaciones: resiliencia, paralelismo y concurrencia, computación heterogénea, ancho de banda y accesibilidad. Se extrajeron estas características de las dos aplicaciones, Geost y SETI@home, mediante el uso de una combinación de técnicas y herramientas de análisis de aplicaciones e inteligencia. Finalmente, los resultados de la comparación son analizados y estudiados para entender mejor el fenómeno.

Esta tesis tiene dos grandes aportes. En primer lugar, proveer la primer comparación científica de características HPC en sistemas botnet y sistemas estándar. En segundo lugar, proveer una metodología de extracción y análisis de características de la computación de altas prestaciones en botnets.

Abstract

A botnet is a network of computers controlled remotely and simultaneously by a single entity. Running a botnet is usually illegal, since the incorporation of computers (bots) is done without authorization by attacking those computers. In this thesis we consider botnets for the first time as high-performance systems. High performance computing (HPC) aims to add computational power to obtain better performance and solve problems that require high computational capacities To achieve this goal, HPC uses a variety of methods and technologies at software, hardware and even resource management levels. HPC systems have been extensively studied and are generally well understood. For their part, botnets are also generally well understood in the field of computer security, which studies their attacks, threats and how to detect them are mostly studied.

The goal of this thesis is to compare botnets as HPC systems. For this purpose, HPC features are extracted from two systems; a malicious and a standard one in order to compare them.The malicious system is a botnet called Geost, a banking trojan botnet, and its features are compared to a standard high performance system known as SETI @ home, a tool to help to detect intelligent life outside Earth. This comparison will serve to better understand the power of botnets in relation to traditional and legal HPC systems.

In order to drive this comparison, five characteristics of HPC were selected: resilience, parallelism and concurrency, heterogeneous computing, bandwidth and accessibility. These features were extracted from the two applications, Geost and SETI @ home, by using a combination of intelligence and application analysis techniques and tools. Finally, the results of the comparison are analyzed and studied to better understand both systems.

This thesis has two main contributions. The first one is to provide the first scientific study of HPC features on botnet and standard systems. Secondly, to provide a methodology for extracting and analyzing characteristics of HPC in botnets.

Agradecimientos

Quiero agradecer en primer lugar a mi familia, que me apoyo durante todo el transcurso de la carrera, no podría haber llegado hasta acá sin su apoyo incondicional.

Quiero agradecer a mis directores, a Sebastián García por ser siempre una fuerte de inspiración, motivación y por empujarme siempre a cruzar mis limites y llegar mas lejos de lo que imaginé. A Adrian Pousa por su apoyo, consejos y revisiones durante el cursado de su materia y durante el desarrollo de esta tesis.

Quiero agradecer al equipo del laboratorio de Stratosphere, en particular a Verónica Valeros y Maria Rigaki por sus valiosas revisiones, apoyo y paciencia durante el desarrollo de la tesis.

Quiero agradecer al ITIC, que me recibió durante la primer etapa de los estudios de esta maestría y donde pude participar en diversos proyectos de investigación que me permitieron crecer. A las personas que me acompañaron durante mi periodo como investigadora en el ITIC, en especial, Elina Pacini, Pablo Godoy, Lucas Iacono, Emmanuel Millan, David Monge, Rubén Santos y Yisel Gari.

A la Universidad Nacional de Cuyo, en particular a la SIIP (Secretaría de Investigación, Internacionales y Posgrado), que me otorgo una beca de ayuda económica durante el cursado de la maestría, y una beca para realizar la estancia en el extranjero que me permitió comenzar esta tesis. También a la Facultad de Ingeniería de la UNCuyo, que me otorgo la ayuda económica para llevar a cabo el proyecto PIN para investigadores noveles, a cargo de mi director Carlos García Garino, a quien le estoy agradecida por haberme dado el espacio para comenzar mis investigaciones en el ITIC.

La última parte de esta tesis se desarrolló en el marco del proyecto Aposemat, un proyecto entre la CVUT y Avast Software, Praga. Agradezco a ambas instituciones por darme la oportunidad de ser parte de este proyecto

y darle un marco a la tesis.

Finalmente, quiero agradecer a todo el equipo administrativo de la Universidad Nacional de La Plata que siempre estuvieron atentos a mis consultas e inquietudes durante mi cursado y desarrollo de tesis, en particular a Natalia Otero y Alejandra Pizarro.

Índice general

1. Introducción **5**
- 1.1. Hipótesis de Trabajo 7
- 1.2. Metodología de trabajo 8
- 1.3. Aportes de la tesis 8
- 1.4. Estructura de la Tesis 9

2. Marco Teórico **10**
- 2.1. Computación de Altas Prestaciones 10
  - 2.1.1. Ejemplos Reales de HPC 11
  - 2.1.2. Resiliencia 12
  - 2.1.3. Disponibilidad 12
  - 2.1.4. Paralelismo y Concurrencia 13
  - 2.1.5. Computación Heterogénea 14
  - 2.1.6. Ancho de banda 15
- 2.2. Botnets 15
- 2.3. Arquitectura Android 16
  - 2.3.1. Paquete de Aplicaciones Android (APK) 18
  - 2.3.2. Estructura de los APKs 19
- 2.4. Herramientas utilizadas para el análisis 20
  - 2.4.1. VirusTotal y RiskIQ 20
  - 2.4.2. tcpdump 21
  - 2.4.3. Herramientas para el análisis de los APK 22

3. Botnet Geost **25**
- 3.1. Descripción de la Botnet Geost 25
- 3.2. Experimento de Ejecución de Geost 27
- 3.3. Características de altas prestaciones en Geost 29
  - 3.3.1. Resiliencia 29

3.3.2. Accesibilidad y disponibilidad . . . 31
3.3.3. Paralelismo y concurrencia . . . 33
3.3.4. Computación heterogénea . . . 34
3.3.5. Ancho de Banda . . . 35

4. Aplicación SETI@home **38**
4.1. Aplicación estándar: BOINC . . . 38
4.1.1. SETI@home . . . 40
4.2. Resiliencia . . . 43
4.3. Accesibilidad y disponibilidad . . . 44
4.4. Paralelismo y concurrencia . . . 44
4.5. Computación heterogénea . . . 45
4.6. Ancho de Banda . . . 45

5. Comparativa y Análisis **48**
5.1. Resiliencia . . . 48
5.2. Accesibilidad . . . 49
5.3. Paralelismo y concurrencia . . . 50
5.4. Computación heterogénea . . . 50
5.5. Ancho de banda . . . 51
5.6. Discusión de los resultados . . . 52

6. Conclusiones **56**

Capítulo 1

Introducción

Una botnet es un sistema computacional de tipo distribuido que consta de varios nodos, llamados bots, interconectados entre si y actuando de forma simultánea [45]. Este tipo de sistemas puede ser utilizando para fines ilícitos, como envío de SPAM y ataques de denegación de servicio. Usualmente la comunidad considera toda botnet como maliciosa, aunque la ilegalidad de las botnets no es un requerimiento necesario para su funcionamiento. Aunque existen botnets benignas, por simplicidad esta tesis utiliza el término botnet como referencia a las botnets maliciosas.

Una de las ideas fundamentales de esta tesis es estudiar las botnets como sistemas HPC. Una de las razones de esta afirmación es que la infraestructura de varios nodos interconectados se la puede ver como un sistema de cómputo, es decir como una gran computadora [22]. Cada botnet puede contener cientos, miles y hasta millones de bots, por lo que tiene un gran potencial de intensidad computacional. Cada nodo por lo general tiene diversas características a nivel computacional tales como diferentes tipos de CPU, memoria RAM, sistemas operativos, etc. Esta visión y análisis de las botnets como un sistema HPC no ha sido planteada hasta el momento.

Si se define a un sistema HPC estándar como cualquier aplicación de HPC sin fines maliciosos, hasta el momento de presentar esta tesis no hay trabajos en los que se compare los sistemas HPC estándar con botnets. Es decir, no hay estudios donde se analice la potencia computacional de las botnets y se la compare con un sistema HPC estándar.

Las botnets son sistemas muy bien estudiadas en el ambiente de la seguridad informática en general. Sin embargo, la mayoría de los estudios se enfocan en sus capacidades para atacar, sus arquitecturas de comunicación

y sus acciones en ataques concretos, dejando de lado el aspecto menos malicioso de las botnets en referencia a cuan grande y eficiente es su poder de cómputo. Esto motiva a realizar un estudio de la utilización de los recursos computacionales por parte de las botnets con el fin de entender su capacidad en comparación con otros sistemas de cómputo. Es posible que no se haya estudiado este aspecto *no malicioso* de las botnets porque son sistemas ilegales por lo que no son vistas como sistemas de gran potencia computacional, sino como una amenaza para los consumidores de tecnología.

Un clásico estudio de las botnets en el ámbito de la seguridad informática se enfoca en su capacidad de realizar ataques de fuerza bruta [13]. Este tipo de ataques requieren de una gran carga computacional, de ancho de banda y cantidad de bots para lograr este objetivo de forma eficiente. Los atacantes hacen uso de la botnet de forma distribuida y ordenan a los bots a actuar en simultaneo para saturar un sistema víctima. La motivación detrás de este tipo de ataques es por un lado disminuir la carga computacional de la víctima o su disponibilidad. La mayoría de las versiones de este tipo de ataque ocultan el verdadero origen de los ataques ya que las peticiones se realizan desde diferentes dispositivos con diferentes direcciones IP.

Otro aspecto clásico del análisis de botnets se refiere al estudio de su rendimiento desde sistemas simulados [14]. Sin embargo, dado que la simulación se realizo en un único sistema de computo de gran capacidad usando computadoras virtuales, este trabajo no puede analizar ciertos aspectos fundamentales de las botnets, como su resiliencia o su consumo del ancho de banda.

Otro trabajo orientado al análisis de los modelos de las botnets es [51]. En este caso, se recalca que las botnets crecen en tamaño y complejidad a medida que incrementan y evolucionan los nodos debido al crecimiento de nuevas tecnologías de Internet de las cosas (IoT por sus siglas del inglés: Internet of Things) o al aumento en el consumo de redes sociales y otras tecnologías. En la mayoría de los casos, la arquitectura de las botnets se estima que es descentralizada.

Hasta donde sabemos, esta tesis considera por primera vez a las botnets como sistemas de altas prestaciones (HPC). Hasta la fecha, no hemos encontrado trabajos de investigación que consideren a las botnets en su capacidad de controlar remotamente una gran potencia computacional, ni que estudie cuan potente puede ser una botnet en relación a un sistema HPC que se utilice, por ejemplo, en un centro académico. Tampoco sabemos de ningún estudio hasta el momento sobre como las botnets se podrían comparan con

sistemas HPC *estándar*, ni tampoco sobre si, como esta tesis hipotetiza, las botnets podrían ser mejores sistemas HPC que un sistema HPC estándar.

La hipótesis que se plantea en esta tesis es que las botnets pueden ser consideradas y estudiadas como sistema de HPC y pueden resolver problemas de forma similar. Para demostrar la hipótesis se compara determinadas características de los sistemas HPC en las botnets y en los sistemas estándar HPC. Para alcanzar el objetivo, se extraen las características de la computación de altas prestaciones en un sistema malicioso (botnet Geost) y un sistema estándar (SETI@home). Durante el desarrollo de todo este documento, se hace referencia a los sistemas de altas prestaciones como HPC (por sus siglas *High Performance Computing*).

1.1. Hipótesis de Trabajo

La hipótesis de esta tesis es que las botnets poseen una gran potencia computacional similar a los sistemas de HPC estándar. En esta sub-sección se detalla la metodología de trabajo a desarrollar durante esta tesis. Para demostrar esta hipótesis, se describirán las características de un sistema de HPC tradicional y los de una botnet para compararlos. En este sentido, se tendrán en cuenta, las siguientes características:

- Resiliencia (tolerancia a fallos)
- Paralelismo y concurrencia
- Computación heterogénea
- Ancho de banda
- Accesibilidad

Mediante el estudio de las características de HPC en botnets se pretende, por un lado, medir las características de las botnets como sistemas de altas prestaciones y por el otro, estudiar qué características se pueden obtener de una botnet mediante el procesos de Inteligencia de amenazas (llamado en inglés *Threat Intelligence*) y de ingeniería reversa de aplicaciones maliciosas. El proceso de Inteligencia de amenazas incluye técnicas de análisis de dominios, direcciones IPs, comportamiento de red de las aplicaciones maliciosas. Por otro lado el proceso de Ingeniería reversa, está orientado al análisis del

código de la aplicación. En el caso particular de Geost, se estima que la botnet infectó aproximadamente 800,000 víctimas.

1.2. Metodología de trabajo

Para realizar la comparación de las botnets con los sistemas HPC estándar se siguieron varios pasos. Primero, se seleccionó una botnet a ser estudiada, la botnet se llama Geost y es de tipo *trojano bancario*. Esta botnet afecta a teléfonos celulares con sistema operativo Android. Luego se buscaron los dominios, las direcciones IP y los APKs (acrónimo de Paquete de Aplicación Android, del ingles *Android Application Package*) que estaba siendo utilizadas por esta botnet. Se analizaron los APKs mediante la ejecución de las aplicaciones en emuladores, teléfonos reales y analizando el código, es decir, por medio de ingeniería inversa. Con la información obtenida, se extrajeron las características HPC seleccionadas (Resiliencia, Paralelismo y concurrencia, Computación heterogénea, Ancho de banda y Accesibilidad). Por otro lado, la aplicación estándar para realizar el estudio comparativo es SETI@home. Esta aplicación es un modelo de computación voluntaria, que pretende resolver un problema de gran escala dividiéndolo en pequeñas porciones y utilizando la potencia computacional de los usuarios. Se identificó los dominios, las direcciones IP y el APK utilizados por SETI@home. De la misma forma, extrajeron las características de HPC para esta aplicación. Finalmente se realizo una comparación entre las características de Geost y SETI@home.

1.3. Aportes de la tesis

Los aportes de esta tesis son:

- Proporcionar un estudio comparativo entre sistemas normales de altas prestaciones y botnets mediante la extracción y el análisis de las características de éstos sistemas.
- Una medición analítica aproximada de las diferencias entre sistemas de altas prestaciones y botnets con el objetivo de compararlos.
- Una metodología de estudio de APK de Geost para estudiar sus características de altas prestaciones

- Análisis de las características de los APK de Geost mediante ingeniería reversa

1.4. Estructura de la Tesis

Esta tesis se estructura de la siguiente forma, en el capítulo 2 se describe el marco teórico de esta tesis. Se define computación de altas prestaciones y las características que se tienen en cuenta para el análisis de las aplicaciones maliciosas y estándar. Luego se definen y se explica el funcionamiento de las botnets. En este capítulo también se describe la aplicación estándar que será analizada en la tesis para luego ser comparada con la aplicación maliciosa. Finalmente se describen las herramientas utilizadas para el análisis. En el capítulo 3 se describen y analizan las características de altas prestaciones en la botnet Geost. En el capítulo 4 se describen las características de altas prestaciones de la aplicación SETI@home. En el capítulo 5 se realiza un análisis y comparativa de los datos obtenidos en las aplicaciones Geost y SETI@home. Finalmente, en el capítulo 6 se presentan las conclusiones finales del trabajo.

Capítulo 2

Marco Teórico

El objetivo de esta tesis es realizar un estudio comparativo entre las características de altas prestaciones de un sistema malicioso y un sistema estándar. En este capítulo se define computación de altas prestaciones (HPC) y se describen las características que se tendrán en cuenta para el análisis. Luego se define el concepto de botnet. También, dado que los dos sistemas que se van a analizar (Geost y SETI@home), funcionan en Android, se detalla la arquitectura Android. Finalmente, se describen las herramientas utilizadas para extraer las características de HPC de Geost y de SETI@home.

2.1. Computación de Altas Prestaciones

La computación de altas prestaciones (HPC por sus siglas del inglés, *High-Performance Computing*) hace referencia a cualquier forma de computación donde la densidad de procesamiento o el tamaño de los problemas abordados requieren más de un sistema informático estándar o básico para lograr el resultado esperado bajo las restricciones dadas. Además tiene en cuenta la aplicación de técnicas avanzadas como el uso de múltiples procesadores (decenas, cientos, miles) conectados entre sí por algún tipo de red para lograr un rendimiento superior al de un único procesador [18]. La tecnología HPC abarca la implementación de un algoritmo dado (o código) y el hardware en el que se ejecuta un algoritmo [22]. En esta tesis se utiliza el acrónimo HPC para hacer referencia a la computación de altas prestaciones.

HPC es un área de ciencias de la computación bastante amplia que evoluciona constantemente con el objetivo de obtener más potencia computacional

para resolver diferentes problemas. El crecimiento en este área se ha visto potenciado en los últimos años debido al desarrollo de nuevas tecnologías como Cloud Computing y a la generación masiva de datos, lo cual trae nuevos campos de estudio como Big Data [18].

HPC cuenta con varias implementaciones, entre las mas conocidas se encuentran cluster, grid y cloud computing. Cluster se define como una colección de computadoras que están interconectadas por medio de una red [12]. Las computadoras que integran el cluster pueden tener las mismas características (en ese caso es un cluster homogéneo o Beowulf), o diferentes características (cluster heterogéneo). Grid computing se define como un sistema que coordina recursos distribuidos utilizando protocolos para entregar servicios [26]. Cloud Computing hace referencia a las aplicaciones entregadas, servicios a través de Internet y al hardware y software de sistemas en los centros de datos que proporcionan esos servicios [53].

En esta tesis se analiza el uso de los recursos computacionales de dos aplicaciones HPC, una maliciosa y un estándar. Para este análisis se tienen en cuenta ciertas características de HPC. Estas son: resiliencia, accesibilidad y disponibilidad, paralelismo y concurrencia, computación heterogénea y ancho de banda. A continuación se describen ejemplos de HPC y se detallan cada una de las características de HPC que se analizan durante el desarrollo de la tesis.

2.1.1. Ejemplos Reales de HPC

El crecimiento y la evolución de HPC se ven reflejados en la cantidad de herramientas y aplicaciones al alcance de la comunidad. Un ejemplo es el caso del desarrollo de TensorFlow [48]por parte de Google. TensorFlow es una librería open-source utilizada principalmente en algoritmos de aprendizaje de máquinas y permite hacer uso de varios CPUs y GPUs, también funciona en teléfonos celulares (*smartphones*). Otro ejemplo es la incorporación de la computación de propósito general en unidades de procesamiento gráfico, o *GPGPU* por su siglas en ingles, en teléfonos celulares. Además, el CERN (European Organization for Nuclear Research), la Organización Europea para Investigación de Física Nuclear, hace uso de HPC. El CERN provee recursos computacionales a nivel global para el almacenamiento, distribución y análisis de los datos generados por el el acelerador de partículas más grande a nivel mundial (Large Hadron Collider, conocido por su acrónimo LHC). El proyecto llamado Worldwide LHC Computing Grid (WLCG),

Grid Computacional a nivel mundial para LHC, combina recursos de 900,000 núcleos de computadoras de todo el mundo [52].

2.1.2. Resiliencia

Una de las definiciones de resiliencia es: *la capacidad del sistema para reaccionar ante perturbaciones, fallas internas y eventos ambientales mediante la absorción de la perturbación y / o la reorganización para mantener sus funciones* [2].

Los métodos que existen para que un sistema sea resiliente pueden incluir soluciones a nivel de hardware, sistemas (como códigos, ya sea de paridad, de chequeo por suma, cíclicos entre otro, sistemas de discos resilientes: RAID, replicación de datos), redes, software y sistemas de checkpoint [27]. La estructura RAID, de las siglas en inglés *Redundant Arrays of Independent Disks*, en español Arreglos Redundantes de Discos Independientes, es un método de redundancia de datos y tiene 5 niveles diferentes para codificar los datos y evitar perdidas de información [36]. Otro método usa sistemas de *checkpoint*. Un checkpoint que es un *snapshot*, esto es una imagen del estado completo de un proceso en un momento determinado. Es decir, que permite representar toda la información que necesitaríamos para reiniciar el proceso desde ese punto [36].

2.1.3. Disponibilidad

La disponibilidad es la medida en que un sistema está disponible para ser utilizado, incluso teniendo en cuenta que puede no funcionar correctamente [44], es decir, que tiene que ver con la probabilidad de que un sistema esté operativo en el momento en que se lo necesite. La disponibilidad se mide desde la perspectiva del usuario. Es decir, un sistema está disponible cuando las aplicaciones están disponibles para los usuarios [33]. Los sistemas de alta disponibilidad pueden notificar la disponibilidad en términos de minutos u horas de tiempo de inactividad por año.

Las características de disponibilidad permiten que el sistema permanezca operativo incluso cuando se producen fallas. Un sistema altamente disponible deshabilitaría la sección que presente un mal funcionamiento y continuaría operando a una capacidad reducida. Por el contrario, un sistema menos disponible, podría fallar y volverse totalmente no operativo.

La disponibilidad es una medida en relación al tiempo en el que un servidor esta funcionando con normalidad. Una de las formas para calcular esa medida es utilizando la siguiente ecuación [30]:

$$A = \frac{MTBF}{MTBF + MTTR} \tag{2.1}$$

En la Ecuación 2.1, A es el grado de disponibilidad expresado como porcentaje. $MTBF$ es el tiempo intermedio entre fallas (*Meant Time Between Failures*), y $MTTR$ es el tiempo máximo para solucionar un problema particular (*Maximum Time To Repair*). Entonces, mientras $MTTR$ se aproxime a cero, A incrementa al 100 por ciento, mientras $MTBF$ incrementa, $MTTR$ tiene menos impacto en A. Por ejemplo, si un sistema tiene un valor de MTBF de 100,000 horas y un MTTR de 1 hora, tiene una disponibilidad de 99.999 %.

2.1.4. Paralelismo y Concurrencia

La computación paralela es un tipo de computación en la que varios cálculos o la ejecución de varios procesos se llevan a cabo de forma simultánea[3]. Este tipo de computación se utiliza para resolver grandes problemas que pueden ser divididos en fracciones mas pequeñas y ejecutadas al mismo tiempo.

Una aplicación concurrente contiene dos o más procesos que se ejecutan de forma simultanea. Cada proceso es una secuencia de instrucciones que se ejecutan una tras otra [11] [8]. Mientras que un programa secuencial tiene un único flujo de instrucción, un programa concurrente tiene múltiples flujos de instrucciones. Los procesos de un programa concurrente pueden trabajar juntos comunicándose entre sí. La comunicación se programa usando variables compartidas o el paso de mensajes. Cuando se utilizan variables compartidas, un proceso escribe en una variable que es leído por otro. Cuando se utiliza el paso de mensajes, un proceso envía un mensaje que es recibido por otro [8]. Una aplicación paralela es una aplicación concurrente donde los procesos se ejecutan sobre varios procesadores con el objetivo de resolver un problema en menor tiempo que un algoritmo secuencial.

Independientemente de la forma de comunicación, los procesos también necesitan sincronizarse entre sí. Hay dos niveles básicos de sincronización: la exclusión mutua y la sincronización de la condición. La exclusión mutua es el problema de asegurar que las secciones críticas de las declaraciones no se

ejecutan al mismo tiempo. La sincronización de condiciones es el problema de retrasar un proceso hasta que una condición dada sea verdadera [8].

2.1.5. Computación Heterogénea

La computación heterogénea hace referencia a los sistemas que utilizan más de un procesador para alcanzar mejores prestaciones. En este sentido, no es necesario que todos los procesadores sean iguales [43]. Los sistemas de computación heterogénea se pueden encontrar en diferentes situaciones. Uno de los casos se presenta en los sistemas distribuidos que contienen varios dispositivos con diferentes características. Esto se logra mediante la conexión de dos o más dispositivos con procesadores de diferentes características para formar clusters o grids. También aparece cuando un dispositivo contiene diferentes procesadores. Además, el concepto de computación heterogénea no solo hace referencia al hardware, si no también al Sistema Operativo, lenguajes de programación y aplicaciones.

Los dispositivos son heterogéneos cuando tienen dos o más CPUs diferentes o varias GPUs incorporadas en un único dispositivo. Hay aplicaciones y algoritmos especialmente diseñados para hacer uso de los CPUs y el GPU de los sistemas de forma óptima[39].

También se dice que un dispositivo es heterogéneo cuando hay diferentes tipos de núcleos en un mismo chip del procesador, este es el caso del procesador *Cell Broadband Engine de IBM* [21]. En este procesador, los distintos núcleos tienen un repertorio de instrucciones disjunto (llamado *heterogenous-ISA*, ISA heterogéneo). Sin embargo, la existencia de núcleos con distinto ISA (por sus siglas *Instruction Set Architecture*, conjunto de instrucciones) acarrea desafíos para el programador, como tener que compilar una misma aplicación para cada tipo de núcleo.

Los multiprocesadores de chips heterogéneos, también llamados asimétricos o *AMP* (por su siglas del inglés asymmetric multicore processors), presentan oportunidades únicas para mejorar el rendimiento del sistema. La heterogeneidad en el chip permite adaptar los recursos de ejecución dependiendo de las necesidades de cada aplicación [35].

Los procesadores AMP con el mismo repertorio de instrucciones (*single-ISA*), combinan las altas prestaciones de núcleos potentes con el bajo consumo de núcleos pequeños. Estos procesadores han alcanzado altas prestaciones a nivel de consumo energético a comparación de los chips de múltiples

procesadores convencionales. Este potencial se puede obtener mediante planificaciones a nivel del sistema operativo[34].

2.1.6. Ancho de banda

El ancho de banda es una medida de la cantidad de datos transmitida en un tiempo determinado [29]. Esta medida se expresa en bit/s o múltiplos de él como serían los Kbit/s, Megabit/s y Gigabit/s o bytes/s, Kilobytes/s, etc. [7].

En [22] se hace hincapié en que la sobrecarga de comunicación puede tener impacto en el rendimiento de la aplicación. Las características de la red que conecta las unidades de ejecución, procesadores, nodos de cómputo, o lo que sea que desempeñe un papel dominante en este caso. Una gran variedad de tecnologías de red y topologías están disponibles para mejorar las prestaciones de las redes y disminuir la sobrecarga de comunicación.

2.2. Botnets

Las Botnets son un tipo de sistema que constan de una red de dispositivos interconectados. Los dispositivos conectados ejecutan un programa autómata que es controlado por uno o varios nodos. A los equipos que son parte de la botnet se les denomina bots o zombies [17]. A los nodos de control se los denomina servidores de comando y control, *Command and control*, ahora en adelante llamados C&C. Estos nodos de control son operados por individuos llamados botmasters.

Este tipo de sistemas puede ser malicioso cuando se utiliza con fines ilícitos (por ejemplo robo de credenciales, envío de SPAM, ataques de denegación de servicio, minado de bitcoins sin consentimiento de las víctimas, etc).

Las botnets tienen por lo menos un botmaster y varios bots distribuidos en diferentes dispositivos (computadoras, teléfonos, tablets, etc). Se denomina botmaster a la persona que posee y administra la botnet, puede ser una persona o varias personas. Los botmasters tienen control de los bots y les pueden enviar órdenes usando diferentes protocolos de comunicación.

La arquitectura de las botnets puede ser centralizada o descentralizada como se ilustra en la figura 2.1. Para controlar a los bots que son parte de la red, los botmasters tienen servidores desde donde dan las órdenes a los bots. Estos servidores se los denomina *Command and control*, ahora en

adelante llamado C&C. Estos servidores pueden usar diferentes protocolos, como HTTP (HyperText Transfer Protocol), HTTPs (HyperText Transfer Protocol Secure) o IRC (Internet Relay Chat) entre otros.

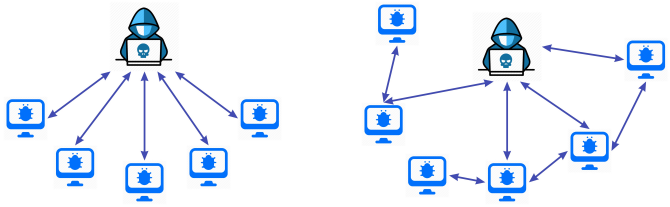

Figura 2.1: Arquitecturas de botnets centralizada a la izquierda y descentralizada a la derecha

Según el informe de ENISA (de las siglas en inglés *European Union Agency for Network and Information Security*, Agencia de la Unión Europea para la Seguridad de las Redes y la Información) de 2018, las botnets son una amenaza que está activa y responden a diferentes actividades maliciosas [32]. En el reporte de ENISA se enumeran varios tipos de botnets que están activas y se explica su evolución.

2.3. Arquitectura Android

En esta tesis se analizan aplicaciones que funcionan sobre el sistema operativo Android. En la Figura 2.2, se puede observar el modelo de capas de la arquitectura Android [16]. Las capas son:

Capa Kernel de Linux

Esta capa es la que se encuentra en el nivel más bajo del sistema operativo y está basada en el núcleo de Linux. Esta capa se encarga de la administración del hardware, la red, acceso al sistema de archivos y a los procesos. Esta capa es la que más cerca se encuentra del hardware.

Capa nativa de espacio de usuario

Sobre la capa Kernel de Linux, se encuentra la capa nativa de espacio de usuario, que consta del programa *init* (el primer proceso iniciado, que inicia todos los demás procesos), varios demonios nativos y unos cientos de bibliotecas nativas que se utilizan en todo el sistema. La mayor parte de

Android se implementa en Java y, como tal, se ejecuta mediante una máquina virtual Java (también conocida por sus siglas JVM: por su acrónimo del inglés *Java Virtual Machine*).

Capa de tiempo de ejecución Dalvik

La implementación actual de máquina virtual Java de Android se llama *Dalvik* y es la siguiente capa en la pila del modelo de Android. Dalvik fue diseñado teniendo en cuenta los dispositivos móviles y no puede ejecutar el bytecode de Java (archivos con extensión *.class*) directamente: su formato de entrada nativo se llama Dalvik Executable (DEX) y está empaquetado en archivos *.dex*. A su vez, los archivos *.dex* se empaquetan dentro de las bibliotecas Java del sistema (archivos con extensión JAR) o dentro de las aplicaciones de Android (archivos APK).

Librerías Java de tiempo de ejecución

Las librerías de Java sirven para la implementación del lenguaje Java. Estas librerías están definidas en los paquetes *java.** y *javax.**. Se accede a la capa de librerías de tiempo de ejecución de Java tanto desde los servicios del sistema como desde las aplicaciones.

Servicios del sistema

Los Servicios del sistema se encargan de implementar la mayoría de funciones de Android, incluidas el soporte para display y pantalla táctil, servicio de telefonía y conectividad de red. La comunicación entre procesos (IPC por sus siglas del ingles: Inter Process Communication) se realiza a través de Binder, que es un mecanismo de IPC.

Abstracciones de IPC de nivel superior en Android, como *Intents* (comandos con datos asociados que se entregan a los componentes a través de los procesos), *Messenger* (objetos que permiten la comunicación basada en mensajes a través de los procesos) y *ContentProviders* (componentes que exponen una interfaz de gestión de datos entre procesos) están construidos sobre Binder.

Librerías Android

Lo siguiente en la pila son las librerías Android (Android Framework Libraries), que incluyen todas las librerías Java que no forman parte del *runtime* estándar de Java (*java.**, *Javax.**, etc.) y es para la mayor parte alojada bajo

el paquete de nivel superior de Android. Esta capa incluye los bloques básicos para crear aplicaciones de Android, como las clases base para actividades, servicios y proveedores de contenido (en los paquetes *android.app.**); Widgets GUI (en los paquetes *android.view.** y *android.widget*); y clases para acceso a archivos y bases de datos (principalmente en los paquetes *android.database.** y *android.content.**). También incluye clases que le permiten interactuar con el hardware del dispositivo, así como clases que aprovechan los servicios de alto nivel ofrecidos por el sistema.

Aplicaciones Android

Finalmente, en el nivel más alto se encuentran las aplicaciones, que son los programas con los que los usuarios interactúan directamente.

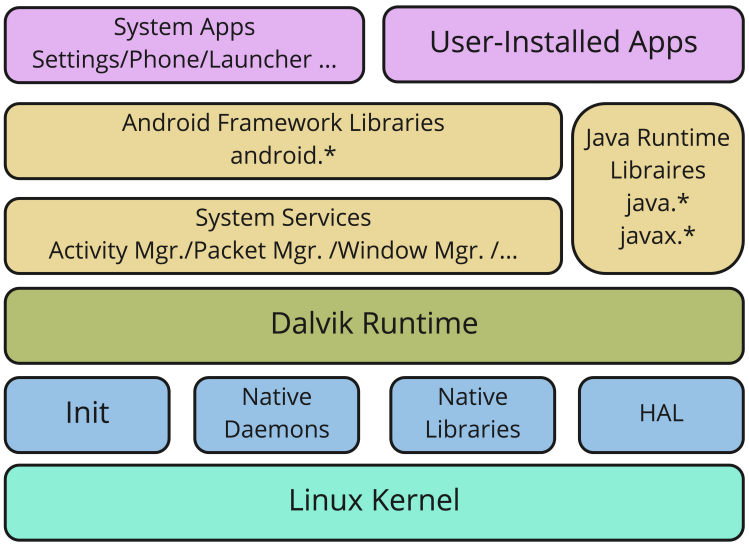

Figura 2.2: Diagrama de las capas de la arquitectura del Sistema Operativo Android

2.3.1. Paquete de Aplicaciones Android (APK)

Las aplicaciones de Android se distribuyen e instalan en forma de archivos de paquete de aplicaciones (APK por sus siglas del inglés Android Package), que generalmente se denominan archivos APK. En esta tesis se hará referencia a estos archivos como APK. Los archivos APK son contenedores que

incluyen código de aplicación y recursos, así como el archivo de manifiesto de la aplicación. También pueden incluir una firma de código. El formato APK es una extensión del formato Java JAR, que a su vez es una extensión del popular formato de archivo ZIP. Por lo que los APKs se pueden descomprimir para analizar su contenido.

2.3.2. Estructura de los APKs

Cada archivo APK incluye:

- ***AndroidManifest.xml***: En este archivo se declara el nombre del paquete, la versión, los componentes y otros meta datos de la aplicación.
- ***classes.dex***: Es un archivo que contiene el código ejecutable de la aplicación y está en el formato DEX nativo de la máquina virtual Dalvik.
- ***Resources.arsc***: Este archivo incluye todos los recursos compilados de la aplicación, como cadenas y estilos.
- ***assets***: Este directorio se utiliza para agrupar archivos de activos sin procesar con la aplicación, como fuentes o archivos de música.
- ***lib***: Es un directorio que contiene las librerías que son utilizadas por las aplicaciones que hacen uso de las librerías nativas. Este directorio tiene subdirectorios para cada arquitectura de plataforma compatible.
- ***res***: En este directorio se almacenan los recursos a los que se hace referencia directamente desde el código de Android, ya sea utilizando directamente la clase *android.content.res.Resources* o indirectamente a través de la API de nivel superior. Esto se almacena con directorios separados para cada tipo de recurso (animaciones, imágenes, menú definiciones, etc.)
- ***META-INF***: En este directorio se aloja el archivo de manifiesto del paquete y las firmas de código.

Muchas veces los desarrolladores utilizan un método llamado ofuscación, que pretende modificar un código par que no sea comprensible para los humanos. Los programadores pueden ofuscar código deliberadamente para ocultar

su propósito o la lógica de programación, para evitar la manipulación o disuadir el análisis de ingeniería inversa. Existen varias técnicas comunes de ofuscación utilizadas por aplicaciones Android, incluido el cambio de nombre de identificador, cifrado de cadena, reflexión, etc. [15].

2.4. Herramientas utilizadas para el análisis

Para analizar y comparar las botnets contra los sistemas estándar de altas prestaciones, se extraen las características que se describieron la sección 2. Con respecto al análisis de Geost, este consiste en varias etapas. De una forma simplificada, se lo puede dividir en dos etapas. Una etapa consiste en el análisis de la botnet como sistema y la otra es el análisis de las aplicaciones falsas que son parte de la infección de Geost. Las aplicaciones son los archivos APK, cuyo formato se describió en 2.3.

La primer etapa incluye el análisis de los dominios, direcciones IPs, conexiones y personas involucradas. Las principales herramientas que se usan en esta primer instancia son: Virus Total y RiskIQ. La segunda etapa tiene que ver con el análisis de las características de las aplicaciones Android. Para esto se analizaron las aplicaciones con varias herramientas que se describen en este capítulo: MobSF, jadx, Genymotion, APKlab.

2.4.1. VirusTotal y RiskIQ

Para poder analizar amenazas, aplicaciones maliciosas, dominios e IPs, se necesitan herramientas que permitan observar la conexión entre ciertos elementos, como por ejemplo, qué dominios o IPs fueron contactados por ciertas aplicaciones maliciosas. Esto permite conectar los elementos para poder comprender cómo funcionan los sistemas maliciosos y conocer la infraestructura subyacente en éstos sistemas. Dos de las herramientas más poderosas para este tipo de análisis son VirusTotal y RiskIQ.

VirusTotal [50] es una herramienta gratuita muy utilizada por analistas de seguridad informática y de análisis de malware. Esta herramienta analiza ciertos elementos (dominios, direcciones IP, programas, código, etc) cargados por los usuarios e inspecciona los elementos con más de 70 antivirus y servicios de listas negras de URL, dominios e IPs.

RiskIQ [37] permite realizar un análisis de los dominios y las direcciones IP en el tiempo. A través de un sistema de recolección pasiva de nombres

de dominio permite consultar en detalle cuándo fue la primera y la ultima vez que un dominio fue resuelto y por las direcciones IPs que lo resolvieron. Muestra en detalle el registro de la IP (nombre de los servidores, persona que lo creó, fechas, etc). Se puede observar los certificados asociados a esa dirección IP (si es que hay alguno), los subdominios y los archivos relacionados. Un ejemplo de esta herramienta se muestra en la Figura 2.3.

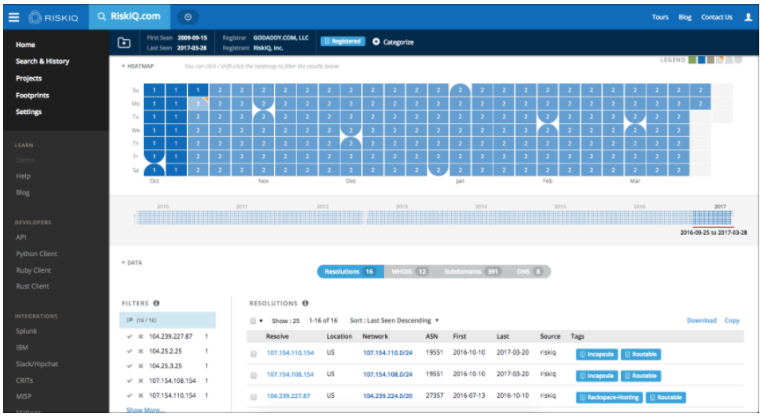

Figura 2.3: La herramienta PassiveTotal de RiskIQ permite observar el uso de un nombre de dominio y todas las direcciones IP con las que estuvo asociado [37].

2.4.2. tcpdump

tcpdump [47] es una herramienta para capturar y analizar tráfico de red mediante línea de comando. Esta herramienta permite examinar los paquetes que se transmiten en la red. Es posible realizar filtrado de paquetes, de esta forma se pueden ver los paquetes que acceden a determinados puertos, o hacia una IP destino, desde una IP origen, etc. Esta herramienta fue utilizada para analizar el tráfico de red cuando las aplicaciones que se analizan en esta tesis fueron ejecutadas. En la Figura 2.4 se ilustra un ejemplo de utilización de la aplicación tcpdump, el tráfico de red del dispositivo fue capturado y analizado con tcpdump. En el ejemplo se puede observar que el dispositivo infectado con Geost se comunica con la dirección IP `173.231.184.52` en el puerto `80`.

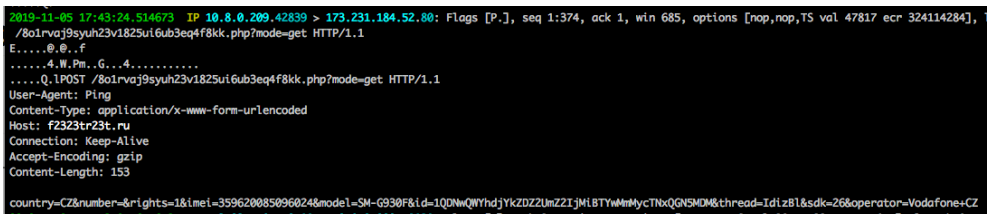

Figura 2.4: Captura del tráfico de red mientras se ejecutó una muestra de Geost en un teléfono celular.

2.4.3. Herramientas para el análisis de los APK

Mediante el proceso de *Threat Intelligence*, se descubrieron los dominios, direcciones IPs y los archivos (APKs) utilizados por esta botnet. Esto se realizó mediante un procesos de *pivoteo* de dominios y direcciones IPs utilizando VirusTotal. Una vez que se encontraron las muestras de las aplicaciones relacionadas con Geost, se analizaron los APKs para obtener las características que se describieron en 2. Para realizar el análisis, se utilizaron las herramientas que se describen a continuación.

MobSF

Mobile Security Framework (MobSF) es un sistema para realizar pruebas de vulnerabilidades aplicaciones móviles (Android, iOS y Windows) de forma automática. Esta aplicación es de código abierto y está disponible en github [31]. Se puede usar para un análisis de seguridad de forma rápida de aplicaciones móviles de Android. Esta herramienta se puede instalar en cualquier servidor o computadora y tiene una interfaz gráfica de usuario a la que se puede acceder mediante cualquier navegador web. El servicio web consiste en un tablero que presenta los resultados del análisis, su propio sitio de documentación, un emulador integrado y una API que permite a los usuarios activar el análisis automáticamente. El servicio está alojado en un entorno local, por lo que los datos son confidenciales, no interactúan con conexiones a internet.

Esta aplicación se instala en un servidor local y se accede mediante navegador web. Se pueden subir las muestras a analizar como se muestra en la Figura 2.5. Una vez que se selecciona el archivo APK a analizar, la aplicación realiza un análisis y muestra información de esa muestra, como el icono de la aplicación, el nombre del archivo, el tamaño, el nombre del paquete, la ver-

sión Android, la versión del código, análisis de seguridad, análisis de malware, etc. En la Figura 2.6 se puede observar los datos que están disponibles para el análisis en la columna derecha.

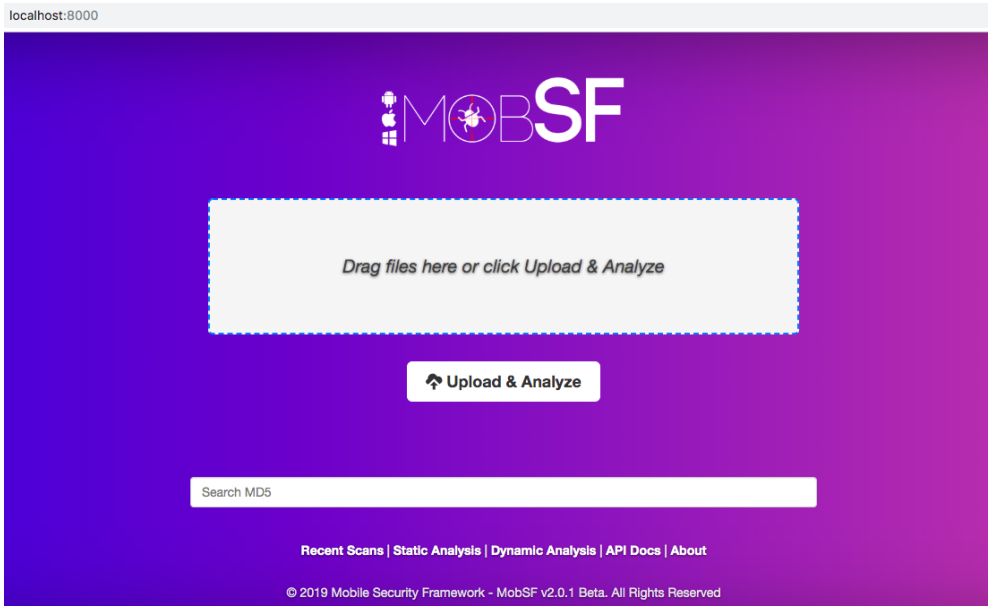

Figura 2.5: Vista de la aplicación MobSF desde en navegador, desde aquí se puede seleccionar la muestra de la aplicación a analizar [31].

Genymotion

Genymotion [20] es un sistema de virtualización para simular la ejecución de las aplicaciones Android. Este software proporciona una plataforma de virtualización de dispositivos móviles que permite realizar análisis de las aplicaciones y es comúnmente utilizado por desarrolladores. También permite analizar por consola en tiempo real qué acciones está ejecutando la aplicación mediante el uso de ADB (Android Debug Bridge). ADB es una herramienta que se comunica con el dispositivo móvil a través de la consola y permite realizar varias acciones en el dispositivo virtual, como instalar aplicaciones, iniciar acciones, listar permisos, etc [9]. En particular, para esta tesis, se utiliza la función *log*, que permite observar las acciones que se están ejecutando en el dispositivo móvil en tiempo real.

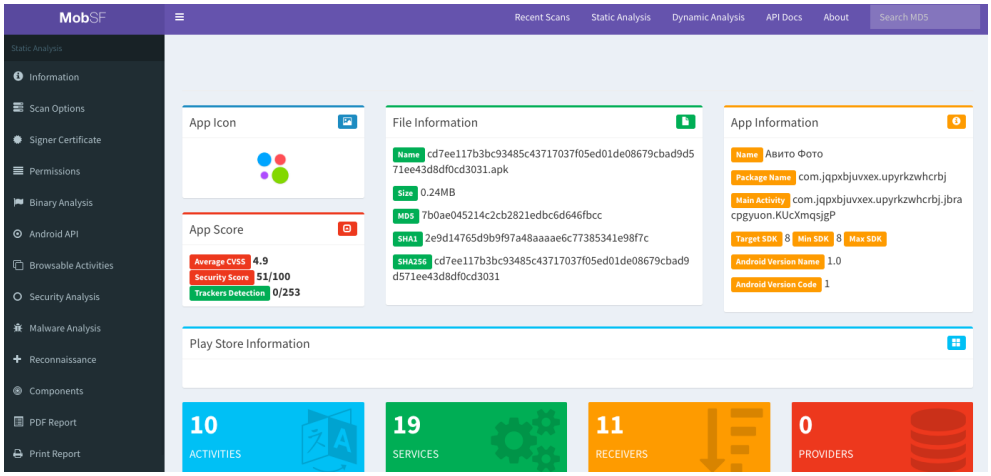

Figura 2.6: Información de una de las muestras APK de la botnet Geost en MobSF.

Jadx

Jadx es una aplicación de código abierto para el análisis de APKs[23]. Esta herramienta produce código fuente Java desde archivos de tipo APK. En la Figura 2.7 se puede observar el funcionamiento de esta herramienta, una vez que se selecciona el archivo con extensión *.apk*, esta herramienta genera el código en Java. Tiene además una interfaz gráfica que facilita el análisis del código. Es posible visualizar en una vista de árbol, realizar búsquedas en el código y ir hacia partes del código desde el AndroidManifest.xml [24].

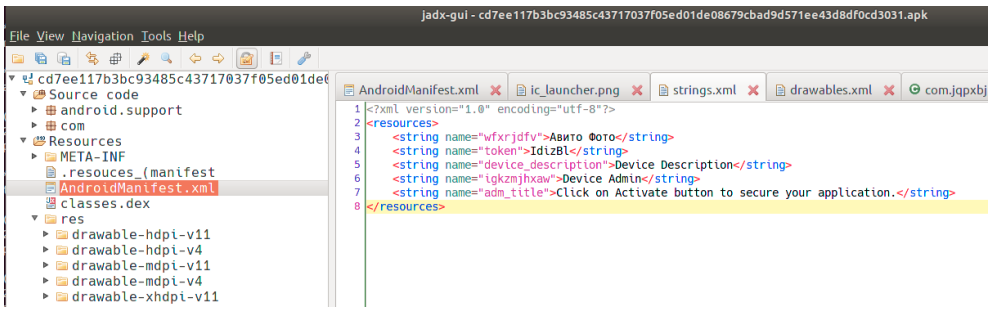

Figura 2.7: Ejemplo del código del APK de una de las muestras de Geost

Capítulo 3

Botnet Geost

Con el fin de comparar aplicaciones HPC estándar con botnets se selecciono una botnet en particular que fue estudiada a fondo y de la cual fui co-autora de su descubrimiento. Esta sección describe la botnet, llamada Geost y sus características HPC.

3.1. Descripción de la Botnet Geost

Geost es una botnet de tipo *Troyano Bancario* que afecta a dispositivos con sistemas operativos Android [40, 41]. El término *Troyano* hace referencia al caballo de Troya ya que este tipo de infecciones se trata de aplicaciones que son instaladas en los equipos de las víctimas quienes creen que están instalando un juego, una actualización o cualquier aplicación normal pero en realidad están siendo infectadas. Las víctimas de Geost instalan una aplicación y sus mensajes e información sobre cuentas bancarias son interceptadas por los atacantes.

Geost fue descubierta en el laboratorio de ciberseguridad de la Universidad Técnica Checa, Stratosphere [46], cuando otra botnet estaba siendo analizada. En el 2018 se ejecutó una muestra de la botnet Htbot. Htbot es una botnet que ofrece un servicio de proxy. Esta botnet convierte a los dispositivos infectados en servidores proxy. Cuando se analizó el tráfico de red de la ejecución de Htbot, se observaron varios indicios en el tráfico de red que nos llevaron al descubrimiento de una nueva botnet. Estos indicios fue el uso del proxy por parte de los botmasters para acceder a un servidor C&C y la transferencia de datos desde el servidor Command and control hacia el

dispositivo del botmaster. La transferencia de datos que se observó, se trataba de la información de los mensajes de textos que los atacantes estaban almacenando y que fueron descargados mientras el tráfico estaba siendo interceptado. Es decir, que se pudo observar el panel de Command and control al cual accedieron los atacantes, los mensajes de textos que estaban almacenando y que fueron consultados mientras se capturaba y almacenaba el tráfico de red. Cuando el tráfico de red se analizó en profundidad, se descubrió que se trataba de una nueva botnet *Banking Trojan* a la que llamamos Geost.

La infraestructura de Geost consiste en servidores *C&C*, donde se encuentra toda la información de los dispositivos Android infectados. La infraestructura evoluciona, ya que los dominios y las direcciones IPs de los servidores *C&C* cambian constantemente para no ser detectados. Como se puede observar en la Figura 3.1, el botmaster usa el proxy htbot, desde donde se capturó el tráfico de red, para acceder al servidor *C&C*. El dispositivo infectado con HtBot, ubicado en nuestro laboratorio cumplía la función de proxy, por lo que fue posible analizar el trafico de red cuando el atacante estaba accediendo al servidor CC&C.

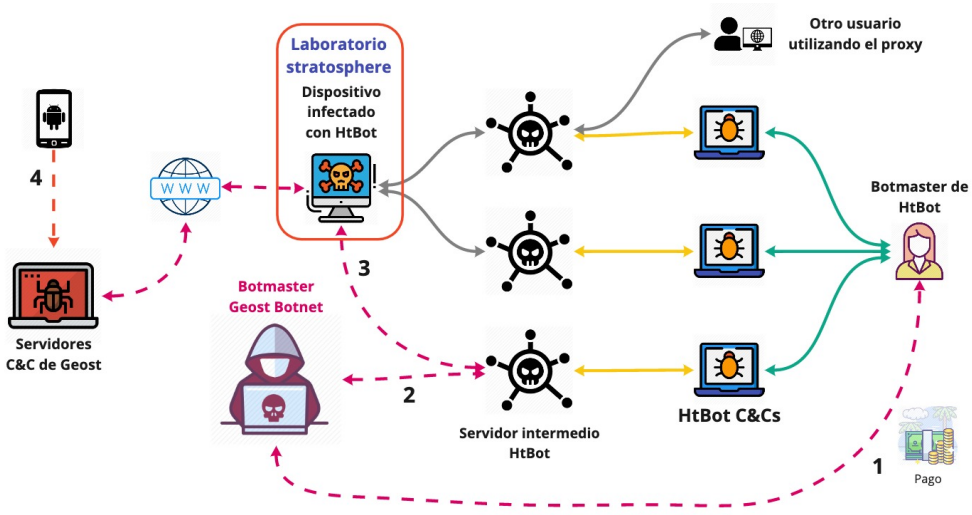

Figura 3.1: Funcionamiento de la botnet htbot. El análisis del funcionamiento de htbot llevó al descubrimiento de la nueva botnet Geost.

Profundizando la investigación y mediante el uso de las herramientas que se mencionaron en la sub-sección 2.4 del Marco Teórico, se descubrieron do-

minios, direcciones IPs y los APKs relacionados con la botnet Geost. Para detectar mas dominios y direcciones IP relacionadas con esta botnet, se analizó el dominio y la dirección IP del servidor *Command and Control*. Se observó que el dominio estuvo registrado previamente bajo otras direcciones IPs. Cuando se examinaron esas direcciones IPs, se detectaron varios dominios con patrones similares en el nombre al dominio de Geost. Por ejemplo, algunos dominios relacionados con Geost son: `w23t2t2tfwg.ru`, `wg34gh34t.xyz`, `32r3t23wef.ru`, `ijsdggrur.ru`, `wgg4ggefwg.ru`. Además, VirusTotal muestra los archivos descargados desde esos dominios e IPs, así se encontraron los APKs relacionados con Geost. Mediante esta metodología de búsqueda se encontraron más de 100 dominios, 17 direcciones IPs y cientos de APKs relacionados la botnet Geost.

Desde el panel *C&C*, los botmasters pueden acceder a los datos de la botnet que se actualizan con información de los teléfonos infectados. Recuperando los paquetes de la captura de red, fue posible también observar el panel *C&C*. El panel del servidor *C&C* se puede observar en la Figura 3.2 los datos que los botmasters tienen de sus bots. Hay 10 columnas para la información de los bots: Estado, ID, IMEI, permisos, versión, operador, país, balance, categoría y flujo. La analogía a este panel de control en los sistemas tradicionales sería el frontend de la aplicación, en este caso, la botnet.

3.2. Experimento de Ejecución de Geost

Para obtener las características de altas prestaciones de la botnet Geost se utilizaron tres métodos, *(i)* a través de la ejecución de la aplicación de Geost (en emulador y en un teléfono real), *(ii)* mediante el análisis del código mediante ingeniería inversa (análisis del código del APK) y *(iii)* por medio del proceso de threat intelligence previamente mencionado (búsqueda de dominios, direcciones IPs y APKs). Para la ejecución del APK de Geost se utilizó Genymotion y un teléfono celular Samsung Galaxy S7. La ejecución de la aplicación Genymotion permitió observar el funcionamiento de la aplicación. La herramienta adb [9] permite comunicar con el dispositivo emulado mediante línea de comando. Abd permitió observar el comportamiento de la aplicación cuando fue ejecutada (los dominios a los que contactaba, las funciones y métodos utilizados).

Una vez que se ha instala la aplicación de Geost en el teléfono Samsung Galaxy S7, se pudo observar que la misma solicita permiso al usuario para

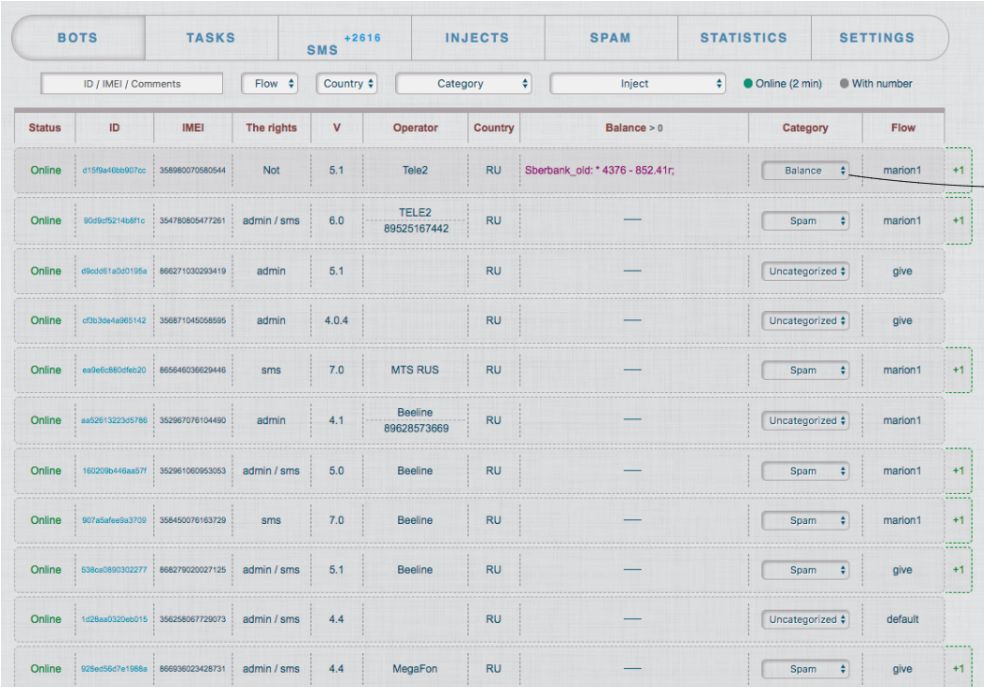

Figura 3.2: Panel de C&C de la botnet Geost, reconstruido a partir del tráfico de red capturado durante el estudio de la botnet htbot.

ser la aplicación de mensajería por defecto. Cuando el usuario acepta, el malware intercepta los mensajes de texto del dispositivo de la víctima. Esto se pudo confirmar en la captura de tráfico de red. Esto se puede observar en la Figura 3.3, donde se ilustra el paquete que fue captura cuando se envío un mensaje desde otro celular con la frase *Ui, Obed?*. En este paquete se puede detectar que se captura el mensaje y número del remitente.

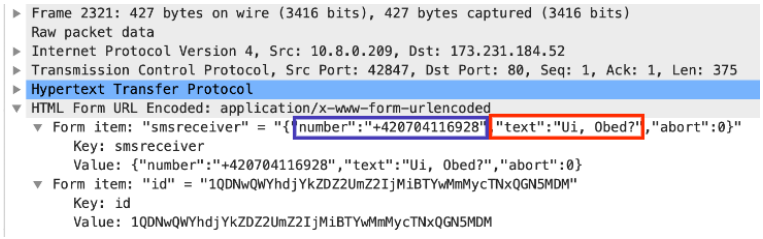

Figura 3.3: Captura de tráfico de red cuando la aplicación Geost fue ejecutada en un teléfono celular donde se puede observar el número de teléfono y el mensaje enviado desde otro teléfono. Esta información fue interceptada por la aplicación de Geost

Para el análisis del código se utilizó la aplicación jadx, detallada en el capítulo Marco teórico, subsección 2.4. Esta herramienta permitió analizar las funciones presentes en el código que presentan características de altas prestaciones y serán detalladas en este capítulo. Se analizaron varias de las muestras de Geost, pero en esta tesis se mostrará el análisis de una de ellas a modo de ejemplo.

3.3. Características de altas prestaciones en Geost

3.3.1. Resiliencia

Como se menciona en la sección 2.1.2, un sistema resiliente es un sistema que toma medidas para maximizar su disponibilidad. Los sistemas HPC estándar buscan ser resilientes de diferentes formas, ya sea cuidando al hardware para evitar que se deteriore (por ejemplo manteniendo condiciones óptimas de temperatura y humedad), teniendo planes de back-up y de redundancia, teniendo personal profesional que mantiene el software, etc. En el caso

de Geost se analizan algunos aspectos que tienen que ver con la resiliencia. Uno de ellos es el cambio constante de direcciones IP y de dominio de los servidores Command and Control. Otra de los aspectos a tener en cuenta es la arquitectura y mantenimiento descentralizado. Una de las peculiaridades que se describen en esta sub-sección tienen que ver con la aplicación en sí, ya que cuando el APK de Geost fue ejecutado en un emulador y en un teléfono celular, se observaron características relacionadas con la resiliencia.

Acceso al servidor Command and control Geost se conecta al servidor de Comando y Control para recibir órdenes o para reportar información sobre el dispositivo. En el código de la aplicación Android del malware se encuentra el dominio al que se conecta el teléfono una vez que la aplicación se ha instalado. Además, hay una lista de más de 100 dominios relacionados con esta botnet. Estos dominios pueden estar legalmente activos, de acuerdo a su registro, entre horas y meses. Cada dominio puede estar asociado a una o más direcciones IP. En la Tabla 3.1 se listan los dominios, las IPs y la cantidad de tiempo total (en días) que estos dominios estuvieron activos.

Mantenimiento descentralizado Durante el proceso de investigación del funcionamiento de la botnet Geost, se encontró un chat log con conversaciones de las personas relacionadas con la operación de Geost. Este chat comprende más de 6200 líneas de chat de 8 meses de conversaciones privadas de 29 personas. Sin embargo no todas las personas estaban relacionadas con el negocio de la botnet Geost [41]. El idioma del chat es ruso y el análisis llevó varios meses. Se detectaron 7 personas involucradas directamente con la operación de Geost en ese chat. Mediante el análisis detallado de las conversaciones entre los administradores de Geost, se pudo observar que el mantenimiento de la botnet Geost es descentralizado. Las personas involucradas en la administración de Geost cumplen diferentes roles, algunas personas están encargadas del desarrollo de los APKs, otras personas están encargadas de la distribución de los APKs, otros administran los servidores. Si alguna de estas personas decide abandonar el proyecto o tiene problemas legales, la botnet sigue funcionando, porque hay otras personas que siguen trabajando.

APKs Otra de las características de los APK de Geost, es que una vez que la aplicación se instala en el dispositivo móvil, no es posible desinstalarla. Si bien la aplicación de Geost no aparece en el menú principal del teléfono, se

puede observar que está instalada accediendo a *Configuración -¿Aplicaciones*. Esto se puede observar en la Figura 3.4.

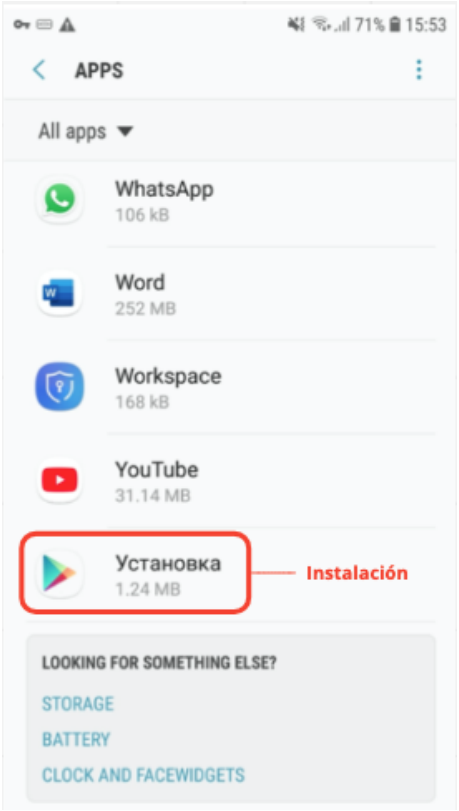

Figura 3.4: APK llamado *Instalación* de la aplicación Geost instalada en un teléfono celular.

Si bien los servidores de C&C aparecen y desaparecen de la red, la botnet sigue activa. Es decir, cambian los dominios y direcciones IP para que los bots establezcan la comunicación con el servidor principal, pero también se modifican los nuevos APKs para que se conecten a los nuevos dominios. Esto implica que hay un mantenimiento y un trabajo de *administrador de sistemas* activo para mantener la botnet activa.

3.3.2. Accesibilidad y disponibilidad

El acceso al servidor Command and control de Geost, no provee accesibilidad las 24 horas, los 7 días de la semana ya que, como se describió en la

sub-sección 3.3.1, los dominios y las direcciones IP utilizados para contactar al servidor C&C cambian de forma constante. La comunicación con el botmaster a través del C&C está disponible por un período corto de tiempo, pero luego desaparece y cambia de dominio y dirección IP. Este tiempo varía entre días, semanas y meses. Estos cambios de direcciones IPs y dominios se pueden observar en la Figura 3.5. En este gráfico se muestran los dominios relacionados con Geost y el tiempo en que estuvieron activos, también se puede observar que hay un cambio en las IPs de los dominios, este cambio se ve reflejado en los valores que toman las diferentes barras.

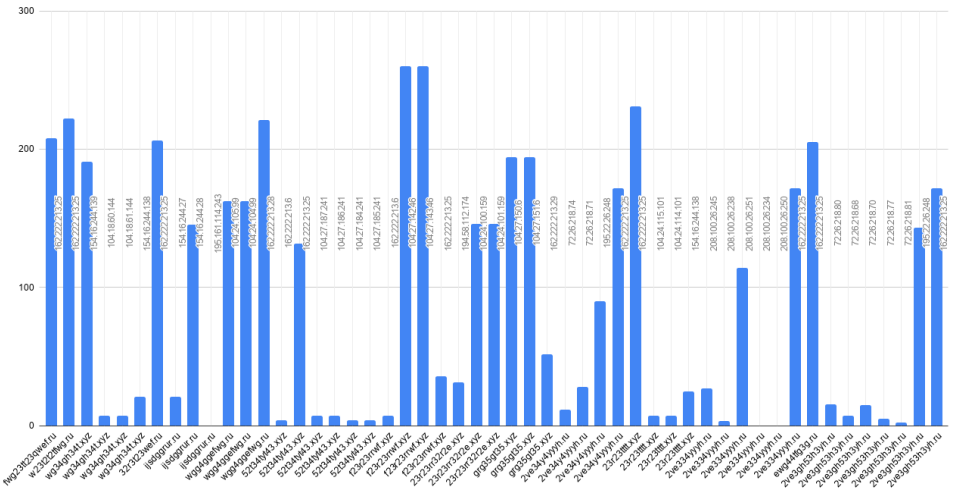

Figura 3.5: Tiempo total (en días) de actividad de los dominios utilizados por Geost

No se pudo realizar un estudio con respecto a la disponibilidad de los servidores C&C actualmente, porque hasta la fecha no se ha encontrado ninguno activo, esto dificulta en análisis de esta característica en particular. Sin embargo, fue posible recompilar los datos y estimar la actividad de los servidores en el pasado. Por lo general, la accesibilidad suele ser limitada en este tipo de aplicaciones maliciosas, ya que el acceso es limitado a un periodo corto de tiempo impide que la botnet sea detectada [19].

3.3.3. Paralelismo y concurrencia

Mediante el análisis del código de la aplicación maliciosa, se pudo observar que la misma soporta dos o más acciones al mismo tiempo. Esto es posible porque utiliza un método de java llamado *syncronized*. Los métodos en Java son bloques de código que son ejecutados cuando se los invoca [25], en otras palabras, los métodos son funciones en Java. El método *syncronized* se utiliza para sincronizar hilos que intenten acceder al mismo objeto.

Muchas veces, en los programas multi-hilos se presenta la situación en la que varios hilos intentan acceder al mismo recurso, esto produce errores inesperados. A este tipo de situaciones, se las llama *condiciones de carrera*. Para evitar las situaciones de carrera, se utilizan los mecanismos de sincronización.

Hay dos mecanismos de sincronización [8]:

- **Sincronización por exclusión mutua**: Este método se utiliza cuando varios procesos intentan acceder al mismo recurso. Este mecanismo excluye a los demás procesos para que sólo uno pueda acceder al recurso y evitar errores por solapamiento.
- **Sincronización por condición**: Este método se aplica en las situaciones en las que un proceso se demora esperando que un evento lo active. Cuando esto sucede, otro proceso lo despierta que detectó que ese evento ocurrió.

Java tiene varias formas de sincronizar, en el caso de la muestra de aplicación de Geost que se analizó, se observó que la aplicación utiliza la sincronización por exclusión mutua (mediante el uso de bloques). En la Figura 3.6 se puede observar que Geost está haciendo uso de este método.

```
private static Method XjsWiBn() {
    if (OiobCnWUWJD == null) {
        synchronized (XjsWiBn) {
            OiobCnWUWJD = ClassLoader.class.getDeclaredMethod("defineClass", new Class[]{String.class, byte[].class, Integer.TYPE,
            try {
                OiobCnWUWJD.setAccessible(true);
            } catch (Exception e) {
            }
        }
    }
    return OiobCnWUWJD;
}
```

Figura 3.6: Utilización del método *synchronize* en el APK de Geost indica que la botnet usa un sistema de sincronización por exclusión mutua.

Teniendo en cuenta que hay varios APKs que son parte de esta botnet y que se ejecutan al mismo tiempo, la botnet Geost se la puede considerar como

un sistema paralelo. Es decir que en un mismo momento, varios teléfonos van a estar ejecutando este APK, aunque la ejecución no haya comenzado necesariamente al mismo tiempo. Además, se analizaron los modelos de los teléfonos en los que esta aplicación fue ejecutada, y que cada dispositivo móvil tiene más de un núcleo en el procesador. Teniendo en cuenta se que el código de la aplicación utiliza mecanismos de sincronismo y el tipo de dispositivo en el que fue ejecutada tiene más de un núcleo. Esto implica que si se generan varios hilos, y cada hilo se ejecuta sobre núcleos diferentes no sólo hay concurrencia sino también paralelismo. Por esta razón, se puede considerar que esta botnet ya es un sistema paralelo en cada teléfono que la ejecuta.

3.3.4. Computación heterogénea

Como se explicó en el Capítulo 2, la computación heterogénea se la puede analizar desde dos perspectivas, una es desde el sistema que contiene varios dispositivos con diferentes características de procesador conectados entre sí. En este caso se trata de sistemas distribuidos o paralelos que están conectados con dispositivos de diferentes características en cuanto al modelo de procesador, cantidad de núcleos, memoria RAM, etc. La otra perspectiva tiene en cuenta la arquitectura interna de cada dispositivo, es decir, la arquitectura de cada chip.

Desde la perspectiva de Geost como un sistema: En este caso se considera que Geost contiene la infraestructura de los botmasters y todos los dispositivos infectados. Los dispositivos infectados, tienen diferentes arquitecturas ya que son diferentes modelos de celulares y tablets. Geost como sistema utiliza diferentes arquitecturas ya que los APK funcionan en diferentes dispositivos. La capacidad de cómputo de Geost, que contiene casi un millón de teléfonos infectados, tiene una capacidad de cómputo que no está siendo utilizada en su totalidad, pero podría ser utilizada.

Desde la perspectiva de cada dispositivo individual: Los dispositivos que fueron infectados con este malware tienen arquitecturas diferentes. Son diferentes marcas de teléfonos y tablets con diferentes arquitecturas de procesador. Cuando se analizó el código del malware, no se detectó que en el código se haga uso de la arquitectura de forma heterogénea. En el área

de la computación de altas prestaciones se relaciona el *balance de carga* con la computación heterogénea ya que para resolver problemas sobre más de un equipo con arquitecturas diferentes, se le debería dar tareas o cargas de trabajo acordes a la potencia computacional de cada uno de los equipos o elementos en el equipo (núcleos de procesador o GPU). En el caso de Geost, habría que realizar un análisis detallado sobre la carga computacional que se le asigna a cada núcleo de cada dispositivo y la carga que se le asigna a cada móvil. Sin embargo este estudio no se tiene en cuenta en el desarrollo de esta tesis.

3.3.5. Ancho de Banda

Para medir el ancho de banda utilizado por Geost, se ejecutó la aplicación maliciosa en un teléfono Samsung Galaxy S7, se monitorizó el tráfico de red para observar el comportamiento y medir el ancho de banda. El objetivo es cuanto ancho de banda podría utilizar toda la botnet Geost en conjunto. En la captura de red se busco la dirección IP del dominio malicioso y se obtuvo la cantidad de bytes y bits enviados por segundo. El resultado fue de 4 bytes/s. La comunicación sólo constaba de el contacto inicial con el servidor Command and Control. Sin embargo, el servidor con el que se estaba comunicando no era el servidor malicioso, si no que se trataba de un *sinkhole*. Mediante el análisis del tráfico de red con tcpdump, se observó que se resuelve el dominio `f2323tr23t.ru`, la IP de ese dominio es `173.231.184.52`. Luego desde el teléfono celular infectado, se puede observar que envía paquetes en el puerto `80/TCP` hacia el servidor Command and Control, la información del primer paquete se puede observar en la Figura 3.7, donde se destacan:

- **País**: en este caso CZ (por las siglas Czech Republic, República Checa)
- **IMEI**: es el identificar único de cada dispositivo móvil
- **Model**: es el modelo de teléfono celular en el que corre la aplicación Geost
- **Thread**: es una identificación el hilo
- **SDK**: es la versión de sdk que se está utilizando en el teléfono infectado
- **Operator**: es el nombre del operador del teléfono, en este caso, Vodafone

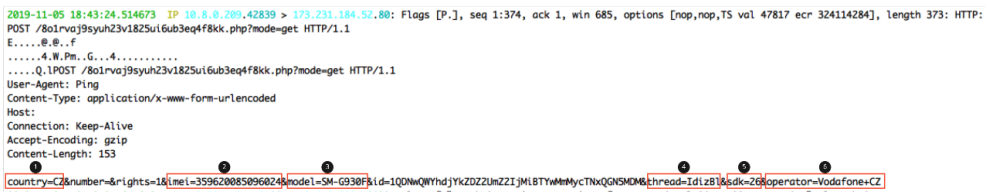

Figura 3.7: Captura de tráfico de red cuando la aplicación Geost fue ejecutada en un teléfono celular, donde se puede observar: País, IMEI, Modelo, Hilo, versión SDK y Operador

La utilización del ancho de banda en la botnet Geost es bastante bajo dado que el objetivo de esa botnet no es consumir el ancho de banda del teléfono. Sin embargo muchos otras botnet abusan del ancho de banda de la víctima y lo venden a otras personas (como el caso ya discutido de la botnet Htbot). Sin embargo, desde el punto de vista de la utilización potencial de la botnet Geost deberíamos considerar el ancho de banda que podría utilizar cada teléfono. En el caso de LTE que es actualmente la tecnología mas distribuida de conexión celular, hay un ancho de banda teórico de 100 Mbit/s en la conexión de bajada y 50 Mbit/s en la conexión de subida [38].

Si se calculo que Geost tenia al menos 800,000 teléfonos infectados, la botnet podría poseer 40,000,000 Mbit/s de subida. Solo se considera el envío de datos ya que es el recurso mas valioso y usado. Esta cantidad es equivalente a 5 Terabytes de información enviada **por segundo** a disponibilidad de la botnet Geost.

Cuadro 3.1: Dominios usados como Command & Control por Geost, el país en que está registrada la dirección IP, la dirección IP, y tiempo total de actividad de esa dirección IP en días.

Dominio	País	IP	Tiempo activo
fwg23tt23qwef.ru	US	162.222.213.25	207,7
w23t2t2tfwg.ru	US	162.222.213.25	222
wg34gh34t.xyz	US	154.16.244.139	191,2
	N/A	104.18.60.144	7,23
	N/A	104.18.61.144	7,23
	US	154.16.244.138	20,91
	US	162.222.213.25	206,34
ijsdggrur.ru	US	154.16.244.27	20,9
	US	154.16.244.28	145,49
	RU	195.161.114.243	0
wgg4ggefwg.ru	N/A	104.24.105.99	162,5
	N/A	104.24.105.99	162,5
	N/A	104.24.105.99	221,35
52t34tyt43.xyz	US	162.222.213.6	3,9
	US	162.222.213.25	131,6
	N/A	104.27.187.241	7,35
	N/A	104.27.186.241	7,35
	N/A	104.27.184.241	3,85
	N/A	104.27.185.241	3,85
f23r23rrwf.xyz	US	162.222.213.6	7,30
	US	104.27.142.46	260,24
	US	104.27.143.46	260,24
	US	162.222.213.25	35,48
23r23rr32r2e.xyz	RU	194.58.112.174	31,54
	US	104.24.100.159	145,73
	US	104.24.101.159	145,73
grg35gt35.xyz	US	104.27.150.6	194,16
	US	104.27.151.6	194,16
	US	162.222.213.29	51,31
2ve34y4yyyh.ru	NL	72.26.218.74	11,27
	NL	72.26.218.71	28,08
	PT	195.22.26.248	89,91
	US	162.222.213.25	171,92
23r23tttt.xyz	US	162.222.213.25	231,18
	US	104.24.115.101	7,16
	US	104.24.114.101	7,16
	US	154.16.244.138	24,57
2ve334yyyh.ru	US	208.100.26.245	27,02
	US	208.100.26.238	3,07
	US	208.100.26.251	113,84
	US	208.100.26.234	0,09
	US	208.100.26.250	0,044
	US	162.222.213.25	171,79
ewg44tfg3g.ru	US	162.222.213.25	205,30
2ve3gh53h3yh.ru	NL	72.26.218.80	15,105
	NL	72.26.218.68	6,98
	NL	72.26.218.70	14,99
	NL	72.26.218.77	4,96
	NL	72.26.218.81	2,38
	PT	195.22.26.248	143,22
	US	162.222.213.25	171,91

Capítulo 4

Aplicación SETI@home

4.1. Aplicación estándar: BOINC

La computación voluntaria hace referencia a un tipo de computación distribuida, que tiene como objetivo resolver problemas computacionales intensivos con costos mínimos. En este sistema, los dispositivos pertenecen a los usuarios que colaboran con el proyecto. La capacidad computacional de los dispositivos se utiliza cuando los usuarios no están haciendo uso del dispositivo [49]. En este sentido, los participantes que deseen colaborar con un proyecto científico instalan una aplicación en su dispositivo (computadora portátil, teléfono, tablet) que descarga y ejecuta trabajos desde los servidores operados por los proyectos científicos. Alrededor de 700,000 dispositivos participan activamente en proyectos de Computación Voluntaria[5]. Estos dispositivos tienen alrededor de 4 millones de núcleos de CPU y 560,000 GPU, y colectivamente proporcionan un rendimiento promedio de 93 PetaFLOPS [5].

BOINC, por sus siglas en inglés *Berkeley Open Infrastructure for Network Computing* (Infraestructura Abierta de Berkeley para la Computación en Red) es una infraestructura para la computación distribuida desarrollada en la Universidad de Berkeley [10].Esta plataforma le permite a cualquier persona colaborar como clientes en cualquier lugar del mundo con la investigación científica de vanguardia. Esto es posible utilizando una computadora (con cualquier sistema operativo, ya sea Windows, Mac, Linux) o dispositivo móvil Android mientras éstos no estén siendo utilizados. BOINC descarga trabajos de computación científica en el dispositivo de los voluntarios y los

ejecuta en segundo plano.

La arquitectura de BOINC incluye múltiples componentes que interactúan a través de interfaces RPC (Remote Procedure Call), que es un mecanismo utilizado en computación distribuida que permite la ejecución de subrutinas en otro espacio de direcciones. BOINC utiliza RPC sobre TCP y en la capa de aplicación utiliza el protocolo HTTP. Este diseño le permite ser modular y extensible [5]. Cada proyecto que utilice BOINC, tiene su servidor que distribuye los trabajos a los dispositivos de los voluntarios. Cada proyecto tienen un sitio web al que se puede acceder públicamente y un administrador de las cuentas de los usuarios. Además, BOINC cuenta con módulos que envían los trabajos y estadísticas de créditos al servidor de cada proyecto. Este funcionamiento se detalla en la Figura 4.1.

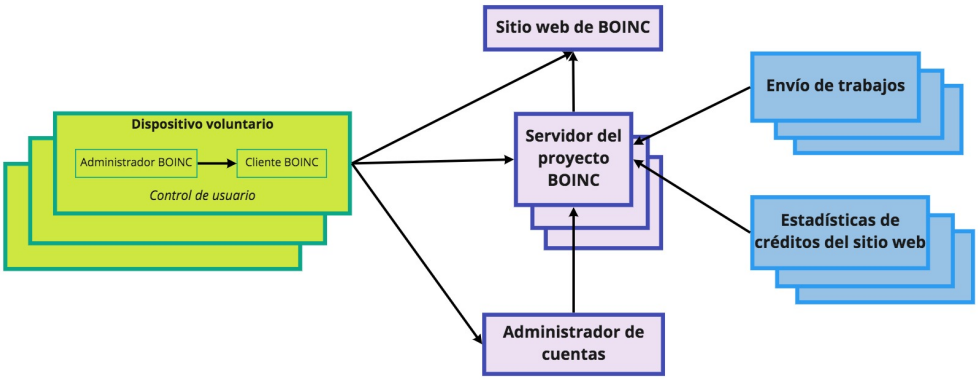

Figura 4.1: Arquitectura de BOINC. Los elementos son Dispositivo del cliente voluntario (que incluye el administrador BOINC y el cliente), conectado al sitio web de BOINC, al servidor del proyecto BOINC y al administrador de cuentas. A su vez, hay trabajos y estadísticas de los créditos de cada usuario que se envían al servidor BOINC.

Los dispositivos voluntarios cuentan con un Administrador BOINC que se comunica con el cliente BOINC utilizando un sistema de control de usuario. Los voluntarios que deseen colaborar tienen que seguir los siguientes pasos:

- **Instalación**: Se instala el cliente BOINC en el dispositivo del voluntario.
- **Selección del proyecto**: Se selecciona el o los proyectos con los que quieran colaborar, creando un cuenta por proyecto.

- **Conexión**: Se conecta cada cuenta al/los proyecto(s) seleccionado(s) en el paso anterior. Cada conexión es asignada a un recurso compartido indicando la carga computacional a ejecutar para el proyecto.

BOINC tiene una interacción cliente servidor. En este sentido, cuando un cliente esta conectado a un proyecto envía mensajes al servidor para reportar los trabajos finalizados y obtener nuevos trabajos. El cliente descarga la aplicación, los archivos de entrada, los trabajos a ejecutar y sube los archivos de salidas, es decir, los datos procesados. Todas las comunicaciones utilizan protocolo HTTP [5].

Los proyectos BOINC operan en un servidor que consiste en uno o más sistemas de computadoras que ejecutan el software del servidor de BOINC. Estos sistemas suelen correr en hardware, máquinas virtuales o en nodos en cloud. Estos servidores tienen varios componentes: planificadores de los trabajos, memoria compartida, cache para los trabajos, lista de trabajos (que son administrados por los planificadores) y bases de datos relacionales. Las bases de datos relacionales se comunican con varios demonios (*transitionar, assimilators, DB purger, validators, file deleter y job creators*). La conexión entre éstos elementos se puede observar en la Figura 4.2.

4.1.1. SETI@home

SETI (Search for Extraterrestrial Intelligence), cuyas siglas significan *Búsqueda de inteligencia Extraterrestre*, es un proyecto cuyo objetivo es detectar inteligencia fuera del planeta tierra. Uno de los enfoques para atacar este problema es mediante el uso de radio telescopios. Estos telescopios escuchan señales espaciales de ancho de banda limitado (narrow-bandwidth). Este tipo de señales no suelen ocurrir de forma natural, por lo que su detección ayudaría a proveer evidencia de la existencia de tecnología extraterrestre *SETI@home*[1].

SETI@home ha demostrado la viabilidad de la computación distribuida basada en voluntarios para resolver problemas científicos. Del proyecto de SETI@home, surge BOINC (Berkeley Open Infrastructure for Network Computing), que permite usar el modelo de computación distribuida voluntaria en otras aplicaciones [28].

El modelo computacional utilizado en SETI@home es simple. La señal de datos recibida se divide en unidades de trabajo que son distribuidas a través de internet hacia el programa del cliente (estos programas corren en

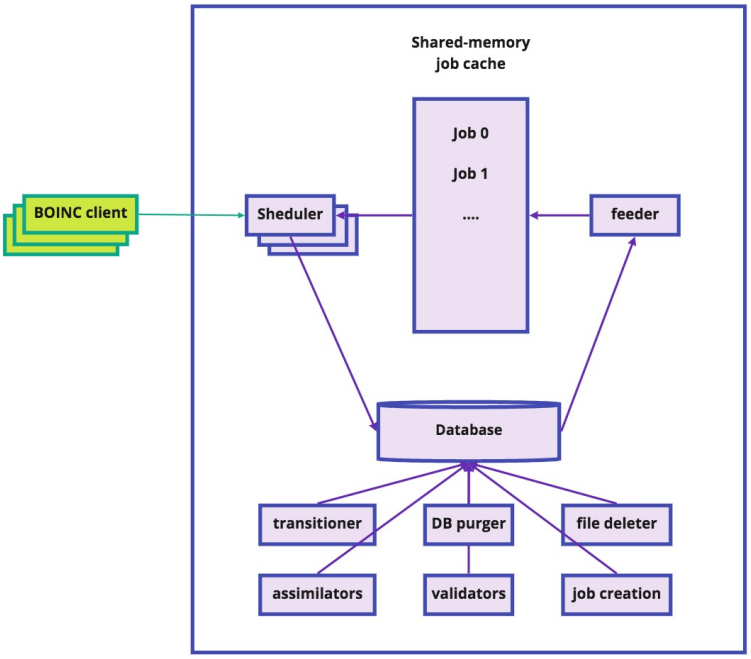

Figura 4.2: Arquitectura de los servidores BOINC. El cliente se conecta al servidor BOINC que tiene un administrador de procesos (scheduler) que recibe los trabajos. La bases de datos se comunica con los demonios (trasitioner, assimilatos, DB purger, validators, file deleter y job creator). Ésta base de datos alimenta al componente llamado "feeder", que luego lista los trabajos (jobs) que son enviados a los clientes

los dispositivos de miles de clientes voluntarios). Cuando el cliente computa el resultado, lo devuelve al servidor y recibe otra unidad de trabajo, por lo que no hay comunicación entre los clientes, es decir que el sistema es centralizado. El uso de los recursos computacionales de los clientes, se realiza mientas éstos están inactivos, es decir mientras el usuario no está utilizando su dispositivo [6]. Esto se comprobó cuando se instalo la aplicación BOINC en un teléfono con sistema operativo Android. Por defecto, la aplicación de BOINC solo se ejecuta cuando el teléfono celular esta cargando (por cable, por USB o de forma inalámbrica), esto se puede observar en la Figura 4.3 se puede observar esta configuración, además, la opción de utilizar la aplicación solo con la batería, se encuentra marcada en color rojo. Además, por defecto BOINC se activa cuando el teléfono celular tiene una carga de mas del 90 % de batería. El usuario de la aplicación puede acceder a esta configuración y modificar los valores por defecto.

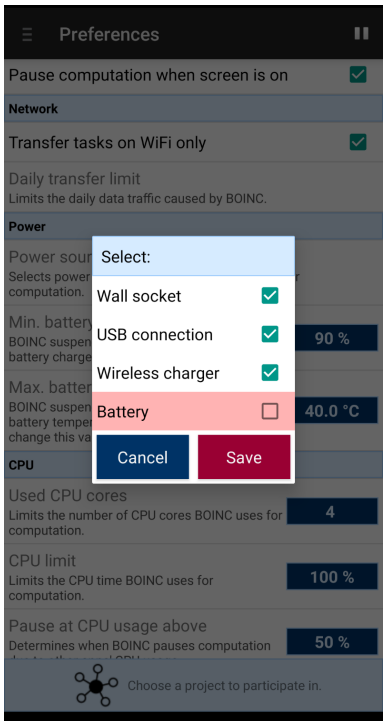

Figura 4.3: Configuración por defecto de la aplicación BOINC instalada en un teléfono celular Android

El servidor que procesa los datos y los resultados utiliza un sistema multi-

hilo y un protocolo basado en HTTPS para que los clientes detrás de firewalls puedan conectarse sin dificultad.

4.2. Resiliencia

Para los desarrolladores de SETI@home, mantener el sistema del servidor en funcionamiento ha sido la parte más difícil y costosa del proyecto [6]. Las posibilidades de fallas del sistema, tanto de hardware como de software, parecían ilimitadas. Para resolverlo, se diseñó una arquitectura que minimiza las dependencias entre los subsistemas de servidores. Por ejemplo, el servidor de datos / resultados puede funcionar en un modo en el que, en lugar de utilizar la base de datos para enumerar las unidades de trabajo, el servidor obtiene la información de un archivo de disco, lo que permite distribuir datos incluso cuando la base de datos está inactiva.

En la pagina del proyecto de SETI@home [42], se puede observar el estado del servidor, los programas que están corriendo en tiempo real, por ejemplo hay una base de datos maestra y una réplica de esa base, también se encuentra la base de datos con los resultados, entre otros archivos y datos. Para toda esta información hay tres estados: corriendo (es decir que el programa se encuentra corriendo con normalidad), no corriendo (el programa falló) o desactivado (el programa fue desactivado por mantenimiento). Estas métricas se muestran en la Figura 4.4.

Average floating point operations per second	1,056,451.9 GigaFLOPS / 1,056.452 TeraFLOPS
Contribution to BOINC combined total credit	1.54181%
Contribution to BOINC combined total RAC	4.55347%
Users	1,793,135
Active users	89,310 (4.98%)
Hosts	166,406
Active hosts	146,644 (88.12%)

Figura 4.4: Estado del proyecto de SETI@home que se muestra en la pagina oficial del proyecto [42].

BOINC utiliza un mecanismo para verificar que los clientes estén conectados. Este mecanismo, llamado *heartbeat* (traducido al español es latido de corazón) se trata de un mensaje que se envía cada un segundo desde el cliente a la aplicación, si la aplicación no recibe un mensaje de heartbeat por 30 segundos, deja de ejecutarse [4].

Cuadro 4.1: Dominios usados como Command&Control por SETI@Home, y el país en que está registrada la dirección IP, la dirección IP, y tiempo total de actividad (en días).

Dominio	**País**	**IP**	**Tiempo activo**
boinc.berkeley.edu	US	208.68.240.115	1743

Como se explicó en la sección 4.1, el servidor BOINC tiene un a arquitectura multi-demonio (esto se ilustra en la Figura 4.1). Esta arquitectura multi-demonio proporciona una forma de tolerancia a fallos. Si un demonio falla (por ejemplo, un asimilador falla porque una base de datos externa está caída) los otros componentes continúan trabajando, el trabajo para el componente fallido se acumula en la base de datos y eventualmente es procesado una vez que el componente este funcional.

4.3. Accesibilidad y disponibilidad

El proyecto está funcionando desde 1999 y provee accesibilidad desde ese momento. No hay información sobre los métodos que utilizan para la disponibilidad del servicio, pero ya que no necesita una gran cantidad de ancho de banda, si los clientes pierden conectividad por un período de tiempo, van a seguir ejecutándose y cuando vuelvan a tener conectividad transmitirán los resultados al servidor. La dirección IP utilizada por SETI@home es `208.68.240.115`, el subdominio es `boinc.berkeley.edu`, fue registrado el 18 de Agosto de 2015 y continúa activo hasta la fecha, como se muestra en la Tabla 4.1. Este subdominio pertenece al dominio `berkeley.edu` y fue registrado por primera vez el 24 de Abril de 1985.

4.4. Paralelismo y concurrencia

En esta aplicación, los clientes no están necesariamente ejecutando la aplicación al mismo tiempo, ya que depende de la inactividad de los dispositivos. La aplicación hace uso de la concurrencia en su versión para computadoras de escritorio y laptops, ya que permite configurar en una computadora la cantidad de núcleos que se van a disponer para que se ejecute. Mediante el

análisis del APK de esta aplicación, se observó que también hace uso del método *synchronized* como se muestra en la Figura 4.5.

```
public void freeBackStackIndex(int index) {
    synchronized (this) {
        this.mBackStackIndices.set(index, null);
        if (this.mAvailBackStackIndices == null) {
            this.mAvailBackStackIndices = new ArrayList();
        }
        if (DEBUG) {
            Log.v(TAG, "Freeing back stack index " + index);
        }
        this.mAvailBackStackIndices.add(Integer.valueOf(index));
    }
}
```

Figura 4.5: Utilización del método *synchronized* en el APK de SETI@home

4.5. Computación heterogénea

La aplicación SETI@home utiliza millones de computadores y dispositivos móviles en todo el mundo para realizar cálculos. El rendimiento se puede observar en su sitio web [42] donde hay una lista detallada de el modelo de procesador, la cantidad de computadoras, el promedio de núcleos por computadora, los GFlops por núcleo y los GFlops por computadora, esto se detalla en la Figura 4.6. Además, la aplicación hace uso de diferentes GPUs [42] , esta lista está clasificada por marca (NVidia, ATI/AMD e Intel) y por sistema operativo (Windows, Linux y Mac). En la Figura 4.7 se muestra la lista de los GPU Intel, en la Figura 4.8 se listan los GPU NVIDIA y en la Figura 4.9 se muestran la lista de GPU ATI/AMD. Es decir que se puede considerar como un sistema heterogéneo, no sólo porque utiliza los procesadores de diferentes dispositivos, si no por que también es capaz de utilizar la capacidad de cómputo de las GPU.

4.6. Ancho de Banda

La aplicación resuelve el dominio `boinc.berkeley.edu` que devuelve la IP `208.68.240.115`. El servicio funciona en el puerto `443/TCP`. Mediante el análisis de la captura de tráfico de red para un día completo del servicio corriendo, se obtuvo que la tasa de bytes de datos, *Data byte rate*, es de 287

CPU performance

This table shows peak CPU speed (based on Whetstone benchmarks) of computers participating in this project.

CPU model	Number of computers	Avg. cores/computer	GFLOPS/core	GFLOPs/computer
Intel(R) Core(TM) i5-5675R CPU @ 3.10GHz [x86 Family 6 Model 71 Stepping 1]	53	4.00	6.45	25.82
Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz [x86 Family 6 Model 158 Stepping 13]	16	6.00	6.34	38.05
Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz [x86 Family 6 Model 158 Stepping 12]	28	6.00	6.29	37.73
Intel(R) Core(TM) i7-5775R CPU @ 3.30GHz [x86 Family 6 Model 71 Stepping 1]	18	8.00	6.29	50.29
Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz [x86 Family 6 Model 158 Stepping 12]	46	16.00	6.26	100.09
Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz [x86 Family 6 Model 158 Stepping 13]	34	16.00	6.22	99.47
Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz [x86 Family 6 Model 158 Stepping 9]	167	8.00	6.07	48.56
Intel(R) Core(TM) i5-8600 CPU @ 3.10GHz [x86 Family 6 Model 158 Stepping 10]	15	6.00	6.04	36.23
Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz [x86 Family 6 Model 158 Stepping 10]	10	12.00	5.92	71.07
Intel(R) Core(TM) i5-5575R CPU @ 2.80GHz [x86 Family 6 Model 71 Stepping 1]	50	4.00	5.92	23.67
Intel(R) Core(TM) i5-7600K CPU @ 3.80GHz [x86 Family 6 Model 158 Stepping 9]	102	4.00	5.81	23.24
Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz [x86 Family 6 Model 158 Stepping 10]	86	6.00	5.75	34.49
Intel(R) Core(TM) i5-7600 CPU @ 3.50GHz [x86 Family 6 Model 158 Stepping 9]	57	4.00	5.70	22.80
AMD Ryzen Threadripper 3960X 24-Core Processor [Family 23 Model 49 Stepping 0]	15	46.40	5.67	263.22
AMD Ryzen Threadripper 3970X 32-Core Processor [Family 23 Model 49 Stepping 0]	17	60.24	5.66	340.65

Figura 4.6: Lista de modelos de procesadores de SETI@home.

Intel

Total

1. (1.000) Iris Pro
2. (0.754) Intel(R) HD Graphics 530
3. (0.746) Intel(R) UHD Graphics 630
4. (0.731) Intel(R) HD Graphics 630
5. (0.629) Iris
6. (0.619) Intel(R) HD Graphics 4600
7. (0.532) Intel(R) HD Graphics 520
8. (0.512) HD Graphics 4000
9. (0.492) Intel(R) UHD Graphics 620
10. (0.466) Intel(R) HD Graphics 620

Windows

1. (1.000) Intel(R) HD Graphics 530
2. (0.953) Intel(R) HD Graphics 630
3. (0.855) Intel(R) UHD Graphics 630
4. (0.844) Intel(R) HD Graphics 4600
5. (0.725) Intel(R) HD Graphics 520
6. (0.672) Intel(R) UHD Graphics 620
7. (0.636) Intel(R) HD Graphics 620
8. (0.619) Intel(R) HD Graphics 4400
9. (0.585) Intel(R) HD Graphics 4000
10. (0.562) Intel(R) HD Graphics 5500

Linux

1. (1.000) Intel(R) HD Graphics 6000 BroadWell U-Processor GT3
2. (0.031) Intel(R) Gen9 HD Graphics NEO
3. (0.013) Intel(R) HD Graphics IvyBridge GT2
4. (0.007) Intel(R) HD Graphics Cherryview

Mac

1. (1.000) Intel(R) Iris(TM) Pro Graphics 6200
2. (0.802) Intel(R) Iris(TM) Graphics 550
3. (0.659) Iris Pro
4. (0.652) Intel(R) Iris(TM) Plus Graphics 640
5. (0.569) Intel(R) HD Graphics 530
6. (0.566) Intel(R) UHD Graphics 630
7. (0.415) Iris
8. (0.363) HD Graphics 5000
9. (0.338) HD Graphics 4000
10. (0.278) Intel(R) Iris(TM) Graphics 6100

Figura 4.7: Utilización de GPUs Intel en SETI@home.

Top GPU models

The following lists show the most productive GPU models on different platforms. Relative speeds, measured by average elapsed time of tasks, are shown in parentheses.

NVIDIA

Total

1. (1.000) GeForce RTX 2080 Ti
2. (0.983) GeForce GTX 1080 Ti
3. (0.840) GeForce RTX 2080
4. (0.750) GeForce RTX 2060 SUPER
5. (0.677) GeForce RTX 2060
6. (0.633) GeForce GTX 980
7. (0.592) GeForce GTX 1070
8. (0.581) GeForce RTX 2070 SUPER
9. (0.538) GeForce GTX 1070 Ti
10. (0.497) GeForce GTX 780
11. (0.492) GeForce GTX 1660

Windows

1. (1.000) GeForce RTX 2080 Ti
2. (0.983) GeForce GTX 1080 Ti
3. (0.840) GeForce RTX 2080
4. (0.750) GeForce RTX 2060 SUPER
5. (0.677) GeForce RTX 2060
6. (0.586) GeForce GTX 1070
7. (0.575) GeForce GTX 980
8. (0.531) GeForce RTX 2070 SUPER
9. (0.508) GeForce GTX 1070 Ti
10. (0.494) GeForce GTX 1060 3GB
11. (0.492) GeForce GTX 1660

Linux

1. (1.000) GeForce GTX 1050 Ti
2. (0.266) GeForce GTX 1660 Ti

Mac

1. (1.000) GeForce GTX 680
2. (0.844) GeForce GTX 770
3. (0.750) GeForce GTX 780M
4. (0.643) GeForce GTX 680MX
5. (0.622) GeForce GTX 775M
6. (0.622) GeForce GTX 675MX
7. (0.441) GeForce GTX 660M
8. (0.268) GeForce GT 750M
9. (0.192) GeForce GT 640M
10. (0.076) GeForce GT 650M
11. (0.023) GeForce 9400

Figura 4.8: Utilización de GPUs NVIDIA en SETI@home.

ATI/AMD

Total

1. (1.000) AMD Radeon (TM) R9 390 Series
2. (0.844) Radeon (TM) RX 470 Graphics
3. (0.757) AMD Radeon RX 5700 XT
4. (0.745) AMD Radeon Pro 580X Compute Engine
5. (0.721) Radeon (TM) RX 480 Graphics
6. (0.714) Radeon RX Vega
7. (0.681) AMD Radeon (TM) RX 480
8. (0.667) Radeon RX 590 Series
9. (0.633) AMD Radeon Pro 580 Compute Engine
10. (0.627) Radeon RX 570 Series
11. (0.565) Radeon RX 580 Series
12. (0.557) AMD Radeon HD 7870/7950/7970/R9 280/R9 280X series (Tahiti)

Windows

1. (1.000) AMD Radeon (TM) R9 390 Series
2. (0.844) Radeon (TM) RX 470 Graphics
3. (0.757) AMD Radeon RX 5700 XT
4. (0.721) Radeon (TM) RX 480 Graphics
5. (0.714) Radeon RX Vega
6. (0.681) AMD Radeon (TM) RX 480
7. (0.667) Radeon RX 590 Series
8. (0.624) Radeon RX 570 Series
9. (0.579) Radeon RX 580 Series
10. (0.557) AMD Radeon HD 7870/7950/7970/R9 280/R9 280X series (Tahiti)
11. (0.461) AMD Radeon RX 580 2048SP
12. (0.457) AMD Radeon R9 200 Series

Linux

1. (1.000) Vega 20
2. (0.457) Radeon RX 570 Series
3. (0.186) AMD Radeon (TM) RX 480 Graphics
4. (0.174) Radeon RX550/550 Series
5. (0.169) AMD Radeon (TM) RX 460 Graphics
6. (0.123) AMD Radeon HD 7850/7870 series (Pitcairn)
7. (0.103) Radeon RX 580 Series

Mac

1. (1.000) AMD Radeon Pro Vega 56 Compute Engine
2. (0.808) AMD Radeon Pro 580X Compute Engine
3. (0.687) AMD Radeon Pro 580 Compute Engine
4. (0.602) AMD Radeon Pro 570 Compute Engine
5. (0.595) AMD Radeon HD - FirePro D500 Compute Engine
6. (0.562) AMD Radeon Pro 575 Compute Engine
7. (0.543) AMD Radeon Vega Frontier Edition Compute Engine
8. (0.536) AMD Radeon Pro 5500M Compute Engine
9. (0.503) AMD Radeon HD 7950 Compute Engine
10. (0.460) AMD Radeon R9 M395X Compute Engine
11. (0.395) AMD Radeon R9 M390 Compute Engine

Figura 4.9: Utilización de GPUs ATI/AMD en SETI@home.

bytes/segundo, la tasa de bits de datos, *Data bit rate* es de 2301 bits/segundo, y el tamaño promedio del paquete, *Average packet size*, es de 573.83 bytes.

Capítulo 5

Comparativa y Análisis

5.1. Resiliencia

Durante esta tesis se analizaron las características de resiliencia de dos aplicaciones: Geost y en SETI@home. En la Tabla 5.1 se muestra una comparación entre los métodos utilizados por Geost y por SETI@home relacionados con la resiliencia. Con respecto al acceso a los servidores, en Geost se observo que los dominios y las direcciones IPs se modifican en el tiempo, sin embargo, los dominios y direcciones IP usados por SETI@home son estables.

Lo que difiere es la técnica que se utiliza para la resiliencia. Mientras los sistemas maliciosos buscan ser resilientes mediante el uso de técnicas evasivas para evitar ser detectados y evolucionan en el tiempo, los sistemas estándar, como SETI@home, son resilientes porque aplican una administración de servidores redundantes para evitar fallas. Otra razón para hacerlos resilientes son las personas intermediarias y la conexión entre ellas. En los sistemas maliciosos las personas involucradas no se conocen entre ellas, es decir el desarrollador del APKs muchas veces no sabe quien es la persona que se ocupa de la distribución de los archivos. En los sistemas estándar, hay una (o más) persona(s) encargada(s) de que el sistema funcione y los usuarios tienen conocimiento de la identidad de esta persona. Por ejemplo, en el caso de SETI@home, las personas involucradas en el proyecto son: Eric Korpela (director), David P. Anderson (científico y arquitecto de software), Dan Werthimer (científico), Jeff Cobb (desarrollador de software y administrador de sistemas) y Matt Lebofsky (desarrollador de software y administrador de sistemas).

Cuadro 5.1: Comparación de las técnicas de resiliencia utilizadas en Geost y Seti at home, teniendo en cuenta los dominios y direcciones IPs, Arquitectura/Mantenimiento, características del APK y Heartbeat

Técnicas de resiliencia	**Geost**	**SETI@home**
Dominios e IPs	Cambio en los dominios/direcciones IPs	Mantiene los dominios y direcciones IP
Arquitectura/Mantenimiento	Descentralizada	Centralizada
Características del APK	Imposible de desinstalar	El cliente puede desinstalar BOINC
Heartbeat	Si	Si

Otra característica importante de las botnets como Geost es que la cantidad de víctimas crece continuamente ya que hay nuevos teléfonos que se infectan cada hora. Esto produce que si bien la *aplicación* pueda dejar de funcionar para el cliente, el servicio de la botnet como un sistema global sigue en funcionamiento y se expande continuamente. Esta es una diferencia fundamental con los sistemas HPC ya que necesitan autorización para tener nuevos teléfonos.

5.2. Accesibilidad

Con respecto a la accesibilidad, se puede afirmar que los sistemas analizados tienen una accesibilidad limitada. Con respecto al sistema malicioso la accesibilidad se reduce debido a cambios, es decir que los APKs que no tengan el dominio actualizado no acceden al sistema. Esto se debe a que la botnet modifica los dominios y las direcciones IPs para mantenerse resiliente. Por otro lado, SETI@home mantiene el mismo dominio funcionando desde hace años.

Luego del análisis se puede afirmar que el sistema de SETI@home es resiliente y que hay investigadores comprometidos con el proyecto para que esté funcionando. La accesibilidad no es de 24/7 ya que se observó hay momentos en los que se realizan tareas de mantenimiento y el servicio no está disponible. Esto se debe a que el proyecto no tiene fines comerciales por lo que la accesibilidad no parece ser una prioridad, es decir, si el servicio no funciona por algunas horas, no habrá consecuencias peligrosas o graves. Si el sistema está inactivo por cierto periodo de tiempo (por ejemplo un día), esto implica que se va a perder algunas horas de procesamiento, no afecta a todo el proyecto ya que las unidades de trabajo serán entregadas de todos modos.

Cuadro 5.2: Comparación de paralelismo y concurrencia en Geost y SETI@home

	Geost	**SETI@home**
Concurrencia	Método synchronized	Método synchronized
Paralelismo	Si	Si

Este sistema es distribuido y funciona en diferentes tipos de dispositivos.

5.3. Paralelismo y concurrencia

Ambos sistemas son de tipo distribuidos y no requieren que todos los dispositivos estén ejecutando la aplicación al mismo tiempo. En los dos casos hacen uso de los múltiples procesadores de los dispositivos móviles. Por lo que este recurso es utilizado de igual manera tanto en las aplicaciones maliciosas como en un tipo de aplicación HPC estándar. En la Tabla 5.2 se muestra la comparativa de los métodos de paralelismo y concurrencia utilizados por Geost y por SETI@home.

5.4. Computación heterogénea

Con respecto a la computación heterogénea, en ambos casos se hace uso de diversos procesadores con arquitecturas diferentes. En el Capítulo 4.5, se describen las características de SETI@home como aplicación de computación heterogénea. Los clientes que utilizan SETI@home tienen diferentes arquitecturas de procesadores y pueden seleccionar ejecutar SETI@home con uno o varios procesadores, por lo que si internamente poseen diferentes tipos de núcleos en un cliente se puede observar computación heterogénea. La misma situación se observa en Geost. Esto se resume en la Tabla 5.3. Geost tiene una ventaja porque tiene al menos 800,000 dispositivos y tienen dispositivos que se incorporan y aumenta la capacidad de computo. Por otro lado, con respecto a SETI@home, el crecimiento de la cantidad de dispositivos es menor porque los usuarios tienen que aceptar incorporarse al proyecto y colaborar de forma voluntaria. Es decir, que al ser ilegal e infectar a las víctimas que no tienen conocimiento de que están ejecutando la aplicación, el creci-

Cuadro 5.3: Computación Heterogénea entre Geost and SETI@home

Computación heterogénea	**Geost**	**SETI@home**
Como sistema	Si	Si
Dispositivo individual	Depende la arquitectura del cliente	Depende la arquitectura del cliente

miento tiende a ser mayor y mas rápido. En este sentido, la botnet es crece mas rápido y posee mas ventajas. Por otro lado, las aplicación estándar en una institución como sistemas individuales, poseen mejores características en cuanto a la capacidad de computo (CPU, RAM, optimización de recursos, etc) porque están diseñadas para resolver problemas complejos.

5.5. Ancho de banda

Con respecto al ancho de banda se observó que Geost hace menos uso del ancho de banda que la aplicación estándar SETI@home. En este caso de estudio, lo que la aplicación estándar envía son unidades de trabajo, que son resueltas de forma local en el dispositivo, los datos se devuelven al servidor principal una vez que se resolvió la unidad de trabajo. Por otro lado, Geost hace uso de la red, en este caso, para comunicarse con el servidor Command and Control, la aplicación intercepta los mensajes de texto y envía información sobre el dispositivo al Command and Control. En este experimento no se pudo terminar de demostrar si los datos que se enviaron son suficientes, ya que en servidor Command and Control no estaba activo, sino que era un *sinkhole*. Sin embargo desde el dispositivo se observó un intento de conexión hacia el servidor Command and Control. En la Tabla 5.4 se muestra el ancho de banda utilizado por ambas aplicaciones en términos de Bytes por segundo. La gran diferencia es que SETI@home va a utilizar el ancho de banda del dispositivo mientras el usuario no lo utilice, mientras que Geost puede hacer uso del ancho de banda del dispositivo de la víctima en cualquier momento. Esto hace una gran diferencia con respecto a SETI@home ya que, como se menciono en la sección 4.1.1, BOINC sólo se activa en los momentos en que el usuario no hace uso del teléfono celular y esta cargando la batería. Además, en la configuración por defecto, comienza a funcionar cuando el telefono celular tiene una carga de batería del 90 %. Teniendo en

Cuadro 5.4: Ancho de banda en Geost and SETI@home

Ancho de banda	**Geost**	**SETI@home**
Por dispositivo capturado	4 bytes/s	287bytes/segundo
Por dispositivo máximo subida	50 Mbit/s por día	Limitado

cuenta esta característica de SETI@home, suponemos que el uso total de la aplicacion por día promedio es de 3 horas (teniendo en cuenta que por defecto solo funciona cuando el dispositivo esta inactivo y cargando batería y con nivel de batería mínimo 90 %). Esta condición limita a SETI@home contra Geost, que esta activo las 24 horas. La relación en este sentido es que Geost esta activo 6 veces mas contra SETI@home. SETI@home utiliza 16 % de los recursos computacionales y de ancho de banda en comparación a Geost. En este sentido, Geost esta aprovechando mejor los recursos que SETI@home.

5.6. Discusión de los resultados

A continuación se resumen cada una de las características analizadas en esta tesis:

- **Resiliencia.** Luego de analizar las características de resiliencia en Geost, se puede afirmar que esta botnet es resiliente. Esta afirmación se basa en que Geost utiliza mecanismos para mantenerse activa sin ser detectada. Éstos mecanismos no son los típicos mecanismos utilizados por las aplicaciones de altas prestaciones estándar, ya que el objetivo es diferente, es decir, Geost se mantiene activa no solo mientras sus sistemas funcionen, pero también mientras no sea detectada. En el caso de Geost, se logra resiliencia cambiando las direcciones IPs y los dominios de los servidores Command and Control, desarrollando nuevas aplicaciones APKs y teniendo una infraestructura de negocio descentralizada, donde no todas las personas involucradas en el negocio de la botnet se conocen entre ellas. Por otro lado, la aplicación SETI@home, mantiene la resiliencia utilizando métodos tradicionales, es decir, redundancia en los datos, monitorizando el estado de los servidores y los servicios.

- **Paralelismo y concurrencia.** En ambos casos se puede considerar que los sistemas son paralelas, ya que corren al mismo tiempo en diferentes dispositivos. Mediante el análisis del código del APK de Geost,

se pudo observar que la aplicación utiliza una función llamada *synchronized* que es utilizada para sincronización de hilos. Es decir que en el código de esta aplicación también es concurrente. Por otro lado, la aplicación BOINC también hace uso de varios hilos, se observaron funciones como *synchronized y timer*

- **Computación heterogénea.** Esta característica se la tiene en cuenta desde dos perspectivas. Una es desde cada sistema (Geost y SETI@home) como una sola aplicación y la otra es desde cada uno de los dispositivos que están ejecutando la aplicación. Desde el punto de vista del sistema como un todo se puede sostener que ambas aplicaciones, tanto Geost como SETI@home, son aplicaciones que hacen uso de arquitecturas heterogéneas. Ambas se están ejecutando en dispositivos con diferentes características (RAM, memoria, CPU, etc). Sin embargo, Geost posee menos variabilidad de dispositivos ya que es una botnet que ataca a dispositivos con sistemas operativos Android, mientras que SETI@home también funciona en computadoras de escritorio y laptops (con sistemas operativos Windows, Mac y basados en Linux). Por otro lado, las aplicaciones se ejecutan en dispositivos de forma individual, y cada dispositivo tiene arquitecturas diferentes. Muchas celulares y tablets tienen CPU y GPU por ejemplo, pero ninguna de las aplicaciones hace uso de este recurso.

- **Ancho de Banda.** En ninguno de los casos observados se detectó un uso masivo del ancho de banda, o una necesidad de utilizar una red específica para estas aplicaciones. Las comunicaciones son pequeñas, sobre todo en Geost. La transferencia de datos es mayor en la aplicación SETI@home, ya que están transfiriendo porciones de datos para ser analizadas y se devuelven los resultados.

En la Tabla 5.5 se resumen las características de HPC que encontraron en Geost y en SETI@home.

La gran diferencia entre estas dos aplicaciones es el objetivo que persiguen. Por un lado Geost tiene como objetivo obtener dinero de forma ilícita, en este caso en particular mediante el robo de credenciales bancarias de las víctimas que instalan la aplicación. Por otro lado, la finalidad de SETI@home es la de resolver un problema, encontrar inteligencia extraterrestre. Si bien hay diferencias y similitudes en las características HPC de Geost y SETI@home, hay que tener en cuenta que el objetivo de cada aplicación es diferente.

Cuadro 5.5: Comparación de las características de HPC en Geost y SETI@home, teniendo en cuenta Resiliencia, Accesibilidad, Paralelismo y Concurrencia, Computación Heterogénea y Ancho de banda.

Características	**Geost**	**SETI@home**
Resiliencia	Si	Si
Accesibilidad	Limitada	Si
Paralelismo y concurrencia	Si (como sistema y en el código)	Si (como sistema y en el código)
Computación heterogénea	Si (como sistema y en algunos casos en ciertos dispositivos)	Si (como sistema y en algunos casos en ciertos dispositivos)
Ancho de banda	4 bytes/s	287 bytes/s

Con respecto a la potencia computacional, hasta ahora se estima que Geost llegó a tener 800,000 víctimas infectadas [41]. Por otro lado, con respecto a SETI@home, las estadísticas del proyecto indican que hay 1,793,135 usuarios registrados, pero sólo 89,310 están activos, eso es el 4.98 % del total de usuarios registrados [42]. Hay 166,406 dispositivos usando SETI@home, pero 146,644 de forma activa al momento de realizar la consulta (esto es el 88.12 % del total de los dispositivos). Esa estadística se puede observar en la Figura 5.1. La cantidad de usuarios utilizando SETI@home es mucho mayor, sin embargo la potencia computacional depende de el tipo de hardware que tengan los usuarios.

Con respecto al servidor Command and control utilizado en Geost, se lo puede comparar a un frontend de un sistema de un cluster de HPC estándar. Ya que el frontend gestiona la ejecución de los recursos de HPC pero no realiza ningún cómputo. Como se mostró en el Capítulo 3.1, el servidor Command and control contenía la información relacionada con los dispositivos infectados (estado, versión del Sistema Operativo, etc).

Otro elemento que se puede tener en cuenta para esta comparativa es el factor humano. En los sistemas HPC tradicionales no existe el anonimato, en el caso de SETI@home, los voluntarios pueden elegir que su nombre sea visible y hay una estadística y clasificación con los usuarios que más contribuyen al proyecto. Con respecto a la botnet Geost, los botmasters no se conocen

Average floating point operations per second	1,056,451.9 GigaFLOPS / 1,056.452 TeraFLOPS
Contribution to BOINC combined total credit	1.54181%
Contribution to BOINC combined total RAC	4.55347%
Users	1,793,135
Active users	89,310 (4.98%)
Hosts	166,406
Active hosts	146,644 (88.12%)

Figura 5.1: Estadísticas del poder computacional de SETI@home

todos entre ellos, es decir que el desarrollador de la aplicación no conoce al que distribuye esta aplicación en foros. Por otro lado, las víctimas no saben que fueron atacadas y no se conocen entre ellas.

Capítulo 6

Conclusiones

El aporte clave de esta tesis es la visión de sistemas botnet ilegales como sistemas HPC, dada su similitud en la utilización de recursos y la búsqueda de alto rendimiento en sus tareas. Para lograr este análisis se realizó un estudio comparativo sobre el uso de recursos computacionales de altas prestaciones entre un sistema ilegal, como el de la botnet Geost y un sistema HPC estándar como SETI@home. Con el objetivo de recavar datos se ejecutó la botnet Geost en teléfonos reales como también en emuladores, junto con un análisis de su código. Al mismo tiempo se ejecutó el sistema BOINC en teléfonos reales para analizar el uso de los recursos por la aplicación SETI@home. Desde estos datos se extrajeron ciertas características de los sistemas HPC en ambas aplicaciones. Las características que se tuvieron en cuenta son resiliencia, accesibilidad, paralelismo y concurrencia, computación heterogénea y ancho de banda. Finalmente, con los datos obtenidos, se compararon las características de HPC entre Geost y SETI@home.

Hasta la fecha no sabemos de ningún estudio previo que considere las características de altas prestaciones en botnets u otros sistemas maliciosos. Los trabajos previos en el área de seguridad se enfocan en la detección de malware mediante la extracción de características del tráfico de red, o bien el análisis del archivo binario del malware. En los trabajos previos de HPC no hay ninguna mención de análisis de sistemas ilegales como las botnets.

La extracción de características de HPC tanto en Geost como en SETI@home fue realizada desde dos perspectivas. Primero como sistemas completos (es decir toda la infraestructura necesaria para su operación), y segundo desde la perspectiva de los nodos individuales (es decir las capacidades de los teléfonos móviles y el software de Geost y SETI@home).

Podemos concluir que en el caso de Geost, la ilegalidad de la botnet hace que sea más fácil añadir nodos nuevos al sistema, con lo cual la botnet es más resiliente y tiene un flujo continuo de nuevos dispositivos que se integran al sistema. La ilegalidad también hace que sea muy fácil para las víctimas ser parte de la botnet. El malware de Geost sólo necesita ejecutarse una vez y no requiere de ninguna operación por parte de la víctima. Al contrario, el sistema SETI@home necesita autorización del usuario, la obtención de una cuenta oficial, y luego cierta configuración manual. Todo esto hace que haya muchos menos dispositivos que son parte SETI@home que le botnet Geost.

Sin embargo, desde el punto de vista de la resiliencia de los sistemas en el tiempo, el sistema SETI@home es oficial y no tiene a nadie tratando de darlo de baja, con lo cual el sistema sobrevivió por años. Geost sufre continuamente el riesgo de ser dada de baja por las autoridades y por lo tanto su promedio de vida es mucho mas corto.

Una limitación de este análisis consintió en la utilización de varias herramientas para obtener la información. Estas herramientas, detalladas en el Capítulo 2.4, son muy dependientes de el momento en el tiempo en el que se llevo a cabo en análisis, y de la disponibilidad de los datos. Al contrario que los sistemas HPC estándar, con los sistemas ilegales no se puede garantizar una extracción de datos completa. Esto fue una limitación que se tomo en cuenta en el análisis.

Las botnet son un tipo de malware que tiene control de los dispositivos infectados para realizar diferentes acciones, sin el consentimiento o conocimiento de las víctimas. Estas acciones son variadas dependiendo del tipo de botnet, por ejemplo, algunas utilizan a los dispositivos de las víctimas como servidores proxy y los venden en el mercado negro, otras realizan robos de credenciales bancarias. Por esta razón, una botnet se analiza la resiliencia de formas muy diferentes ya que que no tiene acceso legal a los dispositivos infectados y por lo tanto se basa en otras medidas para garantizar la disponibilidad. Entre las medidas más comunes están: tener una gran cantidad de dispositivos nuevos continuamente, mantener una gran flexibilidad en la incorporación de dispositivos nuevos en la botnet, permitir que la salida de dispositivos en la botnet no afecte la operación de la misma. Una parte importante de la resiliencia de la botnet reside en el software malicioso usado.

En particular el malware esta diseñado para que no sea fácil de encontrar por los analistas de seguridad, para que continúe ejecutándose en todo momento, y para que mantenga un contacto continuo con el dueño de la botnet. Como se explicó en el Capítulo 2.2, las botnets también tienen sistemas

de comando y control que evolucionan en sus capacidades y servidores que están activos por meses manteniendo la botnet activa y resiliente. Una parte importante de la resiliencia de una botnet es el diseño de la infraestructura, lo que implica de utilización de dominios y el diseño de utilización de direcciones IP, como se mencionó en el Capítulo 2.2.

Los dos sistemas HPC, Geost y SETI@home, están diseñados de forma completamente diferentes y en ambos casos requieren de un mantenimiento constante. En los sistemas HPC estándar hay uno o más administradores del sistema que monitorean el uso de los recursos, el estado de los equipos, los usuarios que hacen uso del sistema, etc. Por otro lado las botnets en general, y en el caso de Geost en particular, hay administradores del sistema con otro tipo de preocupaciones, como generar nuevos dominios, asignarles diferentes direcciones IP, generar dominios con características DGA, y administrar la botnet de forma tal que si uno de las personas que forma parte del equipo de trabajo decide abandonar o es arrestada, la botnet sigue funcionando. En el caso de las botnets también sucede que las responsabilidades de administración suelen recaer en una jerarquía de personas, en lugar de en un único grupo. En el caso de la botnet Geost hay personas encargadas del desarrollo de las páginas web, personas encargadas de incorporar nuevos dispositivos infectados a la botnet, personas encargadas de verificar estos dispositivos, etc. Usualmente estos roles se mantienen separados por una cuestión de seguridad del grupo que desarrolla la botnet.

Por otro lado, y como trabajo futuro, este análisis permite estudiar las razones por las cuales se realizan este tipo de ataques, es decir, cuales son las motivaciones de utilizar y desarrollar este tipo de botnets en lugar de aplicaciones estándar.

Otra de las características que se analizó es la de accesibilidad y disponibilidad, en el caso de Geost, provee accesibilidad pero cambiando de servidores. Es difícil saber si los servidores en algún momento tuvieron problemas y los bots no pudieron comunicarse. Sin embargo, observando el tráfico de red, se puede ver que los bots intentan acceder al canal cada un segundo cuando la conexión no es exitosa.

Durante el desarrollo de esta tesis se presentaron los resultados sobre el análisis de Geost en conferencias. Como trabajo futuro se pretende estudiar el factor humano y las motivaciones de las personas involucradas en Geost. El objetivo es analizar el ecosistema de los atacantes y comprender las motivaciones detrás del desarrollo e implementación de Geost. Otras de las preguntas abiertas que quedan pendientes para un trabajo futuro es si exis-

ten botnets mas sofisticadas que consideren el hardware subyacente de cada uno de sus bots y les envíen ordenes teniendo en cuenta esta característica.

Bibliografía

[1] *About SETI@home*. 2020. URL: https://setiathome.berkeley.edu/sah_about.php (visitado 05-2020).

[2] Luca Fraccascia ; Ilaria Giannoccaro ; Vito Albino. «Resilience of Complex Systems: State of the Art and Directions for Future Research». En: *Hindawi Complexity* 2018 (2018). DOI: https://doi.org/10.1155/2018/3421529.

[3] G. S. Almasi y A. Gottlieb. *Highly Parallel Computing*. USA: Benjamin-Cummings Publishing Co., Inc., 1989. ISBN: 0805301771.

[4] D. Anderson, Carissa Christensen y Bruce Allen. «Designing a Runtime System for Volunteer Computing». En: *ACM/IEEE SC 2006 Conference (SC'06)* (2006), págs. 33-33.

[5] David P. Anderson. «BOINC: A Platform for Volunteer Computing». En: *Journal of Grid Computing* 18.11 (mar. de 2020), págs. 99-122. ISSN: 0001-0782. DOI: 10.1007/s10723-019-09497-9. URL: https://doi.org/10.1007/s10723-019-09497-9.

[6] David P. Anderson y col. «SETI@Home: An Experiment in Public-resource Computing». En: *Commun. ACM* 45.11 (2002), págs. 56-61. ISSN: 0001-0782. DOI: 10.1145/581571.581573. URL: http://doi.acm.org/10.1145/581571.581573.

[7] David J. Wetherall Andrew S. Tanenbaum. *Redes de computadoras*. Pearson, 2011. ISBN: ISBN 9780132126953.

[8] Gregory R. Andrews. *Foundations Of Multithreaded Parallel And Distributed Programming*. Addison Wesley, 1999.

[9] *Android Debug Bridge*. 2020. URL: https://developer.android.com/studio/command-line/adb (visitado 03-2020).

[10] *BOINC. Compute for Science*. 2020. URL: https://boinc.berkeley.edu (visitado 05-2020).

[11] Clay Breshears. *The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications*. O'REILLY, 2009. ISBN: 978-0-596-52153-0.

[12] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems, Volume 1*. San Francisco, USA: Prentice Hall, 1999. ISBN: 978-0130137845.

[13] Walman Salamatian ; Wasim Huleihel ; Ahmad Beirami ; Asaf Cohen y Muriel Medard. «Why Botnets Work: Distributed Brute-Force Attacks Need No Synchronization». En: *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY* 14.9 (2019).

[14] Ron Minnich ; Andrew Sweeney ; Kristopher Watts ; Don Rudish John Floren ; David Fritz ; Keith Vanderveen ; Yung Ryn Choe ; Casey Deccio. «Large-scale Botnet Analysis on a Budget». En: (2011). URL: https://www.osti.gov/servlets/purl/1108338.

[15] Shuaike Dong y col. «Understanding Android Obfuscation Techniques: A Large-Scale Investigation in the Wild». En: *Security and Privacy in Communication Networks*. Ed. por Raheem Beyah y col. Cham: Springer International Publishing, 2018, págs. 172-192. ISBN: 978-3-030-01701-9.

[16] Nikolay Elenkov. *ANDROID SECURITY INTERNALS. An In-Depth Guide to android's security architecture*. San Francisco, USA: William Pollock, 2015.

[17] María José Erquiaga. «Botnets: Control and Propagation Mechanisms». En: *CACIC 2011* (2011), págs. 1076-1085.

[18] ETP4HPC. *European Technology Platform for High Performance Computing*. 2018. URL: https://www.etp4hpc.eu/pujades/files/ETP4HPC_book_singlePage.pdf.

[19] Joseph Gardiner, Marco Cova y Shishir Nagaraja. *Command & Control: Understanding, Denying and Detecting - A review of malware C2 techniques, detection and defences*. 2014. arXiv: 1408.1136 [cs.CR].

[20] *Genymotion*. 2020. URL: https://www.genymotion.com (visitado 03-2020).

[21] Michael Gschwind. «The Cell Broadband Engine: exploiting multiple levels of parallelism in a chip multiprocessor». En: *International Journal Parallel Programming* 35.3 (2007), págs. 233-262. DOI: 10.1007/s10766-007-0035-4.

[22] Georg Hager y Gerhard Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. CRC Press, 2011. ISBN: 978-1-4398-1192-4.

[23] *Jadx*. 2020. URL: https://github.com/skylot/jadx (visitado 03-2020).

[24] *Jadx Features Overview*. URL: https://github.com/skylot/jadx/wiki/jadx-gui-features-overview (visitado 03-2020).

[25] *Java methods*. 2020. URL: https://www.w3schools.com/java/java%5C_methods.asp (visitado 03-2020).

[26] Ian Foster ; Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, USA: Morgan Kaufman, El Sevier, 2003. ISBN: 9781558609334.

[27] I Koren y Mani Krishna. *Fault Tolerant Systems*. San Francisco, USA: Morgan Kaufman, El Sevier, 2007.

[28] Eric Korpela y col. «Status of the UC-Berkeley SETI efforts». En: *Proceedings of SPIE - The International Society for Optical Engineering* 8152 (ago. de 2011). DOI: 10.1117/12.894066.

[29] Kevin Lai y Mary Baker. «Measuring bandwidth». En: *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*. Vol. 1. IEEE. 1999, págs. 235-245.

[30] Evan Marcus y Hal Stern. «Blueprints for High Availability». En: (2003).

[31] *MobSF*. 2020. URL: https://github.com/MobSF/Mobile-Security-Framework-MobSF/ (visitado 03-2020).

[32] European Union Agency for Network e Information Security (ENISA). «ENISA threat landscape report 2018: 15 top cyber-threats and trends». En: (2019). DOI: 10.2824/622757. URL: https://doi.org/10.2824/622757.

[33] Floyd Piedad y Michael Hawkings. *High availability, Design, Techniques and process*. Prentice Hall, 2001. ISBN: 0-13-096288-0.

[34] Juan Carlos Saez; Adrian Pousa; Fernando Castro; Daniel Chaver; Manuel Prieto-Matias. «Towards completely fair scheduling on asymmetric single-ISA multicore processors». En: *Journal of Parallel and Distributed Computing* 102 (2017), págs. 115-131. DOI: 10.1016/j.jpdc.2016.12.011.

[35] R. Kumar ; D.M. Tullsen ; N.P. Jouppi ; P. Ranganathan. «Heterogeneous chip multiprocessors». En: *Annalen der Physik* 38.11 (2005), págs. 32-38. DOI: 10.1109/MC.2005.379.

[36] Arash Rezaei. *Fault Resilience for Next Generation HPC Systems*. Raleigh, North Carolina, USA: PhD Thesis. North Carolina State University, 2016. URL: https://arcb.csc.ncsu.edu/~mueller/ftp/pub/mueller/theses/rezaei-diss.pdf.

[37] *RiskIQ*. 2020. URL: https://riskiq.com (visitado 03-2020).

[38] Moray Rumney. *IMT-Advanced: 4G Wireless Takes Shape in an Olympic Year*. 2016. URL: http://cp.literature.agilent.com/litweb/pdf/5989-9793EN.pdf (visitado 03-2020).

[39] Victoria Sanz y col. «Accelerating Pattern Matching with CPU-GPU Collaborative Computing». En: *Algorithms and Architectures for Parallel Processing*. Ed. por Jaideep Vaidya y Jin Li. Cham: Springer International Publishing, 2018, págs. 310-322. ISBN: 978-3-030-05051-1.

[40] Anna Shirokova Sebastián García Maria Jose Erquiaga y Carlos Garcia Garino. «Geost Botnet: Operational Security Failures of a New Android Banking Threat.» En: *First Workshop on Attackers and Cyber-Crime Operations. IEEE European Symposium on Security and Privacy 2019* (2019). URL: https://wacco-workshop.eu.

[41] Maria Jose Erquiaga Sebastián García y Anna Shirokova. «Geost botnet. The discovery story of a new Android banking trojan from an OpSec error.» En: *Virus Bulletin 2019* (2019). URL: https://www.virusbulletin.com/blog/2019/10/vb2019-paper-geost-botnet-story-discovery-new-android-banking-trojan-opsec-error/.

[42] *Seti@home*. 2020. URL: https://setiathome.berkeley.edu/ (visitado 05-2020).

[43] Amar Shan. *Heterogeneous Processing: a Strategy for Augmenting Moore's Law*. 2006. URL: https://www.linuxjournal.com/article/8368.

[44] R.H. Spencer y R.E. Floyd. *Perspectives on Engineering*. AuthorHouse, 2011. ISBN: 9781463410919. URL: https://books.google.cz/books?id=dYVHOds2vQoC.

[45] Brett Stone-Gross y col. «Your botnet is my botnet: analysis of a botnet takeover». En: *Proceedings of the 16th ACM conference on Computer and communications security*. 2009, págs. 635-647.

[46] «Stratosphere Laboratory, Czech Technical University in Prague». En: (2020). URL: https://www.stratosphereips.org/ (visitado 03-2020).

[47] *TCPDump*. 2020. URL: https://www.tcpdump.org/ (visitado 03-2020).

[48] *Tensor Flow*. 2020. URL: https://www.tensorflow.org (visitado 05-2020).

[49] D. Toth, R. Mayer y W. Nichols. «Increasing Participation in Volunteer Computing». En: *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. 2011, págs. 1878-1882.

[50] «Virus Total». En: (2020). URL: https://www.virustotal.com (visitado 03-2020).

[51] Polly Wainwright y Houssain Kettani. «An Analysis of Botnet Models». En: *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*. ICCDA 2019. Kahului, HI, USA: ACM, 2019, págs. 116-121. ISBN: 978-1-4503-6634-2. DOI: 10.1145/3314545.3314562. URL: http://doi.acm.org/10.1145/3314545.3314562.

[52] *Worldwide LHC Computing Grid*. 2020. URL: https://wlcg-public.web.cern.ch (visitado 05-2020).

[53] Michael Armbrust ; Armando Fox ; Rean Griffith ; Anthony D. Joseph ; Randy Katz ; Andy Konwinski ; Gunho Lee ; David Patterson ; Ariel Rabkin ; Ion Stoica ; Matei Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing*. 2009. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.