UNIVERSIDAD INTERNACIONAL DE LA RIOJA

unir

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Detección y Clasificación de Malware con el Sistema de Análisis de Malware Cuckoo

Trabajo de Fin de Master

Presentado por: Rivera Guevara, Richard

Director/a: Bermejo Higuera, Javier

Ciudad: Madrid

Fecha: Septiembre 2018

Resumen

Actualmente el malware sigue siendo uno de los desafíos más grandes de la seguridad informática. El lanzamiento frecuente de nuevas variantes de muestras de una misma familia de malware, debido a las técnicas de ofuscación que utilizan los autores del software malicioso, hacen que la detección y clasificación del malware sea cada vez un problema más complejo. El presente trabajo tiene como propósito desarrollar un piloto experimental, con el fin de evaluar la exactitud que presenta un enfoque de clasificación de malware utilizando el módulo de firmas de Cuckoo SandBox. Se desarrolla n conjunto de 16 firmas de malware de familias conocidas que han sido evaluadas con dos conjuntos de datos, que comprenden un total de 16K muestras de software malicioso. . Al evaluar nuestro enfoque de clasificación con un conjunto de referencia, este alcanza un valor F-Measure de 91 %.

Palabras Clave: malware; clasificación; detección

Abstract

Currently malware is still one of the biggest challenges in computer security. The frequent release of new variants of samples from the same malware family, due to obfuscation techniques used by authors of malicious software, makes malware detection and classification a complex problem. The goal of this work is to develop an experimental pilot, to evaluate the accuracy of a malware classification approach using the Cuckoo SandBox signature module. A set of 16 malware signatures of known families are developed, which have been evaluated with two data sets, comprising a total of 16K samples of malicious software. The evaluation of our classification approach using a reference dataset as the ground thruth reaches a F-Measure value of 91 %.

***Keywords*:** malware, detection, classification

Índice general

Índice de figuras

Índice de tablas

Índice de archivos de código fuente

Capítulo 1

Introducción

Un reporte de seguridad de AV-TEST informa que su base de datos sumo en un año 127 millones de muestras de malware, lo que se traduce como 5 nuevas muestras de malware por segundo [5], otro reporte de KasperskyLab señala que en el primer cuartil del 2018 se detectaron más de 87 millones de muestras de software malicioso [6]. Cada vez más organizaciones criminales incursionan en el cibercrimen debido al bajo riesgo que este involucra y la alta rentabilidad que presenta [7]. Debido a estas tendencias muchas compañías y usuarios domésticos son el foco de los ataques de malware que pueden ser desde ataques de drive-by-download donde el atacante se aprovecha de alguna vulnerabilidad de la máquina objetivo para silenciosamente instalar el malware, hasta sofisticados ataques de ingeniería social donde el usuario es el que instala el malware sin darse cuenta de ello.

Por estas tendencias uno de los problemas fundamentales a los que se enfrentan los analistas e investigadores en seguridad informática y las empresas que venden productos de seguridad es la ofuscación que se presenta en el malware, es decir las variaciones de un malware de la misma familia, los autores del malware tratan de utilizar técnicas cada vez más sofisticadas de ofuscación, con la finalidad de que el sistema operativo o el software de seguridad no detecte su software malicioso. La ofuscación utiliza técnicas que hacen que un programa sea difícil de comprender [8], lo que complica que el malware se pueda detectar y clasificar en familias utilizando únicamente

características estáticas de los ficheros binarios, por esta razón se lo debe analizar de forma dinámica para entender su comportamiento e incrementar las tasas de detección y la precisión con que se clasifica.

El análisis dinámico del malware se realiza en ambientes controlados conocidos como SandBox [9]. Hoy en día podemos encontrar una gran cantidad de herramientas para analizar el malware, entre estas una de las SandBox más utilizadas es Cuckoo SandBox, un sistema de análisis de malware, que permite realizar análisis dinámico del malware en un ambiente controlado.

En este trabajo se implementará Cuckoo SandBox con varias personalizaciones para analizar un conjunto de muestras de software malicioso y utilizar las características dinámicas obtenidas de la ejecución, con el objetivo de generar firmas que permitan agrupar y clasificar el malware en familias. Estas firmas serán compartidas con la comunidad [10].

1.1 Estructura del trabajo

Este trabajo se estructura de la siguiente forma. En el capítulo 3 Se presenta los objetivos y la metodología utilizada en el desarrollo de esta investigación. En el capítulo 2 se presenta el estado del arte de la clasificación de malware. En el capítulo 4 se presenta el diseño y la arquitectura del piloto experimental que hemos desarrollado, también se describen algunas de las firmas utilizadas para la clasificación. En el capítulo 5 se presenta la evaluación de nuestra clasificación. Finalmente en el capítulo 6 presentamos las conclusiones de este trabajo y las futuras líneas de investigación que se podrían derivar de este.

Capítulo 2

Contexto y estado del arte

En la actualidad cualquier usuario que se conecte a Internet enfrenta muchos problemas de seguridad. Ahora los ataques pueden ser muy sofisticados utilizando técnicas complejas o ataques que no requieren mucho conocimiento técnico en su lugar utilizan técnicas de ingeniería social, muchos de estos ataques suelen depender de un vasto ecosistema de software y herramientas maliciosas. Por otra parte las clásicas formas de ataques [11] como el spam, phishing, denegación servicio, botnets, troyanos, gusanos entre otros, usualmente dependen en gran medida de código malicioso comúnmente conocido como malware. El malware se utiliza a menudo para infectar los ordenadores de las víctimas, explotar vulnerabilidades de otro software o de vulnerabilidades en la configuración de seguridad, o engañar usuarios para que ejecuten el malware permitiendo al atacante acceder a un dispositivo de forma remota, o instalar keyloggers para robar credenciales de acceso, entre otras funciones que puede realizar el malware.

2.1 Malware

El software que cumple deliberadamente los objetivos maliciosos de un atacante se conoce como software malicioso o malware [12]. Estos objetivos maliciosos tienen como propósito obtener acceso a los sistemas y recursos de red, irrumpir las operaciones

computacionales y obtener información personal de los usuarios sin su consentimiento.

El malware se presenta en un gran rango de variaciones, como virus, gusanos, troyanos, rootkits, backdoors, botnets, spywares, etc. Estos tipos de malware no son mutuamente excluyentes, un malware particular puede tener varios objetivos maliciosos, revelando características de múltiples tipos a la vez.

Las infecciones de malware han sido una de las principales preocupaciones para toda la comunidad que esta inmersa en la seguridad informática. Se estima que cada día se tiene mas de un millón de nuevas amenazas [13]. En un estudio realizado por FireEye se encontró que un 47 % de las organizaciones experimentan incidentes de seguridad relacionados con el malware cada año. Para contrarrestar esto, las empresas que venden productos de seguridad como los antivirus, intentan estar al día con la detección de nuevas amenazas para proteger a sus usuarios. Pero esta detección se vuelve cada vez más compleja debido a la sofisticación del malware.

2.1.1 Programas Antivirus

Un programa antivirus trata de detectar, identificar y contener el malware antes de que este pueda llegar a infectar un sistema o red [14]. Los primeros programas de antivirus iniciaron su comercialización a inicios de la década de 1990, como respuesta a los primeros virus que se identificaron, en un inicio fueron pensados unicamente para la detección de virus, de allí su nombre, pero con el incremento en el numero de amenazas, la proliferación de otros tipos de malware y el constante incremento de usuarios que acceden a Internet la industria de los antivirus se ha ido incrementando considerablemente, proveyendo seguridad para más tipos de amenazas. Hoy en día con el incremento del uso de la Internet y los dispositivos móviles, hay más de 60 empresas que ofertan programas antivirus [15], y en algunos casos estas empresas cuentan con varios motores de antivirus en su catalogo para varios tipos de usuarios (e.g., usuarios empresariales o domésticos).

Con tantas opciones de programas de antivirus en el mercado, varias empresas como Gartner [16], proporcionan resultados de estudios de tendencias tecnológicas sobre

las empresas que ofrecen programas de antivirus como se muestra en la Figura 2.1. Por otra parte otras empresas como AV-Test [17] o AV-Comparatives [18] analizan los software antivirus de una forma más técnica proporcionando a los usuarios análisis del rendimiento de estos programas, sus tasas de detección entre otras características. que permiten analizar la calidad de un programa antivirus. En la Figura 2.2 se muestra un análisis de AV-Comparatives [18], donde se presenta para varios antivirus el ratio de detección y la cantidad de falsos positivos detectados. Se debe tener en cuenta que si un producto tiene un alto porcentaje de detección de archivos maliciosos, pero que también presenta un alto numero de falsos positivos no quiere decir que sea un mejor producto, pues puede estar detectando programas que son benignos como maliciosos. Estas discrepancias que se presentan entre los programas de antivirus, se debe a los distintos métodos de detección y a que los antivirus utilizan técnicas de detección avanzadas, pero los programadores de malware usan técnicas de evasión aún más avanzadas [19, 20], para transformar al malware con el fin de que pueda evitar la detección.

Figura 2.1: Cuadrante Mágico de Gartner 2018 de Endpoint Protection Platforms (enero 2018). Fuente Gartner [1]

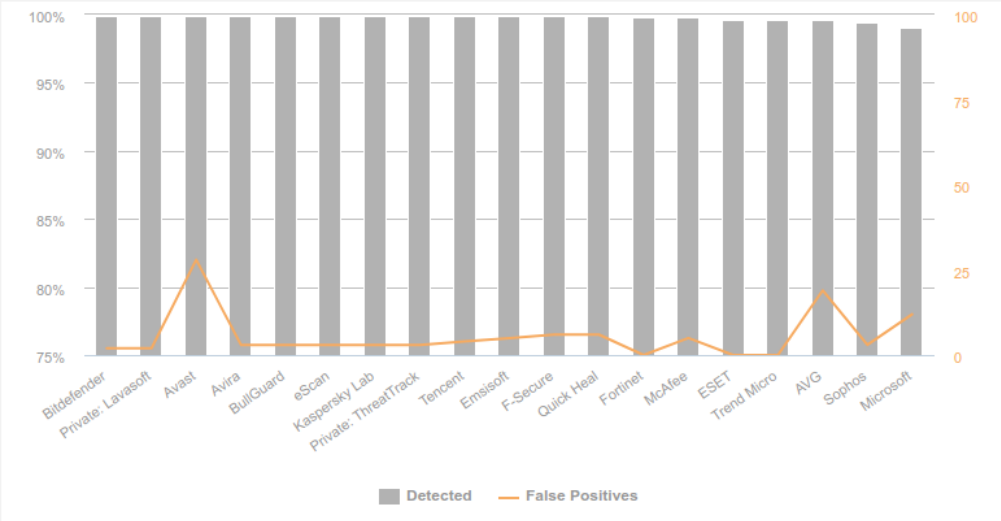

Figura 2.2: Comparativa de Programas de malware sobre el ratio de detección y falsos positivos. Fuente AV-Comparatives [2].

2.1.2 Técnicas de Detección de Malware

El software malicioso o malware ha avanzado en varios aspectos, como en la complejidad de su código, las técnicas de ofuscación, los vectores de distribución. Para llevar a cabo sus objetivos maliciosos los atacantes explotan vulnerabilidades en servicios web, exploradores de Internet y los sistemas operativos, o utilizan técnicas de ingeniería social persuadiendo al usuario para que ejecute el malware. De la misma forma los programas de antivirus también han ido evolucionando, mejorando sus técnicas de detección para poder lidiar con los nuevos malware. Las técnicas de detección de malware son categorizadas desde distintos puntos de vista, dependiendo del enfoque que se haya utilizado para su detección [21] . En este trabajo se han separado las técnicas en 3 categorías ilustradas en la Figura 2.3.

Basadas en firmas Esta es la técnica de detección mas utilizada por los antivirus, pues permite detectar con rapidez al software malicioso, la dificultad de esta técnica se presenta debido a que el programa de antivirus debe estar en constante actualización

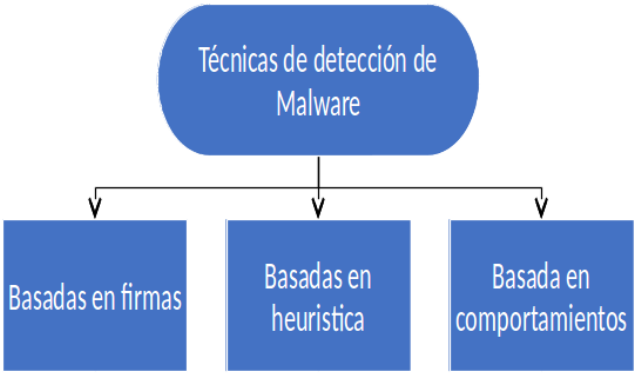

Figura 2.3: Técnicas de detección de malware.

para obtener las nuevas firmas generadas. Si este no es actualizado constantemente el sistema puede ser vulnerable a nuevas amenazas. Las firmas de esta técnica son de dos tipos, las firmas de Hash, y las de Bytes. En la primera el antivirus compara el hash (e.g., MD5 message-digest algorithm, SHA1, SHA256, etc) del archivo analizado con hashes de malwares conocidos, esta técnica es rápida y funciona bien siempre que el malware se encuentre en la base de datos del antivirus, y el malware que se este analizando no haya sido modificado, pues si a un fichero se le modifica un solo carácter el hash resultante será distinto. La segunda firma representa una secuencia de bytes que están presentes en el fichero analizado [22, 23]. Esta técnica suele ser efectiva para detectar familias de malware, analizando secuencias de bytes que tienen en común variaciones de malware de una misma familia.

Una herramienta ampliamente usada para la creación de firmas es YARA [24]. Esta herramienta es creada con el objetivo de ayudar a los investigadores de malware en la detección y clasificación de muestras de malware. Con YARA se crea firmas que consisten en un conjunto de cadenas de caracteres y expresiones booleanas que determinan la lógica de las muestras.

Basadas en heurística El análisis heurístico utiliza reglas y/o algoritmos que buscan comandos (e.g., API calls, OpCode, relaciones de archivos, etc.) que indican que el

fichero analizado tiene intenciones maliciosas [25]. Usando esta técnica, algunos motores de antivirus son capaces de detectar el malware sin utilizar firmas, puesto que muchas de los motores de antivirus utilizan las dos técnicas heurísticas y basadas en firmas para detectar malware. Esta técnica a pesar de ser rápida y efectiva en muchos casos, si las reglas no son mejoradas constantemente pueden provocar falsos positivos.

Basadas en el comportamiento Las técnicas basadas en el comportamiento son más avanzadas, pues estas ejecutan al malware en un ambiente controlado, para conocer todas las acciones que este realiza durante el proceso de infección. La información que se obtiene de la ejecución del malware se utiliza para la agrupación o clasificación del malware [26].

2.1.3 Analisis de Malware

Cuando se analiza el malware se puede considerar dos tipos de análisis, estático y dinámico. El análisis estático utiliza técnicas de ingeniería inversa, permiten analizar las instrucciones del programa, las cuales pueden utilizarse posteriormente para la detección heurística. El análisis dinámico se utiliza para la detección basada en el comportamiento, pues este análisis ejecuta al malware para entender el comportamiento que le lleva a alcanzar su objetivo.

El análisis estático frecuentemente se realiza antes del análisis dinámico, ya que este último demanada más recursos, y los resultados del análisis estático pueden servir de entrada para el análisis dinámico. No en todos los casos es necesario realizar los dos tipos de análisis, esto depende de los objetivos que tenga el analista.

Análisis estático Para realizar el análisis estático de un archivo malicioso se puede utilizar dos enfoques. El primero que demanda menos recursos, es realizar un análisis de las caracteristicas estáticas del archivo, es decir los metadatos, certificados, cadenas de caracteres. El segundo requiere más conocimiento y experiencia sobre el tipo

de archivo, por ejemplo al analizar un ejecutable portable (PE) de Windows [27], se necesita conocer la estructura de un archivo PE. En este tipo de análisis se estudia el código ensamblador de la muestra, las instrucciones de este código permiten que el analista pueda comprender el objetivo y funcionamiento de la muestra. También se utilizan herramientas especificas [28] para realizar el proceso de ingeniería inversa, que consiste en pasar del lenguaje ensamblador a un lenguaje de más alto nivel, que permita identificar las funciones que utiliza la muestra y que es lo que pretende. Uno de los problemas a los que se enfrenta este tipo de análisis es la ofuscación del código, que utilizan los autores del malware para que su código no pueda ser comprendido por los analistas.

Análisis dinámico Para realizar el análisis dinámico de una muestra se ejecuta el malware en un ambiente controlado, con la finalidad de observar su comportamiento y todos las modificaciones que esta ejecución realiza en el sistema. En este análisis realiza un monitoreo de la interacción que tiene el malware con el sistema, algunas de las acciones que se pretende identificar son las siguientes [29]:

- Creación, eliminación, lectura y escritura de archivos en el sistema.
- Creación, eliminación, lectura y escritura de claves en el registro de Windows.
- Conexiones de red detallando el protocolo, origen, destino, puerto, etc.
- Creación y finalización de procesos con sus parámetros y desglose de las funciones utilizadas y llamadas al API de Windows.

Es fundamental tener en cuenta que este tipo de análisis se debe realizar en un entorno controlado de análisis de malware, para evitar que alguna actividad maliciosa pueda propagarse. En la Figura 2.4 se representa los procesos usualmente seguidos por el análisis dinámico. Primero se debe crear una imagen (snapshot) del sistema antes del análisis de la muestra, esta imagen se puede utilizar para revertir el estado del sistema para estudiar más muestras o para comparar las diferencias entre el estado

inicial y el estado después del análisis. El segundo paso es ejecutar la herramienta que estará constantemente monitoreando los cambios que el malware realice en el sistema. A continuación se ejecuta el malware, una vez que todos los procesos que se generaron con la ejecución del malware hayan terminado, se procede a detener la herramienta de monitoreo. Como el malware ha terminado su ejecución, se recolecta la información de la herramienta de monitoreo, esto puede realizarse generando un reporte de la ejecución. Finalmente se procede a revertir el sistema a su estado inicial para que este pueda estar listo para realizar más análisis.

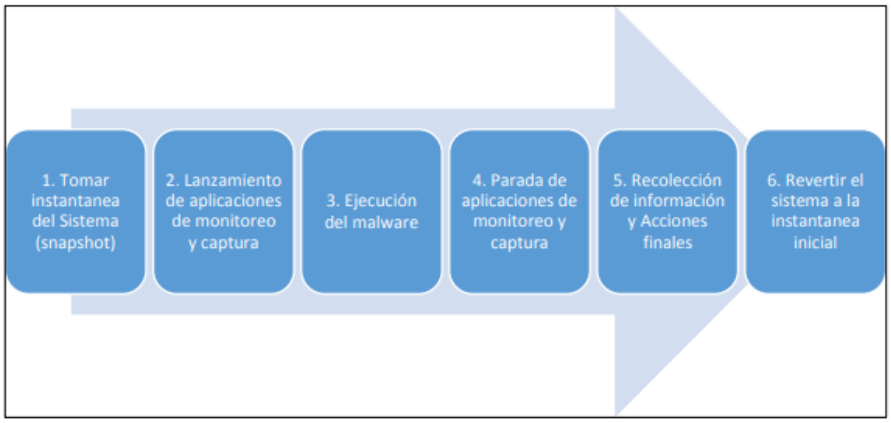

Figura 2.4: Técnicas de detección de malware.

Las compañias de la industria de los antivirus utilizan todas las técnicas de detección de malware estudiados en la Sección 2.1.2, así como los análsis estudiados en la Sección 2.1.3, para proporcionar a los usuarios soluciones más completas que presenten mejores ratios de detección con la menor cantidad posible de falsos positivos. Una vez que los antivirus detectan el malware también le asignan un nombre(label), en base al mecanismo de detección que hayan utilizado para encontrar cada malware.

2.1.4 Etiquetado (Labeling)

Desde el punto de vista de un antivirus, la etiqueta que le asignan a una muestra maliciosa , es solo el nombre para identificar a un software maligno. Etiquetar un soft-

ware malicioso como variante de una familia es importante para varias aplicaciones de seguridad, como identificar nuevas amenazas, seleccionar mecanismos de desinfección, atribución, etc. El etiquetado puede ser realizado manualmente por analistas o automáticamente por enfoques de clasificación de malware utilizando aprendizaje automático supervisado [30–32] y también utilizando enfoques de agrupamiento de malware [12, 33–35] seguidos de un proceso manual de etiquetado para asignar un nombre de familia a cada grupo.

En la Figura 2.5 podemos observar una ventana que suele mostrarse cuando el software antivirus instalado en el sistema detecta un software malicioso. Una vez que el malware es detectado el antivirus le asigna un nombre al cual se le conoce como el AV Label.

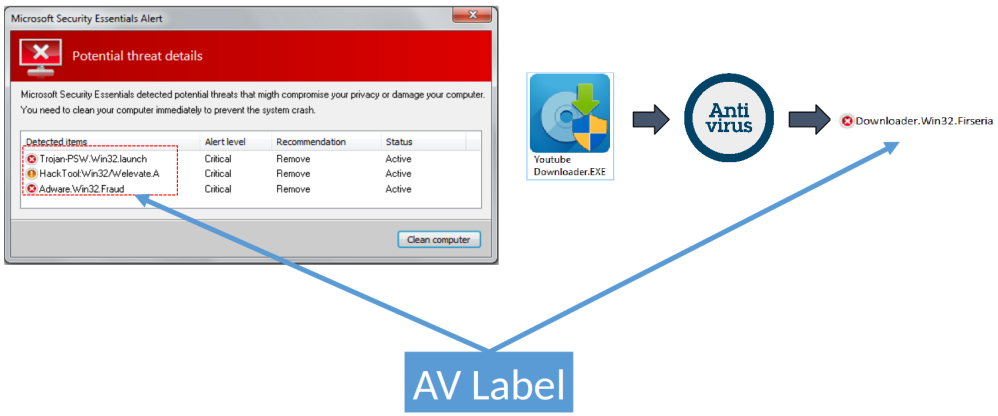

Figura 2.5: AV Label.

Se sabe que estos AV labels presentan inconsistencias [20, 33, 36, 37]. Particularmente los antivirus suelen estar en desacuerdo en si la misma muestra de malware es malicioso o benigno, así como también la familia a la que le asignan puede diferir de un antivirus a otro, esto se debe a la falta de un estándar en la asignación de nombres. Para esto se han propuesto varios estándares como CARO [38] y CME [39], pero estos no han tenido la acogida esperada y no son muy utilizados. Los antivirus no utilizan dichos estándares debido a que su objetivo principal es la detección más no la clasificación.

A pesar de las inconsistencias antes mencionadas, los AV labels son la fuente más común para extraer las etiquetas del malware. Esto debido a que en muchas ocasiones no se cuenta con un conjunto de datos verdaderos(Ground truth). Aún cuando la información que se puede obtener de los AV labels contiene mucho ruido, estos suelen contener el nombre de la familia que el analista necesita.

Kotzias, et al. propusieron AVClass [4], una herramienta que actualmente es ampliamente usada, esta implementa técnicas automáticas para etiquetar el malware, AVClass normaliza los AV labels, remueve los tokens genéricos dentro de este y realiza una detección de alias, esto es que agrupa a muestras de malware que son de la misma familia pero los antivirus les han asignado a familias distintas.

2.1.5 Tipos de malware

Desde la aparición de los primeros malware, estos han aparecido ejecutando distintas acciones maliciosas. Desde simplemente presentar al usuario contenido no deseado, hasta tomar completamente el control de los sistemas, restringiendo el acceso al propio usuario a su sistema. La aparición y evolución de nuevas muestras de malware es muy rápida, actualmente tenemos una gran variedad de tipos de malware, que incluyen virus, gusanos, troyanos, rootkits, backdoors, spyware, adware, pup, bots, ransomware, rogueaware, APTs entre los más comunes. Estos tipos de malware se agrupan de acuerdo al tipo de actividad maliciosa que realizan, más específicamente al vector de ataque que utilizan. El diferenciar el tipo de malware de una muestra es útil para identificar las contra medidas que se debe utilizar para detener el ataque.

Es muy importante diferenciar esta clasificación por tipos de malware, y la clasificación que se realiza por familias de malware. Este trabajo como la mayoría de trabajos de clasificación de malware [4,26,31,33,35,40], se centra en la clasificación entre familias de malware, dado que una muestra sofisticada de software malicioso dependiendo de su objetivo puede encajar en varios tipos de malware. Una familia de malware nos indica las muestras de malware utilizar varios vectores de ataque. Por esta razón es importante clasificar a las muestras en familias de malware con la mayor precisión po-

sible. Una familia de malware comparte patrones de comportamiento que reflejan su origen y propósito [31].

2.2 Clasificación de Malware

Cada día se generan miles de nuevas muestras de software malicioso [6, 13], estas en su gran mayoría son variaciones de familias existentes de malware, por esta razón es importante tener una alta precisión en la clasificación de cada nueva muestra de malware. Los sistemas de antivirus actuales tienen como propósito principal detectar el software malicioso, sin darle mucha importancia a la clasificación. Por otra parte los autores del malware apoyados en el desarrollo de la tecnología, tratan constantemente de innovar el malware haciendo que su software sea cada vez más sofisticado e implementan técnicas que dificultan su análisis. Estas técnicas anti-análisis pueden incluir la ofuscación del código mediante el polimorfismo, metamorfismo, el cifrado o técnicas anti-SandBox.

2.2.1 Técnicas de clasificación de malware

Para solventar los desafíos que conlleva realizar la clasificación del malware, se han realizado varias investigaciones que plantean enfoques orientados al análisis del comportamiento del malware, como se menciona en la Sección 2.1.3 esto se realiza mediante el análisis dinámico del malware en ambientes controlados. A continuación se repasa brevemente algunos de estos enfoques.

Rieck, et al. [31] proponen un enfoque basado en el aprendizaje para la clasificación automática del comportamiento del malware. Su enfoque se desarrolla en tres fases. Primero, analizan de forma dinámica las muestras ejecutándolas en una SandBox para recolectar el comportamiento del malware. Segundo, utilizan un motor de anti-virus para etiquetar las muestras, y las utilizan para entrenar al clasificador mediante técnicas de aprendizaje automático. Finalmente, generan un ranking de las características más prominentes de los modelos de comportamiento, permitiendo agrupar muestras

de malware que comparten dichos comportamientos.

Bayer et al. [26] presentaron un enfoque de clustering escalable para agrupar e identificar muestras de malware en familias. Para esto primero realizan un análisis dinámico para conseguir las trazas de ejecución de las muestras analizadas. Estas trazas se utilizan para generalizar y resumir los comportamientos de la muestra en perfiles de comportamiento, los cuales expresan el comportamiento en términos de operaciones y objetos del sistema operativo. Finalmente presentan un algoritmo que eficientemente agrupa las muestras de malware de acuerdo a sus perfiles de comportamiento.

Rafique y Caballero presentan FIRMA [35] una herramienta que recibe un potencial grupo de tráfico grande de red, el cual se ha obtenido ejecutando muestras de malware sin clasificar. FIRMA procesa esta información y genera clusters de familias de las muestras, así como un conjunto de firmas basadas en los comportamientos de red específicos para cada una de las familias identificadas.

2.3 Entornos de Análisis de Malware

Los investigadores de seguridad informática a menudo se ven en la necesidad de realizar un análisis de malware, esta tarea requiere que se realice un análisis estático y dinámico en un entorno controlado de modo que el comportamiento del malware no afecte a los dispositivos de computo utilizados en dicho análisis. Algunos investigadores optan por desarrollar sus propios entornos de análisis de malware, esto depende de los objetivos que se pretendan alcanzar, o las preguntas de investigación que se plantean solventar, en otros casos se opta por utilizar entornos de análisis de malware ampliamente utilizados en la industria o en el ámbito académico. Implementar un entorno de análisis de malware, es una tarea que requiere de una preparación planificada, pues las decisiones que se tomen en el diseño pueden influir en el rendimiento y los resultados de los análisis [41].

Para analizar el malware se han propuesto varios marcos de trabajo o entornos.

2.3.1 Sandbox

Una sandbox es una plataforma para analizar ficheros desconocidos en un ambiente controlado, sin los riesgos que puede conllevar ejecutarlos en un entorno de producción. Básicamente, una SandBox utiliza entornos virtualizados que simulan un ambiente físico para asegurar que los ficheros analizados se comporten lo mas cercano posible a como lo harían en el ambiente real, con la finalidad de proporcionar información. Desde el punto de vista de la seguridad una SandBox se utiliza para ejecutar software malicioso en un ambiente seguro para obtener información sobre su comportamiento, sobre el tipo de ataque, etc., para que los administradores de seguridad de las organizaciones tomen las medidas necesarias.

Otro aspecto importante de una SandBox es que esta disminuye en gran medida el esfuerzo humano que derivaría la compleja y larga tarea de desensamblar un ejecutable para entender su propósito. Esto permite que los analistas de seguridad con poca experiencia en análisis de malware puedan realizar el triaje de las muestras sospechosas y solo aquellas que se confirmen como maliciosas sean analizadas más detalladamente.

Hoy en día muchos de los productos y herramientas de seguridad en el mercado, utilizan uno o más tipos de SandBox para el análisis del comportamiento, varias de estos productos son de pago (privativas), pero también se puede encontrar soluciones notables que se distribuyen como software libre. A continuación se describen algunas de estas SandBox.

Anubis Es una herramienta de análisis dinámico automatizada [42] basada en TTanalyze otra herramienta para analizar el comportamiento de ejecutables de Windows [12]. Anubis tiene una versión comercial de pago anunciada bajo el nombre *Lastline Analyst*. Anubis usa el emulador QEMU para ejecutar los potenciales software maliciosos en el sistema operativo Windows XP. Utiliza también una segunda maquina virtual que proporciona varios servicios de red para que sean ser explotados por las muestras de malware ejecutadas. Anubis retorna un reporte de alto nivel que lista los archivos, pro-

cesos, registros y la actividad de red. Este reporte es útil para que los analistas revisen el comportamiento de una muestra con respecto a las modificaciones realizadas en el sistema y las conexiones de red que se han establecido durante la ejecución.

Zerowine Es un sistema [43] de software libre que dinámicamente analiza el comportamiento de aplicaciones utilizando Wine, recolectando la información de las llamadas al API del sistema. La desventaja de esta solución es que esta solo analiza aplicaciones de Windows emulando el sistema.

Joe Sandbox Es una plataforma [44] de análisis de malware dinámico que soporta máquinas físicas y virtuales. Para cada muestra se realiza el monitoreo de las llamadas al sistema y al API. Joe Sandbox simula la actividad que realizaría un usuario utilizando los scripts de AutoIT [45], los cuales permiten por ejemplo automatizar la interacción con instaladores. Esta plataforma entrega una lista de las actividades del sistema, recolectando también los archivos que se han escrito en el sistema, así como las trazas de red, toda la información del comportamiento de la muestra se puede obtener en varios formatos de reporte con un alto nivel de detalles de cada análisis. Mediante un modulo especial la herramienta también soporta el análisis estático.

CWSandbox Es una plataforma [46] de análisis dinámico de malware que al igual que su sucesor comercial GFI Sandbox/ThreatAnalyzer [47], pueden utilizar maquinas físicas o virtuales bajo sistemas Windows, donde el análisis se realiza mediante monitoreo de las funciones que realizan llamadas al API de Windows. Al igual que Anubis y Joe Sandbox esta plataforma entrega un reporte que lista los archivos, registros, conexiones de red y otras operaciones del sistema operativo que hayan sido realizadas por la muestra analizada.

FireEye Malware Analysis System Es la versión de laboratorio de análisis de malware de la linea de productos FireEye [48]. Este sistema permite realizar análisis de varios

formatos de máquinas de virtuales, también permite utilizar imágenes de pre configuradas del sistema operativo que contenga software pre instalado (e.g., Flash Player, Adobe Reader). A diferencia de los sistemas o herramientas descritos anteriormente FireEye viene incluido con su propio hardware de varias especificaciones, dependiendo de los requerimientos de los analistas. El sistema retorna una traza textual que incluye información de los archivos, reglas de YARA que han tenido coincidencias y alertas maliciosas provenientes de ciertas llamadas al API o actividades de procesos, previamente especificados. Comparado con las otras plataformas FireEye ofrece la posibilidad de generar los reportes en múltiples tipos de formato, pero estos son menos detallados, dado que esta utiliza un enfoque basado en alertas previamente configuradas de comportamientos maliciosos.

Wildfire Malware Analysis Es un servicio basado en la nube de Palo Alto Networks [49], para identificar y bloquear malware desconocido. Este servicio se integra con los firewalls de Palo Alto Networks, cuando uno de estos firewalls detecta una muestra desconocida (un archivo o un enlace incluido en un correo electrónico), el firewall puede reenviar automáticamente la muestra para el análisis en WildFire. En función de las propiedades, los comportamientos y las actividades que presente la muestra cuando se analiza y ejecuta en el entorno controlado de WildFire. Este determina si la muestra es benigna, gris o maliciosa. A continuación, WildFire genera firmas para reconocer el malware recientemente descubierto y hace que las firmas más recientes estén disponibles globalmente para todos los demás firewalls de Palo Alto Networks.

2.3.2 Análisis en linea de malware

Es importante resaltar que existen sitios web que permiten a los usuarios subir archivos sospechosos para que estos sean analizados, haciendo que la necesidad de implementar alguna de las SandBox antes mencionadas para analizar pocas muestras maliciosas sea innecesario, teniendo en cuenta que este método no proporcionará los

mejores resultados, ya que al ser un servicio web externo, no se podrán hacer personalizaciones en los análisis que se realice. A continuación se describen brevemente algunos otros sitios web populares que ofrecen este servicio.

VirusTotal Es un servicio en linea [50] que examina archivos o URLs enviados por usuarios con una gran cantidad de herramientas de seguridad, entre estas los archivos son analizados con más de 70 motores de antivirus. VirusTotal permite a cualquier usuario subir archivos para ser analizados a través de su sitio web, además de consultar el análisis de cualquier otro archivo mediante algún hash de este. También se puede consultar estos reportes o subir archivos mediante el API que proporcionan. El reporte de VirusTotal no solo incluye la información de si algún antivirus detecto el archivo analizado como malicioso, sino que también muestra el AV label (e.g., I-Worm.Allaple.gen) de cada motor de antivirus que lo analizó. VirusTotal ofrece un servicio llamado *VirusTotal Malware Intelligence Services* el cual permite a los investigadores obtener muestras analizadas así como sus reportes, los cuales han sido utilizados para investigaciones que proponen mejoras en la detección y clasificación de malware [4, 51–54] y para otros análisis sobre el comportamiento del software malicioso [36, 55–58].

Anubis De las SandBox analizadas en la Sección 2.3.1 Anubis [42] también ofrece el servicio de análisis de malware online, en esta plataforma los usuarios pueden subir ejecutables de Windows o archivos de Android APKs para ser analizados.

Malwr Es una plataforma online [59] diseñada por el mismo equipo que diseño Cuckoo SandBox. En esta plataforma a los usuarios se les permite subir archivos o URLs para analizarlas. Adicionalmente los usuarios también pueden ver los reportes de ficheros subidos por otros usuarios, si quien subió originalmente el archivo configuro para que el reporte del análisis sea publico.

2.3.3 Cuckoo

Cuckoo Sandbox es un sistema de análisis de malware [60]. El sistema recibe cualquier archivo sospechoso, lo analiza y presenta un reporte detallado de lo que hizo el archivo durante su ejecución dentro del ambiente aislado. Cuckoo SandBox es un software libre que automatiza las tareas de análisis de malware o ficheros maliciosos para las plataformas Windows, OS X, Linux, y Android.

2.3.3.1 Características de Cuckoo

Cuckoo permite realizar un análisis estático y dinámico del malware en un ambiente controlado. consiste en un software de administración central el cual gestiona la ejecución y análisis de las muestras. En la Figura 2.6 se presenta la arquitectura en alto nivel de Cuckoo. Cada análisis es ejecutado en una máquina virtual o física aislada, creando un ambiente controlado. Los principales componentes de la infraestructura de Cuckoo son la maquina anfitrión (donde se encuentra el software de gestión), y un número de máquinas clientes (máquinas virtuales o físicas para los análisis). La máquina anfitriona ejecuta el componente núcleo de Cuckoo, la cual gestiona las maquinas clientes, inicia los análisis intercepta el tráfico de red y genera los reportes. Mientras que las maquinas cliente están en un ambiente totalmente controlado, permitiendo que las muestras de malware se puedan ejecutar y analizar de forma segura, el comportamiento de la muestra es reportada a la maquina anfitrión de Cuckoo.

Cuckoo es extremadamente modular y 100 % software libre con infinitas posibilidades de aplicación. Por defecto es capaz de realizar tareas que se presentan a continuación.

- Analizar diferentes tipos de ficheros maliciosos:
- Genéricos ejecutables de Windows
- Archivos DLL
- Documentos PDF

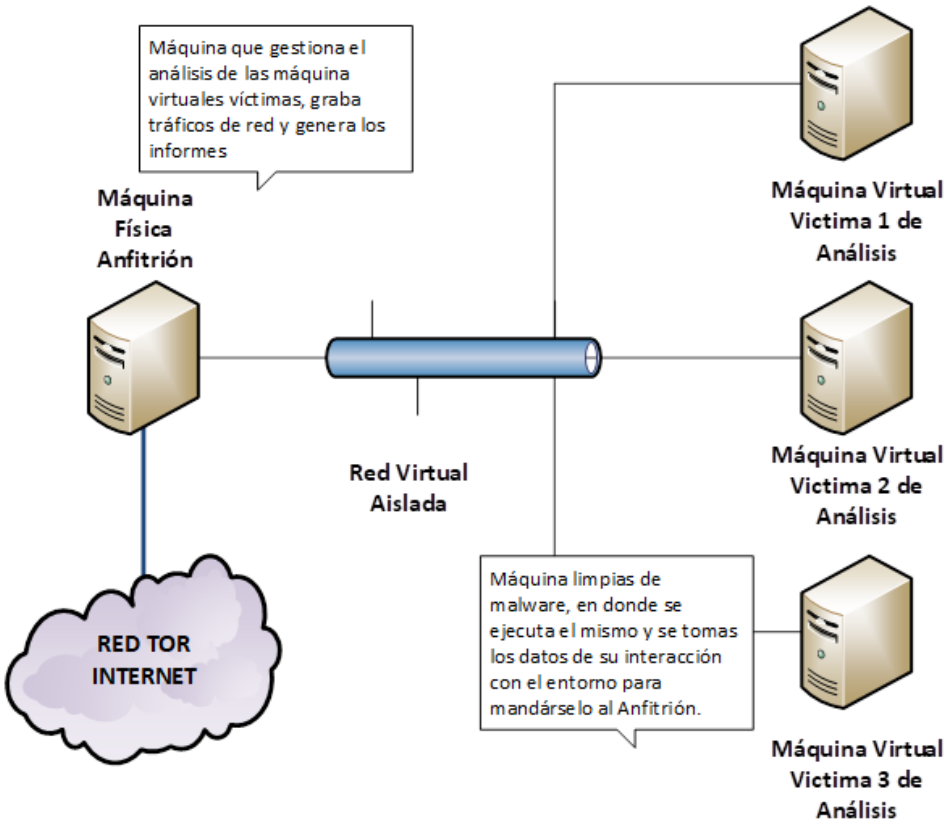

Figura 2.6: Arquitectura de Cuckoo [3].

- Documentos de Microsoft Office
- URLs y archivos HTML
- scripts de PHP
- Archivos CPL
- scripts de Visual Basic (VB)
- Archivos ZIP
- Java JAR
- Python

Se puede analizar estos ficheros sobre los siguientes sistemas operativos siendo estos el sistema de la máquina cliente:

- Windows,
- OS X,
- Linux,
- Android
- Rastrear llamadas a la API y el comportamiento general del archivo.
- Volcado y análisis del tráfico de red, incluso cuando están codificados.
- Realizar el análisis avanzado de memoria del sistema virtualizado infectado.
- Personalización tanto el procesamiento y las etapas de presentación de informes.

Capítulo 3

Objetivos y Metodología de Trabajo

En este capítulo se describe información detalladamente sobre la metodología propuesta. La Sección 3.1 presenta el objetivo principal de este trabajo, así como los objetivos secundarios. La Sección 3.2 presenta en alto nivel los pasos que se seguirán para llegar hasta el objetivo general, detallando los pasos y del entorno de trabajo.

3.1 Objetivos

Como objetivo principal se pretende evaluar la exactitud que presenta un enfoque de clasificación de malware utilizando las firmas de Cuckoo con comportamientos específicos de familias, los cuales pueden utilizar las características estáticas y dinámicas obtenidas del análisis de las muestras.

Como objetivos secundarios(OS) se plantean os siguientes:

- **OS1.** Realizar un estudio de la literatura actual sobre la ejecución, detección y clasificación de malware.
- **OS2.** Diseñar e Implementar una plataforma de para la detección y clasificación de malware utlizando Cuckoo SandBox y firmas de familias de malware.
- **OS3.** Evaluar y analizar los resultados con un conjunto de referencia.

3.2 Metodología del trabajo

La metodología que se seguirá para la realización de este trabajo consta de tres fases generales:

1. Estudio del estado del arte.
2. Diseño e implementación de la plataforma.
3. Evaluación de la clasificación.

En la primera fase de este trabajo realizamos una revisión sistemática del estado del arte de las técnicas de clasificación y detección de malware. En particular en esta fase estudiamos las técnicas actuales de detección, los tipos de análisis que se realiza del software malicioso, las categorías o tipos de malware que podemos encontrar actualmente, las técnicas de clasificación y los entornos de análisis de malware más utilizado actualmente, tanto de software libre como privativo.

En la segunda fase diseñamos e implementamos una plataforma para clasificar el malware utilizando Cuckoo SandBox. Describimos en detalle los componentes utilizados, así como los pasos necesarios para implementar toda la plataforma. En esta fase también crearemos firmas de malware para detectar y clasificar varias familias de malware, las cuales son puestas a prueba analizando dos datasets que comprenden 16.156 muestras de software malicioso.

En la ultima fase de este trabajo, para evaluar si los resultados obtenidos con nuestra plataforma son adecuados, primero utilizamos una herramienta externa AVClass [4], la cual es ampliamente usada en la literatura para realizar el etiquetado de muestras de software malicioso. Con esta herramienta realizamos una clasificación previa, que nos servirá como el conjunto de referencia para evaluar nuestros resultados utilizando las métricas *Precision*, *Recall* y *F-measure*. Finalmente analizamos los resultados de esta evaluación y concluimos remarcando las contribuciones de este trabajo.

Capítulo 4

Diseño e Implementación

Esta capítulo describe la arquitectura de la solución implementada para alcanzar los objetivos de esta trabajo.

4.1 Componentes de la arquitectura

Para realizar todas las actividades experimentales de este trabajo y los posteriores análisis de resultados, se utilizó la arquitectura mostrada en la figura 4.1. Primero se recolecto dos conjuntos de datos de malware de VirusShare, luego analizamos estos archivos en Cuckoo, el cual genera un reporte que es almacenado en una base de datos de MongoDB, en estos reportes se encuentra ya el resutado de la clasificación de nuestras firmas de Cuckoo. Esta clasificación del malware en familias es analizada tomando como referencia la clasificación realizada con la herramienta externa AVClass.

4.1.1 Entorno de Trabajo

Cuckoo se implementó sobre un servidor con las características especificadas en la tabla 4.1, en dicho servidor también se realizaron la mayoría de experimentos, el desarrollo y las pruebas.

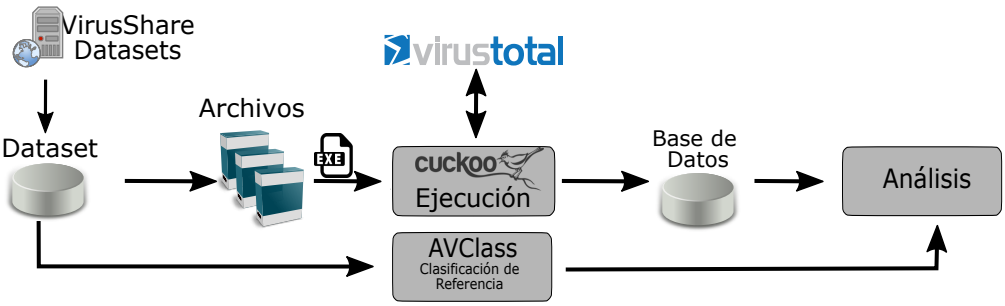

Figura 4.1: Arquitectura del piloto experimental.

Tabla 4.1: Especificación del servidor

Características del servidor	**Especificación**
Modelo del equipo	Dell PowerEdge R730
Procesador	Intel Xenon 3,5 GHz
Memoria RAM	64 GB
Sistema Operativo	Ubuntu 14.04.3 LTS
Almacenamiento	8TB

4.2 Cuckoo

Esta sección describe el proceso de implementación de Cuckoo 2.0.5. El proceso de instalación de Cuckoo se presenta detalladamente en la documentación de la herramienta [3], a pesar de esto su puesta en marcha y configuración suele presentar múltiples problemas, por esta razón en la Sección 4.2.1 se describe los pasos necesarios para poner en marcha Cuckoo, después de detalla la configuración utilizada en la Sección 4.2.2 y en la Sección 4.2.3 se describe la implementación de los entornos de análisis, presentando la planificación, la creación y configuración de estos entornos.

4.2.1 Instalación de Cuckoo en el Host

Cuckoo esta escrito en el lenguaje de programación Python y utiliza varias librerías de Python. El primer paso de la instalación de Cuckoo en el host es verificar que estas

librerías están instaladas y actualizadas en la maquina que será el host, en nuestro caso es el servidor con el sistema operativo Ubuntu 16.04. A continuación se presentan los comandos necesarios para instalar Cuckoo con una breve descripción de cada uno.

4.2.1.1 Instalación de requerimientos

Cuckoo requiere múltiples librerías y estas a su vez requieren de otras. Para facilitar la instalación de estos requerimientos Ubuntu cuenta con el gestor de paquetes aptitude (i.e., *apt-get*). Con esto se controlará que no se instalen paquetes que ya están en el sistema. La estructura de los pasos a realizar se presenta de la siguiente forma: en cuadro se muestra los comandos que se ejecutan con una breve explicación de cada comando.

Para evitar problemas futuros es importante iniciar cerciorándose de que el sistema esta actualizado, para esto abriremos una terminal en donde realizaremos esta comprobación y ejecutaremos todos los demás comandos.

```
$ sudo apt-get update
```

Esto iterará sobre las listas de fuentes de las librerías, actualizando estas listas con las últimas versiones de las librerías disponibles.

```
$ sudo apt-get upgrade
```

Esto actualizará todas las librerías que actualmente están instaladas en el sistema.

```
$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev python-setuptools
```

- **python** Instala el interprete del lenguaje Python 2.7, el cual se utiliza para ejecutar todos los scripts de Python.
- **python-pip** Herramienta para instalar otros librerías de Python.
- **python-dev** Contiene las cabeceras de los archivos para construir extensiones de python.
- **libffi-dev** Librería que sirve como interfaz entre código interpretado y compilado *(Foreign Function Interface)*.

- **libssl-dev** Contiene las librerías de desarrollo, cabeceras de archivos, las páginas de ayuda *(man)* para las librerías *ssl* y *libcrypto*.
- **python-setuptools** Herramienta de ayuda para gestionar las librerías de Python, útil para descargar, compilar, instalar, actualizar y desinstalar estas librerías.

```
$ sudo apt-get install libjpeg-dev zlib1g-dev swig
```

- **libjpeg-dev** Archivos de desarrollo para trabajar con imágenes de la libreria *JPEG*.
- **zlib1g-dev** Archivos de desarrollo para la librería de compresión *zlib*.
- **swig** Librería para conectar programas escritos en *C* y *C++* con lenguajes de scripts.

```
$ sudo apt-get install mysql-server
```

- **mysql-server** Es una base de datos relacional que servirá para almacenar la información referente a cada análisis realizado en Cuckoo. Esta base de datos la gestionará directamente Cuckoo, almacenando información sobre las máquinas virtuales y sus estados; información de cada archivo analizado (e.g., tamaño, md5, sha1, tipo de archivo, etc); información sobre cada análisis (e.g., fecha de envío del archivo, fecha del análisis, si ya se genero el reporte, máquina virtual utilizada, etc.). Esta base de datos es muy útil cuando se realiza el análisis de un conjunto de muestras grande, pues esta permite llevar un control detallado de los análisis realizados.

```
$ sudo apt-get install mongodb
```

- **mongoDB** Es una base de datos no-sql que servirá para almacenar los reportes de las ejecuciones de los archivos analizados y también será el backend para la interfaz web de Cuckoo.

```
$ sudo apt-get install tcpdump apparmor-utils
```

- **tcpdump** Herramienta para capturar el tráfico de red.
- **apparmor-utils** Módulo de seguridad de Linux para otorgar permisos especiales a programas específicos.

```
$ sudo aa-disable /usr/sbin/tcpdump
```

- Esto permite que *tcpdump* pueda escribir los archivos PCAP que contienen la información de las conexiones de red realizadas durante un análisis en el directorio de almacenamiento de Cuckoo.

```
$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

- Permite que *tcpdump* se ejecute como usuario *root* sin que el usuario que ejecuta Cuckoo sea *root*.

El siguiente paso será instalar Cuckoo, en las versiones anteriores a la 2.04 la instalación se realizaba de una forma menos automática, donde el analista tenia que clonar el repositorio de Cuckoo e instalar todo por su cuenta, lo cual conllevaba muchos problemas para los usuarios. En la versión actual Cuckoo se incluye como un paquete de Python haciendo que todo el proceso de instalación se simplifique en gran medida.

```
$ pip install -U cuckoo
```

4.2.2 Configuración del host

Una vez que Cuckoo se ha instalado, se debe realizar la configuración del Host para que este pueda interactuar con las máquinas virtuales, analizar los archivos y generar los reportes de acuerdo a las preferencias del analista. El primer paso de esta configuración es crear un usuario llamado *cuckoo* con el que ejecutaremos Cuckoo como se muestra a continuación.

```
$ sudo adduser cuckoo
```

En la sección 4.2.3.2 se muestra la instalación de VirtualBox, durante esa instalación se creará el grupo de usuarios de *vboxusers*, este grupo se utilizara en la configuración del host, para asegurarnos que el usuario *cuckoo* pertenece al grupo de usuarios de VirtualBox, para que Cuckoo pueda utilizar la interfaz de linea de comandos de VirtualBox *(VBoxManage)* esto lo realizamos con el siguiente comando.

```
$ sudo usermod -a -G vboxusers cuckoo
```

El siguiente paso de esta configuración es ejecutar Cuckoo por primera vez para que este creé el directorio de trabajo *CWD (Cuckoo Working Directory)* en la ubicación por defecto */home/cuckoo/.cuckoo*.

```
$ cuckoo -d
```

Se nos presentará el siguiente mensaje indicando que Cuckoo ha terminado de realizar todas las configuraciones necesarias estableciendo los valores por defecto.

```
$ cuckoo -d
Cuckoo Sandbox 2.0.5
www.cuckoosandbox.org
Copyright (c) 2010-2018

============================================================
Welcome to Cuckoo Sandbox, this appears to be your first run!
We will now set you up with our default configuration.
You will be able to modify the configuration to your likings
by exploring the /home/cuckoo/.cuckoo directory.

Among other configurable things of most interest is the
new location for your Cuckoo configuration:
/home/cuckoo/.cuckoo/conf
============================================================

Cuckoo has finished setting up the default configuration.
Please modify the default settings where required and
start Cuckoo again (by running 'cuckoo' or 'cuckoo -d').
```

Como también se presenta en el mensaje anterior, el siguiente paso es revisar los archivos de configuración de Cuckoo que se encuentran en el directorio */home/cuckoo/.cuckoo/conf*, todas las opciones de configuración vienen por defecto con comen-

tarios que facilitan la personalización de la plataforma, en nuestro trabajo solo modificaremos las opciones que necesitamos para conseguir nuestros objetivos, los archivos de configuración que personalizaremos se describen a continuación.

- **cuckoo.conf** Contiene las opciones de configuración generales que regirán el funcionamiento de nuestra SandBox.
- **processing.conf** Contiene las opciones de configuración de los módulos que procesan la información obtenida en los análisis.
- **reporting.conf** Contiene las opciones de configuración de los reportes que se generarán luego de procesar la información.
- **vbox.conf** Contiene las opciones de configuración de las máquinas virtuales de VirtualBox donde se realizarán los análisis.

4.2.2.1 Configuraciones generales de Cuckoo

El archivo *cuckoo.conf* con las principales configuraciones de cuckoo, nos permitirá gestionar el comportamiento de los análisis, como los tiempos que durará los análisis, el tipo de software de virtualización que utilizaremos, las configuraciones de red utilizaremos entre otros. A continuación se explican las opciones de configuración que se han modificado y se muestra el archivo 4.1 con las opciones que hemos personalizado.

- **cuckoo** Configuraciones generales de Cuckoo.
  - **machinery** Se especifica el nombre del software de virtualización.
  - **terminate_processes** Cuando los tiempos limites de análisis han finalizado, antes de apagar la VM se termina los procesos monitoreados para ver algún comportamiento interesante como el cierre de conexiones establecidas por el proceso.
  - **max_vmstartup_count** Limitar la cantidad de máquinas virtuales que pueden iniciar en paralelo, puesto que una de las tareas mas intensas para

la CPU es iniciar las máquinas, con esta opción se trata de evitar que el procesador trabaje al máximo. Además en base a la experiencia del autor trabajando con Cuckoo, esta opción permite que los análisis que se realizan en paralelo fluyan sin problemas.

  - **tmppath** Se establece el directorio donde se almacenaran los ficheros temporales de Cuckoo, este cambio es debido a que el directorio por defecto */tmp* se limpia cada vez que el sistema se reinicia y en nuestro caso el servidor es compartido por otros usuarios, los cuales podrían reiniciar el sistema al realizar alguna tarea administrativa, por lo que con esta configuración evitamos que se pierdan los archivos temporales de Cuckoo.

- **resultserver**

  - **ip** La dirección IP del adaptador de red que recibirá la información desde las maquinas virtuales de análisis, esta dirección será la que le asignemos al adaptador de red virtual que creamos para VirtualBox en la Sección 4.2.3.2.
  - **upload_max_size** Este valor por defecto se establece en 128MB, hemos incrementado este valor, pues se ha observado que en algunos casos los archivos escritos en el sistema, por el archivo analizado suelen sobrepasar este tamaño, con lo cual se puede perder información del análisis.

- **processing** Configuraciones del modulo de procesamiento que genera los reportes.

  - **analysis_size_limit** Se ha observado que varios reportes no se generan presentando un error que indica que los archivos generados por el análisis, los cuales se usan para generar los reportes sobrepasan el valor por defecto de 128MB.

- **timeouts** Configuraciones que establecen los tiempos máximos que se tomarán en cuenta para cada análisis.

- **default** Se incrementa el tiempo máximo que durará cada análisis en 60 segundos.
- **critical** Se incrementa el tiempo máximo que Cuckoo utilizará para dar un análisis por fallido, esto se debe a que se ha observado que varios análisis necesitan más tiempo para terminar su ejecución normal.
- **vm_state** Se ha observado que en ocasiones VirtualBox tarda mas de un minuto en cambiar el estado de la máquina virtual para que pueda seguir siendo utilizada por Cuckoo [61]. por esta razón se incrementa este valor para evitar tener que reiniciar la instancia de Cuckoo.

Archivo 4.1: Modificaciones en el archivo de configuración general *cuckoo.conf*.

```
[cuckoo]
machinery = VirtualBox
terminate_processes = yes
tmppath = /home/cuckoo/.cuckoo/tmp

[resultserver]
ip = 192.168.56.1
upload_max_size = 536870912

[processing]
analysis_size_limit = 536870912

[timeouts]
default = 120
critical = 90
vm_state = 120
```

4.2.2.2 Configuración de reportes

El archivo *report.conf* contiene las configuraciones que se debe establecer de acuerdo a cada tipo de reporte que se deseé obtener como resultado de los análisis. A continuación se explican las opciones de configuración que se han modificado y se muestra el archivo 4.2 con las opciones que hemos personalizado.

- **jsondump** Opciones de configuración para los reportes en archivo de formato JSON.

  - **disabled** Por defecto el formato principal de los reportes que esta habilitado son archivos de tipo JSON, nuestra implementación no utilizará este tipo de reportes, por lo que se deshabilita esta opción.

- **mongodb** Opciones de configuración para los reportes en MongoDB
  - **enabled** Este trabajo utiliza la base de datos no-sql MongoDB para almacenar los reportes, por lo que se habilita esta opción. Se recomienda establecer configuraciones de seguridad para la conexión con la base de datos, y cambiar el puerto por defecto que se utiliza.

Archivo 4.2: Modificaciones en el archivo de configuración de reportes *reporting.conf*.

```
[jsondump]
enabled = no
indent = 4
calls = yes

[mongodb]
enabled = yes
host = 127.0.0.1
port = 29000
db = cuckoo
store_memdump = no
paginate = 100
```

4.2.2.3 Configuración de VirtualBox

El archivo *VirtualBox.conf* contiene las configuraciones que se debe establecer para que el host interactúe con las máquinas virtuales. A continuación se explican las opciones de configuración que se han modificado y se muestra el archivo 4.3 con las opciones que hemos personalizado.

- **VirtualBox** Opciones de configuración generales de VirtualBox.
  - **mode** Cuando se ejecuta VirtualBox en un servidor que no cuenta con una interfaz gráafica como en este proyecto se debe ejecutar con el modo *headless*, en lugar de el valor por defecto *gui*.

  - **path** Se establece la ruta de la herramienta *VBoxManage* que sirve como la interfaz de VirtualBox para gestionar las máquinas virtuales a través de la linea de comandos.
  - **interface** Establece el nombre de la interfaz de red para todas las máquinas virtuales, si se utiliza una interfaz común para todas las máquinas.
  - **machines** Se lista el nombre que se haya asignado a cada una de las máquinas virtuales que se utilizarán para los análisis, por cada máquina de esta lista se deberá crear una sección de configuración.

- **cuckoo(n)** Configuración especifica de cada máquina virtual, en nuestro caso que se utilizará 10 máquinas virtuales el valor de *n* será desde *n = 1* hasta *n = 10*.
  - **label** Esta variable indica el nombre de la máquina virtual.
  - **platform** Esta variable indica el sistema operativo de la máquina virtual.
  - **ip** Esta variable indica la dirección IP que utilizará cada máquina virtual de análisis, nosotros utilizaremos el rango de IPs desde 192.168.56.101 hasta 192.168.56.110, donde tendremos una IP para cada una de las máquinas virtuales.
  - **snapshot** Esta variable indica el nombre de la Snapshot que se tomará de la máquina virtual en un estado que cuente con todas las configuraciones necesarias para realizar el análisis de algún fichero sospechoso.

Archivo 4.3: Modificaciones en el archivo de configuración de VirtualBox *VirtualBox.conf*.

```
[VirtualBox]
mode = headless
path = /usr/bin/VBoxManage
interface = vboxnet0
machines = cuckoo1, cuckoo2, cuckoo3, cuckoo4, cuckoo5, cuckoo6, cuckoo7, cuckoo8, cuckoo9,
    cuckoo10
```

```
[cuckoo1]
label = cuckoo1
platform = windows
ip = 192.168.56.101
snapshot = Snapshot1

[cuckoo2]
label = cuckoo1
platform = windows
ip = 192.168.56.102
snapshot = Snapshot1

...
...
...

[cuckoo10]
label = cuckoo10
platform = windows
ip = 192.168.56.110
snapshot = Snapshot1
```

4.2.3 Configuración de los entornos de análisis

Actualmente Cuckoo Sandbox puede implementarse utilizando varias soluciones de Software de Virtualización que incluyen VirtualBox [62], Qemu [63], Vmware [64], Vsphere [65], Xenserver [66], Esx [67], Kvm [68]; también permite realizar la implementación utilizando como clientes máquinas físicas mediante un software como fog [69] que permita restaurar las imágenes del sistema operativo a un estado anterior a cada análisis, cuando se utiliza maquinas virtuales como clientes la restauración al estado previo se realiza mediante snapshots. Además Cuckoo esta desarrollado de una forma muy modular lo que permite fácilmente la integración de otras soluciones de virtualización. En este trabajo se utilizo VirtualBox, por ser software libre y presentar una integración intuitiva con Cuckoo.

4.2.3.1 Planificación del entorno

La creación del entorno de análisis es una tarea crítica dentro de la implementación de una Sandbox, esta debe llevarse a cabo con la planificación adecuada, pues las consideraciones que se tome en cuenta a la hora de crear estos entornos afectará a los resultados que se obtengan.

La automatización del análisis de malware es una tarea no determinista, esto quiere decir que el éxito de una ejecución depende de demasiados factores que no se podrán abarcar, ya que a priori no conocemos las intenciones del archivo que se analizará. En este trabajo se han considerado varios puntos para que los análisis de las muestras maliciosas se ejecuten correctamente y se obtengan los mejores resultados posibles, para esto se ha establecido varios lineamientos para los entornos de virtualización que incluyen el sistema operativo que usarán, los niveles de parches de actualización, el software adicional que se instalará, los rastros de uso, técnicas anti-evasión. A continuación se describen con más detalle estos lineamientos.

- **Sistema Operativo** Se utiliza Windows 7 pues al momento de realizar este proyecto el Sistema Operativo Windows sigue siendo el más usado, aunque Windows 10 desde Enero de 2018 ya se usa más que Windows 7, del total de usuarios la diferencia es de un menos de un 10 % [70], además nuestras muestras de malware son ejecutables de Windows.

- **Nivel de parches** Dado que nuestras muestras de malware fueron recolectadas en los meses de Noviembre y Diciembre 2017, nuestro entorno de análisis de Windows se actualizara con los parches hasta Julio del 2017, esto ayudará a que si las muestras de malware tratan de explotar alguna vulnerabilidad reciente puedan hacerlo y así los resultados nos muestren el comportamiento completo de las muestras analizadas.

- **Software adicional** El software malicioso en algunos necesita interactuar con otras aplicaciones para conseguir su objetivo o explotar alguna vulnerabilidad, en

nuestro entorno hemos decidido instalar las siguientes aplicaciones para facilitar la ejecución de las muestras maliciosas:

  - **Frameworks/Lenguajes** .NET, Java, Python.
  - **Utilitarios** Adobe Flash Player, Acrobat Reader, Microsoft Office.
  - **Navegadores Web** Internet Explorer 10, Mozilla Firefox, Google Chrome, Opera.

- **Rastros de Uso** Generalmente el malware se desarrolla para ejecutarse en entornos reales, donde la máquina esta en constante uso, en algunos casos el malware puede detectar que se esta ejecutando en una ambiente nuevo porque no encuentra trazas de que haya sido usado, para evitar esto se utilizará la maquina virtual por una semana como el equipo principal del autor.

- **Técnicas anti-evasión** Como en el punto anterior el malware puede detectar que esta siendo ejecutado en un entorno virtual, en algunos casos incluso pueden detectar por que Sandbox están siendo analizados por lo que aplicamos algunas técnicas de anti-evasión [71, 72].

4.2.3.2 Creación del entorno

Como se mencionó en la Sección 4.2.3.1 para crear las máquinas virtuales (VM) de análsis se utiliza el software de virtualización VirtualBox, la instalación de esta se realiza con el siguiente comando:

```
$ sudo apt-get install VirtualBox-5.2
```

La arquitectura de nuestra plataforma contará con 10 máquinas virtuales para realizar los análisis, para esto crearemos una sola máquina tomando en cuenta las consideraciones de la Sección 4.2.3.1, esta máquina será el modelo que utilizaremos para clonar las 10 maquinas de nuestra plataforma, por facilidad esto lo haremos en un equipo con entorno gráfico. Para esto abriremos el software VirtualBox crearemos una

nueva máquina virtual seleccionando las configuraciones que se describen a continuación y se muestran en la Figura 4.2.

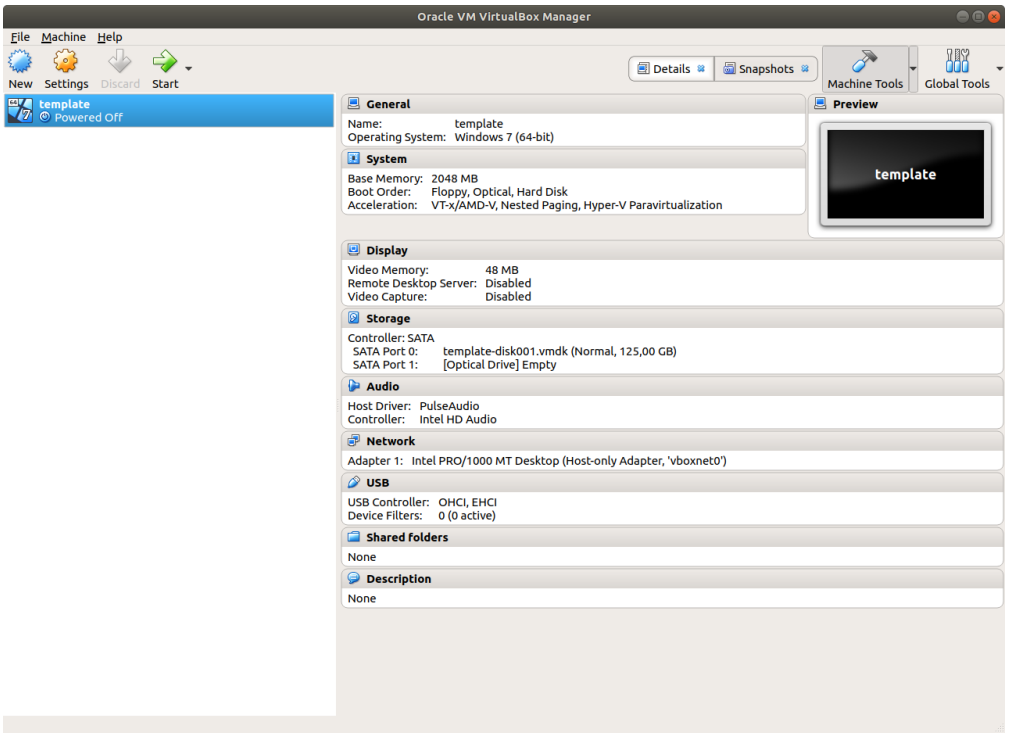

Figura 4.2: Creación de la máquina virtual modelo para el análisis en VirtualBox

- **Configuraciones generales**
  - **Nombre** template.
  - **Tipo** Microsoft Windows.
  - **Versión** Windows 7 64 bits.
  - **Tamaño de memoria** 2GB.
- **Disco virtual**
  - **Tipo de archivo del disco virtual** VDI (VirtualBox Disk Image).
  - **Almacenamiento en el disco duro** Almacenado dinámico. Esto quiere decir que el disco virtual ira creciendo a medida que vaya necesitando más

espacio hasta llegar a el máximo tamaño seleccionado.

- **Versión** Windows 7 64 bits.
- **Tamaño** 256GB, puede parecer que esto es mucho espacio para el disco virtual, pero esto es una de las técnicas anti-evasión a las que nos referíamos en la Sección 4.2.3.1, algunos software maliciosos consultan el tamaño del disco, ya que algunos autores del malware intuyen que a una máquina de análisis no se le asignará más de 50GB lo cual es muy poco y en la práctica irreal para un equipo físico, y como en el punto anterior hemos seleccionado que el almacenamiento sea dinámico, esto no causará problemas de almacenamiento, pues Windows reconocerá que tiene disponibles los 256GB y como los análisis no utilizarán dicho espacio este no se asignará en el disco duro, pero el malware verá que el sistema tiene los 256GB.

- **Configuración de red** Para que las máquinas virtuales puedan acceder a Internet, debemos crear un adaptador virtual de red, esto lo realizaremos con la interfaz de linea de comandos *VBoxManage*, para familiarizarnos con la herramienta. Primero crearemos la adaptador virtual de red ejecutando el el siguiente comando en una terminal de Ubuntu.

```
$ VBoxManage hostonlyif create
```

  - *hostonlyif* Parámetro de *VBoxmanage* para gestionar adaptadores virtuales de red del tipo *Host-only*, los cuales solo permiten conexiones entre el Host y las máquinas virtuales.
  - *create* Parámetro para crear un adaptador virtual de red del tipo *Host-only*. El adaptador creado tendrá por defecto el nombre *vboxnet0*, es posible especificar otro nombre si se desea, en nuestro caso utilizaremos el nombre por defecto.

```
$ VBoxManage hostonlyif ipconfig vboxnet0 --ip 192.168.56.1
```

- *ipconfig* Parámetro de para configurar los adaptadores virtuales de VirtualBox previamente creados, aquí le asignaremos la dirección IP, que utilizaremos para las interacciones entre Cuckoo en la máquina Host y todas las máquinas virtuales de análisis. Esta es la dirección IP que se utiliza en la configuración del Host en la Sección 4.2.2.1.

Una vez que tenemos la máquina virtual creada, el siguiente paso es instalar el Sistema Operativo, para lo cual necesitaremos una imagen de instalación de Windows y la licencia respectiva. La instalación de Windows es un proceso trivial que no se describirá en este trabajo. En cuanto se tenga la máquina virtual con el sistema operativo Windows 7 instalado el siguiente paso será seguir los lineamientos de la Sección 4.2.3.1 como instalar los programas adicionales, crear los rastros de uso y aplicar las técnicas anti-evasión. Luego de esto nuestro maquina estará casi lista, como se muestra en la Figura 4.3. El último paso es realizar algunas configuraciones en la máquina virtual de análisis.

4.2.3.3 Configuración del entorno

Para que podamos utilizar las máquinas virtuales de análisis configuraremos el agente de Cuckoo, la configuración de red y tomaremos una instantánea(Snapshot) del estado de la maquina cuando ya se encuentre lista para trabajar con Cuckoo.

Agente de Cuckoo Cuckoo se comunica con las máquinas virtuales mediante un programa de Python llamado *agent.py*. Este agente abre un puerto en la máquina virtual para que le Host se conecte. El Host enviará los archivos que se analizarán a través de esta conexión, así como la información sobre como debe devolver y analizar los resultados. Este agente también crea una conexión con el *resultserver* configurado en el archivo 4.2.2.1 en la Sección 4.2.2.1, enviando los resultados del análisis a través de esta conexión. El agente de Cuckoo lo encontraremos en el directorio CWD de Cuckoo (*/home/cuckoo/.cuckoo*). Este lo tenemos que copiar dentro de la máqui-

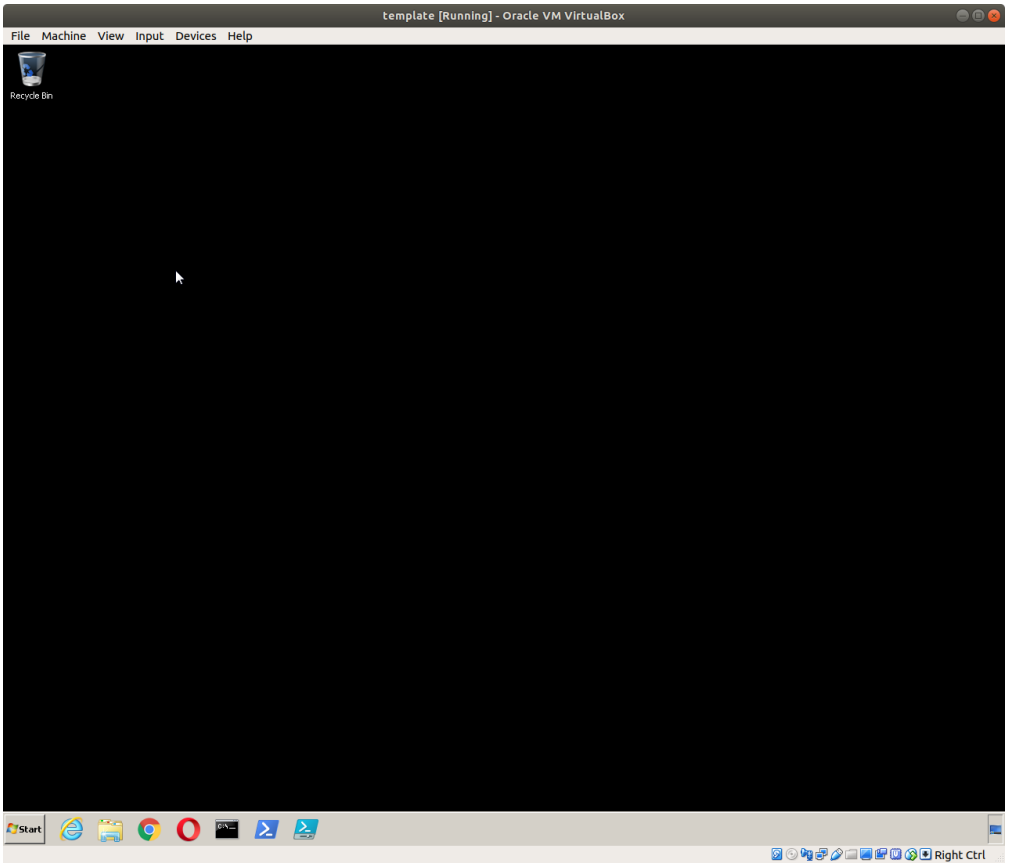

Figura 4.3: Máquina virtual modelo con el sistema operativo instalado.

na virtual con Windows 7 en el directorio *C:\Users\user\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup* , esto se realiza para que el agente se inicie automáticamente cuando inicie el Sistema. Para que la ejecución de el agente no abra una ventana de la terminal de Windows cambiaremos el nombre del agente de *agent.py* a *agent.pyw*, con lo que el agente quedaría en la siguiente ubicación *C:\Users\user\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\agent.pyw*.

Clonación de la máquina modelo En este punto nuestra máquina virtual modelo estará ya con el agente de Cuckoo instalado. Con la máquina apagada procedemos a clonar 10 máquinas iguales para no repetir el proceso de instalación del sistema operativo, las aplicaciones instaladas y configuraciones realizadas en la Sección 4.2.3.2.

Este proceso lo realizaremos con un script de bash que se muestra en el archivo 4.4, creado para automatizar esta tarea aprovechando la interfaz de linea de comandos de VirtualBox *VBoxmanage*.

Archivo 4.4: Script para clonar la máquina virtual de análsis modelo *clone.sh*.

```
#!/bin/bash
for i in {21..30}
do
VBoxManage clonevm cuckoo20 −−mode machine −−name ”cuckoo$i” −−register
echo ”succesfully cloned cuckoo$i”
done
```

Configuración de red Las 10 máquinas virtuales que utilizaremos deben ser configuradas de acuerdo a las configuraciones establecidas en el Host en el archivo 4.3 en la Sección 4.2.2.3. Por ejemplo la máquina *cuckoo1* tendrá la configuración de red que se muestra en la figura 4.4. Lo mismo debe realizarse para cada una de las máquinas.

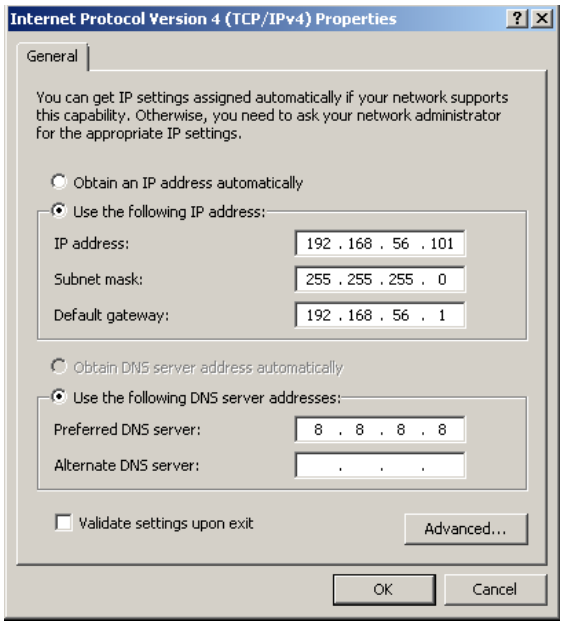

Figura 4.4: Configuración de red una máquina virtual de análisis.

Instantánea de la máquina virtual En este punto tendremos 10 máquinas virtuales configuradas, para que luego de cada análisis realizado, estas vuelvan al estado previo al análisis, tendremos que tomar una instantánea de cada máquina encendida y con el agente de Cuckoo ejecutándose. Esta instantánea la nombramos *Snapshot1* en todas las máquinas, puesto que esto también debe coincidir con la información de las configuraciones establecidas en el Host en el archivo 4.3 en la Sección 4.2.2.3.

4.3 Estructura de datos de los Reportes y las Características

Los reportes de los análisis de Cuckoo se pueden presentar en varios formatos, que incluyen archivos de tipo JSON, HTML, PDF y directamente como documentos en la base de datos MongoDB. En este trabajo hemos dispuesto que los reportes se generen en MongoDB. En la figura 4.5 se muestra la estructura de la información que tiene el documento de reporte de un análisis en MongoDB, cabe mencionar que la estructura de los documentos de MongoDB es la misma estructura de un documento JSON.

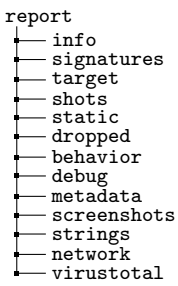

Figura 4.5: Estructura básica de un reporte de Cuckoo. Cada nivel corresponde a un módulo de procesamiento de Cuckoo.

La idea de crear el reporte como un documento en MongoDB, se debe a lo no determinista que es el análisis de malware dinámico, pues no se sabe que comportamientos tendremos antes de haber analizado una muestra, y los comportamientos variaran de una muestra a otra, y este tipo de documentos nos dan la libertad de que cada documento tenga una estructura distinta. En la figura 4.5 se muestra la estructura base de un reporte, los niveles de la figura corresponden a un módulo especifico de Cuckoo los

cuales pueden o no tener más subniveles dependiendo de cada muestra analizada. A continuación se describe brevemente el contenido de cada nivel de la estructura base del reporte de Cuckoo.

- **info** Contiene información relacionada con el análisis de la muestra, por ejemplo el tiempo que duro la ejecución, los parámetros y la fecha de análisis entre otros. Esta información es útil cuando se analiza grandes cantidades de muestras, para llevar un control de todas las muestras analizadas.

- **signatures** Contiene un listado de todas las firmas que han mostrados resultados positivos. Dado que este trabajo utiliza estas firmas para clasificar las muestras, estas firmas se detallaran en la Sección 4.4.

- **target** Contiene información sobre el archivo analizado. Esto incluye algunos tipos de hashes (e.g., sha1, sha256, md5, etc.), el tamaño, nombre y tipo de archivo.

- **shots**. Contiene las capturas de pantalla que toma el sistema cuando detecta cambios en la interfaz gráfica del sistema durante el análisis. Esto puede ser útil para realizar clasificación basada en imágenes. En este trabajo no utilizaremos estas imágenes.

- **static** Contiene información del análisis estático que Cuckoo realiza de la muestra. Esta información depende del tipo de archivo que se este analizando. En este trabajo el las muetras analizadas corresponden a ficheros ejecutables de Windows(Portable Ejecutable PE) [27, 73], por lo que la información corresponde al análisis de un archivo PE, la cual incluye las librerías importadas, imphash [40], marcas de tiempo, información de la versión del ejecutable (e.g., *LegalCopyRight, FileVersion, CompanyName, ProductName, ProductVersion*), e información de las secciones del archivo PE. InternalName,)

- **dropped** Información de los archivos que generados por la ejecución de la muestra. La información de cada archivo es la misma que se presenta del archivo

analizado en el item anterior *target*.

- **behavior** Contiene la información del comportamiento del archivo analizado, esta información se detalla en la Secció 4.3.1.
- **network** Contiene la información de las conexiones de red que ha generado el archivo analizado, esto se realiza mediante el procesamiento del archivo PCAP generado por el programa de linux tcpdump instalado y configurado en la sección 4.2.1.1.
- **virustotal** Contiene la información del reporte obtenido de VirusTotal [50], cuando se genera el reporte de Cuckoo este solicita el reporte a VirusTotal, y en caso de que el archivo no se haya analizado anteriormente, este envía el archivo para su análisis. Este reporte de VirusTotal contiene los AVlabels de cada uno de los sistemas antivirus con los que VirusTotal los haya analizado, durante nuestro análisis los reportes muestran una media de 57 sistemas antivirus utilizado para analizar nuestras muestras.

Una vez presentada la estructura baśica de un reporte de Cuckoo, es importante tener en cuenta que la información que se genera por cada análisis esta muy detallada, cada muestra contiene información distinta y cada elemento de cada uno de los niveles y subniveles de esta estructura pueden ser vistos como potenciales características para clasificar conjuntos de muestras analizadas. Por lo que potencialmente se podría plantear varios modelos de clasificación, de acuerdo a diferentes enfoques, ya que hasta donde conocemos, de la literatura estudiada no se ha presentado un enfoque que utilice todas las características obtenidas del análisis estático y dinámico para el clustering o clasificación de malware. Estos modelos generalmente dependen de un subconjunto de características.

Este trabajo pretende proponer un piloto experimental de clasificación de malware, utilizando la información de los reportes de los módulos (niveles en la figura 4.5) de Comportamiento(Behavior) y Red(network) de Cuckoo. Para esto se utilizará el módulo de firmas(signatures), el cual se detalla en la Sección 4.4.

4.3.1 Características del Comportamiento

El módulo *behavior* de Cuckoo contiene la información de cada uno de los procesos que se han ejecutado a partir de la muestra analizada. En la figura 4.6 se presenta la estructura de la información contenida en el reporte de este módulo y a continuación se describe brevemente cada uno de los subniveles del reporte de este módulo.

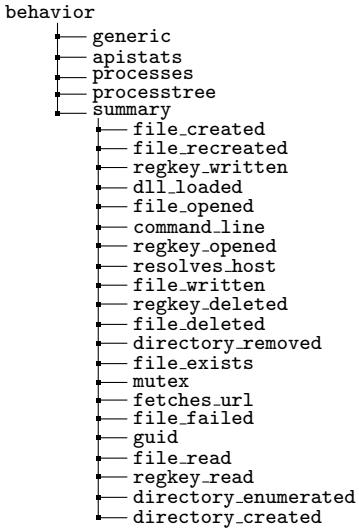

Figura 4.6: Estructura del reporte del módulo *behavior* de Cuckoo.

- **generic** Contiene el listado de los procesos que se han generado a partir del archivo analizado. Estos son los procesos que se monitorearon durante el análisis. Por cada proceso se detalla las características del comportamiento que se presentan en el nivel *summary*.
- **apistats** Contiene el listado de las llamadas a las funciones del API realizadas por cada proceso monitoreado.
- **processes** Por cada proceso se presenta información detallada de los módulos que fueron utilizados y de cada llamada a las funciones del API.
- **processtree** Contiene el árbol de procesos generado a partir del archivo analizado

- **summary** Contiene un resumen de las características del comportamiento, que son las interacciones que tiene con el sistema operativo que provocan un impacto sobre archivos o claves del registro de Windows. Este listado es general, no se detalla por proceso como en el caso de *generic*. Estas características serán la principal fuente de información para nuestro trabajo.
  - **file_created** Contiene el listado de todos los nuevos archivos creados.
  - **file_recreated** Contiene el listado de todos los archivos que se volvieron a crear.
  - **regkey_written** Contiene el listado de todas las claves añadidas en el registro de Windows.
  - **dll_loaded** Contiene el listado de todas las librerias (dlls) que fueron llamadas.
  - **file_opened** Contiene el listado de todos los archivos que se abrieron.
  - **command_line** Contiene el listado de todos los comandos ejecutados en la linea de comandos de Windows.
  - **regkey_opened** Contiene el listado de todas las claves del registro de Windows que se abrieron.
  - **resolves_host** Contiene el listado de nombres de los hosts resueltos.
  - **file_written** Contiene el listado de todos los archivos modificados.
  - **regkey_deleted** Contiene el listado de todas las claves del registro de Windows que se eliminaron.
  - **file_deleted** Contiene el listado de todos los archivos eliminados.
  - **directory_removed** Contiene el listado de todos los directorios que se eliminaron.
  - **file_exists** Contiene el listado de todos los archivos que fueron listados.
  - **mutex** Contiene el listado de todos los mutex creados.

- **fetches_url** Contiene el listado de todas las URLs que fueron buscadas.
- **file_failed** Contiene el listado de todos los archivos que fallaron al intentar ser abiertos, escritos, listados o creados.
- **file_read** Contiene el listado de todos los archivos que fueron leídos.
- **regkey_read** Contiene el listado de todas las claves del registro de Windows que se leyeron.
- **directory_enumerated** Contiene el listado de todos los directorios que se listaron.
- **directory_created** Contiene el listado de todos los nuevos directorios creados.

4.3.2 Características de red

El módulo *network* de Cuckoo contiene la información de todas las conexiones de red que se han ejecutado a partir de la muestra analizada. Para obtener esta información Cuckoo analiza el archivo PCAP. En la figura 4.7 se presenta la estructura de la información contenida en el reporte del módulo que analiza las conexiones de red. Para cada tipo de conexión se presenta la marca de tiempo, la ip y el puerto de origen y destino. Esta información ayuda en muchos casos a identificar cuando una muestra realiza algún tipo de conexión con servidores conocidos por realizar actividades maliciosas.

```
network
├── tls
├── udp
├── dns_servers
├── http
├── icmp
├── smpt
├── tcp
├── smpt_ex
├── mitm
├── hosts
├── dns
├── http_ex
├── domains
├── dead_hosts
├── irc
└── https_ex
```

Figura 4.7: Estructura del reporte del módulo *network* de Cuckoo.

4.4 Firmas de Cuckoo

Cuckoo permite crear firmas personalizadas que se pueden ejecutar sobre los resultados de los análisis, para identificar algún patrón predefinido que pueda representar un comportamiento malicioso particular o un indicador que sea interesante [3]. Estas firmas son útiles para simplificar la interpretación de los análisis, obtener directamente datos procesados de alguna muestra de malware de interés.

Para la creación de firmas utilizamos el modulo *signatures* de Cuckoo, este módulo se ejecuta después del análisis de la muestra, cuando se procesa toda la información de las ejecuciones. Para esto Cuckoo cuenta con un diccionario de Python que contiene los resultados de los módulos procesados, la estructura de este diccionario es la misma que la estructura de los reportes descritos en la sección 4.3. La comprensión de esta estructura nos facilitará el desarrollo de las firmas.

En este trabajo utilizamos las firmas de Cuckoo para clasificar muestras de malware de familias especificas, aislando comportamientos únicos realizados por estas familias de malware.

4.4.1 Creación de firmas

Cuckoo esta diseñado de una forma muy modular y escalable, haciendo que el proceso de creación de firmas sea simple, para esto se requiere un conocimiento del lenguaje Python, y conocer la estructura del diccionario que contiene los resultados de los análisis. Todas las firmas que se crean deben ubicarse en el CWD de Cuckoo, en nuestro caso es en el siguiente directorio: */home/cuckoo/.cuckoo/signatures/windows/*.

En el Archivo 4.5 se presenta el código de un básico ejemplo de una firma de Cuckoo. Esta firma básica de ejemplo tiene como propósito detectar cuando una muestra analizada crea un archivo en el directorio *system32* de Windows. A continuación vamos a recorrer este archivo para detallar brevemente la estructura básica de esta firma.

Archivo 4.5: Ejemplo básico de una firma de Cukoo

```
1  from cuckoo.common.abstracts import Signature
2
3  class NuevoMalware(Signature):
4    name = "nuevo_malware"
5    description = "Crea un archivo en el directrorio system32"
6    severity = 2
7    categories = ["troyano"]
8    families = ["nueva_familia"]
9    authors = ["Richard Rivera"]
10   minimum = "2.0"
11
12   def on_complete(self):
13     summary = self.get_summary()
14       if "file_created" in summary:
15         for f in summary["file_created"]:
16           if f.startswith("c:\\windows\\system32\\"):
17             self.mark_ioc(category = "archivos", ioc = f,description = "archivo en system32")
18     return self.has_marks()
```

1. Importamos la clase base *Signature*, pues esta contiene todos todas las funciones necesarias para que creemos firmas fácilmente.

3. Creamos una nueva clase para esta firma.

4. **name** Asignamos un nombre a esta firma.

5. **description** Escribimos una breve descripción de lo que hace la firma.

6. **severity** Esta es una característica experimental de Cuckoo para asignarle una severidad a cada firma, este puede ser un valor entre 1 y 3, luego suma los valores de todas las firmas que presenten resultados posistivos para darle un valor global de severidad maliciosa a la muestra.

7. **categories** Se define una lista con las posibles categorías de malware que esta firma detectará (e.g., troyano, virus, ransomware, etc).

8. **families** El nombre de la familia a la que corresponden los malware que tienen el comportamiento especial que tratamos de identificar con esta firma.

9. **authors** El nombre del autor de la firma.

10. **minimum** la versión mínima de Cuckoo con la que funcionará esta firma.

12. La función *on_complete* se ejecuta una vez que Cuckoo ha procesado todos los módulos y el diccionario con todos los resultados para el reporte está completo. También se puede utilizar la función *on_call*, esta es llamada cuando Cuckoo esta procesando todas las llamadas al API de Windows, usualmente se usa para detectar parámetros y funciones específicos.

13. Para obtener información de los archivos creados debemos comprobar en el diccionario con la información para el reporte, de acuerdo a la estructura revisada en la sección 4.3.1 *report/behavior/summary/file_created*. Para esto Cuckoo nos facilita una gran variedad de funciones que directamente nos traen la información necesaria, en este caso la función que utilizaremos es *get_summary*.

14. Comprobamos que dentro de la estructura *summary* se encuentra la clave *file_created*.

15. Revisaremos todos los archivos que ha creado la muestra analizada, y si la ruta de alguna de estos archivos inicia con la ruta del directorio *system32*, nuestra firma habrá detectado una muestra que cumple con el comportamiento de esta familia de malware de ejemplo.

16. Completamos información que será añadida en el reporte de las firmas mediante las marcas de firmas(*mark_ioc*):

    - **category** La categoría de esta identificación, en este estamos identificando archivos.
    - **ioc** Pasamos la característica detectada para también tenerla en el reporte, en este caso será la ruta completa del archivo identificado.
    - **description** Una breve descripción de la identificación, esto debe ser referente a la identificación, pues una misma firma puede identificar varias cosas.

17. Retornamos el contenedor con las marcas, el cual es usado por Cuckoo para procesar todas las firmas y añadir esta información en el reporte.

Toda la información de las firmas detectadas se añade en el reporte general como se presenta en la figura 4.5. En la figura 4.8 se presenta el detalle de la estructura *Signatures*. Esta contiene una lista de las firmas detectadas con la información descrita en la firma de ejemplo.

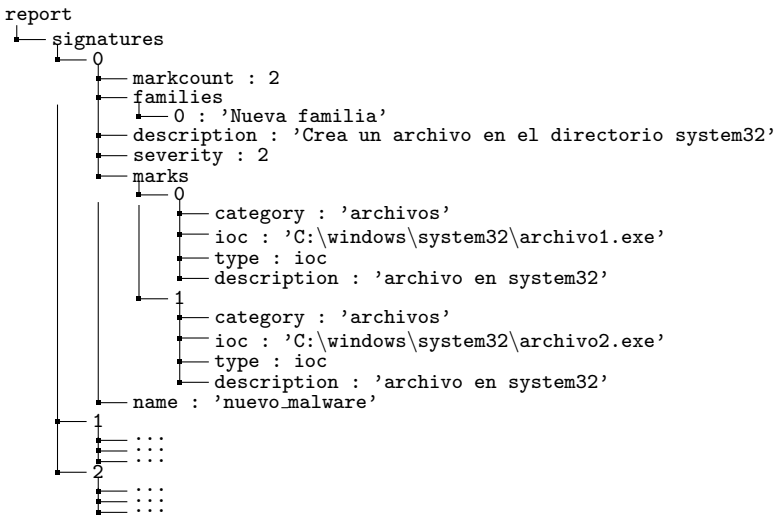

Figura 4.8: Estructura del reporte del módulo *signatures* de Cuckoo.

4.4.2 Firmas desarrolladas

En este trabajo hemos desarrollado 16 firmas para las familias con los clusters más grandes de nuestros conjuntos de muestras, tomando como base la clasificación que se presenta en la sección 5.2. En dicha clasificación se presentan las 10 familias más grandes para cada conjunto de datos, 4 de estas familias aparecen en los dos conjuntos de datos, por esta razón se ha decidido realizar 16 firmas.

El propósito de estas firmas es identificar muestras de una familia de acuerdo a un comportamiento especifico, para esto hemos revisado noticias y reportes de análisis en linea, y hemos identificado comportamientos específicos de estas 16 familias. A

continuación se presentan 2 ejemplos de firmas desarrolladas, para las siguientes 2 familias *installmoster y fairet*.

Todas las firmas que se desarrollan en este trabajo serán publicadas y liberadas, en el sitio web del proyecto en GitHub [10], para que puedan ser usadas por toda la comunidad.

4.4.2.1 Firma para la familia de malware *installmonster*

La familia *Fairet* es una familia de software malicioso de la categoría de programas potencialmente no deseados (PUP) conocida desde 2015 [74]. Esta familia instala software malicioso a la vez que instala otro software que el usuario quería descargar, sin tener el adecuado o explicito consentimiento del usuario, pero con su aprobación, es decir el usuario acepta instalar el software sin ser consciente de que es lo que esta aceptando. Este comportamiento es típico de los PUP.

Para crear la firma que se presenta en el archivo 4.6, hemos analizado la información proporcionada por varios análisis en linea [74–76]. Estos análisis presenta varios comportamientos específicos de esta familia de malware. Entre los comportamientos comunes tenemos dos. El primero es que crea la siguiente clave en el registro de Windows *HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\C379EAD1-CB34-4B09-AF6B-7E587F8BCD80*. El segundo es que crea el siguiente mutex *MutexNPA_UnitVersioning_3184*. Por lo tanto nuestra firma buscara en los comportamientos de registros escritos (*regkey_written*) y en la de mutex creados (*mutex*).

El código de esta firma se presenta en el archivo 4.6.

Archivo 4.6: Firma para la familia de malware installmonster.

```
1 from cuckoo.common.abstracts import Signature
2 class InstallMonster(Signature):
3     name = "installmonster"
4     description = "Signature for installmonster malware family"
5     severity = 2
6     categories = ["pup", "adware"]
7     families = ["installmonster"]
8     authors = ["Richard Rivera"]
9     minimum = "2.0"
```

```
10
11 def on_complete(self):
12   summary = self.get_summary()
13   if "file_created" in summary:
14     for r in summary["regkey_written"]:
15       if r.startswith("HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{C379EAD1-CB34-4
             B09-AF6B-7E587F8BCD80}"):
16         self.mark_ioc(category = "registro", ioc = r,description = "crea un registro con un GUID
               especifico")
17
18   if "mutex" in summary:
19     for m in summary["mutex"]:
20       if m == "MutexNPA_UnitVersioning_3184":
21         self.mark_ioc(category = "mutex", ioc = m,description = "crea un mutex")
22
23   return self.has_marks()
```

4.4.2.2 Firma para la familia de malware *fairet*

La familia *Fairet* es una familia de malware conocida desde 2011 [77]. Este malware que trata de robar información de los usuarios, usualmente es usado para descargar otros malware. Las nuevas variantes de este malware suelen usar una combinación de exploits de PDFs para ejecutar scripts de PowerShell para realizar sus rutinas maliciosas.

También suele usar ataques de fuerza bruta para acceder a archivos compartidos que estan protegidos por contraseña. Tiene un componente spyware que también intenta robar información de cuentas de usuario en ciertos clientes FTP instalados o en programas gestores de archivos. Además a mostrado la capacidad de robar credenciales de correo electrónico de ciertos navegadores web.

Para crear la firma que se presenta en el archivo 4.7, hemos analizado la información proporcionada por varios análisis en linea [77–79]. Estos análisis presenta varios comportamientos específicos de esta familia de malware. Pero hemos visto que todos coinciden en que esta familia crea la siguiente clave en el registro de Windows *HKEY_CURRENT_USER\Software\WinRAR\HWID*, y el valor de esta clave es *GUID*. Por lo tanto nuestra firma buscara en los comportamientos de registros escritos (*reg-*

key_written) la clave de registro antes mencionada.

Archivo 4.7: Firma para la familia de malware fairet.

```
1  from cuckoo.common.abstracts import Signature
2
3  class Fairet(Signature):
4    name = "fairet"
5    description = "Signature for fairet malware family"
6    severity = 2
7    categories = ["spyware", "trojan"]
8    families = ["fairet"]
9    authors = ["Richard Rivera"]
10   minimum = "2.0"
11
12 def on_complete(self):
13   summary = self.get_summary()
14   if "file_created" in summary:
15     for r in summary["regkey_written"]:
16       if r.startswith("HKEY_CURRENT_USER\\WinRAR\\HWID"):
17         self.mark_ioc(category = "registro", ioc = r,description = "crea un registro en WHID")
18   return self.has_marks()
```

Capítulo 5

Evaluación

En este capítulo se realiza la evaluación de la clasificación realizada mediante las firmas que hemos creado en la sección 4.4. Para esto en la sección 5.1 iniciamos con la descripción de la metodología de evaluación utilizada. Después en la sección 5.2 realizaremos una clasificación previa de las muestras con una herramienta externa que servirá como la clasificación de referencia para realizar el análisis. Después en la sección 5.3 se presentan las métricas con las que analizamos los resultados. En la sección 5.4 se presentan los resultados de la clasificación utilizando las firmas que desarrollamos. Finalmente, en la sección 5.5 se presenta el análisis de los resultados utilizando las métricas mencionadas previamente.

5.1 Metodología de Evaluación

En esta sección se describen los pasos seguidos para evaluar y analizar los resultados. Una vez que se han ejecutado todas las muestras de nuestros datasets. Para evaluar y analizar realizamos los siguientes pasos.

1. Primero clasificaremos las muestras utilizando la herramienta externa AVClass, la cual nos entregará los clusters de familias. Esta será nuestra clasificación de referencia.

2. A continuación obtenemos los resultados de los reportes de Cuckoo los cuales estan almacenados en nuestra base de datos de Mongodb.

3. Mediante un script de python obtenemos nuestros resultados de clasificación y detección creando los clusters de cada familia.

4. Analizamos nuestros resultados comparando con los de la clasificación de referencia.

5. Evaluamos los resultados con las métricas *Precision*, *Recall* y *F-Measure*.

5.2 Muestras de malware

Esta sección describe el conjunto de muestras de malware utilizados en la clasificación de malware de este trabajo. Se utilizan dos conjuntos de datos obtenidos desde VirusShare [80], descritos en la Tabla 5.1. Los dos conjuntos de muestras fueron recolectados en los meses de Febrero de 2018.

Tabla 5.1: Descripción de los conjuntos de muestras analizados.

Dataset	**Fecha**	**Muestras**
VirusShare_308	02/2018	7.722
VirusShare_309	02/2018	8.435

5.2.1 Clasificación previa de las muestras

Para probar nuestro modelo de clasificación necesitaremos evaluarlo contra una clasificación de referencia. Para esto hemos utilizado la herramienta AVClass [4] descrita en la sección 2.1.4, esta herramienta utiliza los AVLabels de las muestras obtenidos de los reportes de VirusTotal.

El resultado de esta clasificación se presenta en la tabla 5.2. Donde para cada conjunto de muestras se presenta el número total de muestras; el número de muestras que

no tenían AVlabels, lo utilizamoscual puede deberse a que las muestras son benignas o que ningún antivirus las ha detectado como maliciosas; el número total de familias en que fueron agrupados y el numero total de muestras *singleton*, las cuales son las muestras que no fueron asignadas a ninguna familia.

Tabla 5.2: Clasificación de los conjuntos de muestra con la herramienta AVClass [4].

Dataset	**Muestras**	**Sin AVlabel**	**Familias**	**Singleton**
VirusShare_308	7.721	157	235	1.464
VirusShare_309	8.435	591	246	1.592
Total	**16.156**	**748**	**481**	**2.056**

Para comprender mejor la clasificación de las muestras, en la Tabla 5.3 se presenta el top 10 de las familias con los clusters más grandes para cada conjunto de muestras. Como se puede apreciar al menos las 4 primeras familias de los dos conjuntos de datos son las mismas aun que el orden es distinto, en ambos casos las 3 primeras familias superan las 1.000 muestras y la primera familia que supera las 2.000 muestras es común para los dos conjuntos de datos.

5.3 Métricas

Para evaluar la exactitud de este modelo de clasificación se utilizará las métricas precisión, Recall y F-Measure, las cuales nos indicaran si el enfoque utilizado para clasificar el malware es adecuado, pues este será comparado con otro Conjunto de datos existente previamente clasificado que será utilizado como el Ground truth. Existen varias formas de representar las métricas antes mencionadas que se utilizan en la evaluación de este trabajo [26, 40, 81]. En un trabajo previo del autor [40] se describe cada una de estas métricas las cuales son citadas de forma textual a continuación.

Tabla 5.3: Top10 familias clasificadas con AVClass [4].

	virushare_308			virushare_309	
Top	**Familia**	**Muestras**		**Familia**	**Muestras**
1	installmonster	2.076		installmonster	2.517
2	razy	1.494		fairet	1.544
3	fairet	1.293		razy	1.052
4	zusy	186		zusy	186
5	mywebsearch	85		mikey	51
6	poison	81		winwrapper	50
7	vittalia	58		emotet	42
8	starsurf	54		somoto	39
9	gamarue	41		midie	37
10	delf	34		speedingupmypc	36

5.3.1 Precision

"Al implementar algoritmos de clustering Intuitivamente, nos esforzamos por alcanzar resultados en los que cada grupo contiene sólo los elementos de un tipo particula. El objetivo de la precisión (Precision) es medir la eficacia de un algoritmo de clustering distinguiendo entre las muestras que son diferentes. Es decir, la precisión captura como de bien un algoritmo de clustering asigna muestras de diferentes tipos para diferentes grupos. Más formalmente, la precisión se define de la siguiente manera: Supongamos que tenemos un clustering de referencia $T = T_1, T_2, \ldots T_t$ con t clústers y un clustering $C = C1, C2, \ldots, Cc$ con c clústers (para un conjunto de muestra $A = a1, a2, \ldots, an$). Por cada $C_j \in C$, calculamos el valor Precision del clúster como" [40]:

$$P_j = max(|C_j \cap T_1|, |C_j \cap T_2|, \ldots, |C_j \cap T_t|) \tag{5.1}$$

El valor total de Precision es:

$$Precision = \frac{(|C_1|P_1 + |C_2|P_2 + \cdots + |C_c|P_c)}{n}) \tag{5.2}$$

Donde n es el número total de elementos.

5.3.2 Recall

"Claramente, preferimos una agrupación donde todos los elementos de un tipo se asignen al mismo grupo. El Recall se utiliza para medir qué tan bien un algoritmo de agrupamiento reconoce muestras similares. Es decir, el Recall capta lo bien que un algoritmo asigna muestras del mismo tipo al mismo grupo. Definimos formalmente el Recall de la siguiente manera: Supongamos que tenemos una agrupación de referencia $T = T1, T2, \ldots, Tt$ con t clústers y un clustering $C = C1, C2, \ldots, Cc$ con c clústers. Para cada $Tj \in T$, calculamos el valor Recall del clúster de la siguiente forma" [40]:

$$R_j = max(|C_1 \cap T_j|, |C_2 \cap T_j|, \ldots, |C_c \cap T_j|) \tag{5.3}$$

El valor total de Recall es:

$$Recall = \frac{(|T_1|R_1 + |T_2|R_2 + \cdots + |T_t|R_t)}{n}) \tag{5.4}$$

Donde n es el número total de elementos.

5.3.3 F-Measure

"El algoritmo de clasificación primitivo que crea un clúster para cada muestra alcanza una *Precision* óptima, pero el peor *Recall*. El algoritmo que combina todas las muestras en un solo grupo, en cambio, logra un *Recall* óptimo, pero el peor *Precision*. En la práctica, un algoritmo debe proveer tanto de alta *Precision* y *Recall*. Es decir, cada grupo debe contener todas las muestras de un tipo, pero no más. El *F-Measure* es una métrica que combina el *Precision* y el *Recall* es la media armónica de Precision y el Recall. El valor de F-Measure es" [40]:

$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \tag{5.5}$$

5.4 Resultados de la clasificación con las firmas desarrolladas

Una vez que las muestras de nuestros dos conjuntos de datos fueron analizadas, en el reporte de Cuckoo observamos las muestras que fueron clasificadas como miembros de cada una de las familias. Del modo en que esta diseñada nuestra arquitectura esta no requiere un algoritmo de clasificación, debido a que cuando una firma es detectada esta directamente nos indica el nombre de la familia. En la figura 5.1 podemos ver el ejemplo de uno de los reportes de Cuckoo para la familia installmonster, en este caso el archivo fue detectado debido a que se creaba el mutex especifico que añadimos cuando creamos esta firma. Para obtener este reporte se ha realizado la consulta en mongodb que se muestra en el archivo 5.1 esta consulta nos devolverá las detecciones de nuestra firma para la familia *installmonster* y el md5 del archivo analizado.

Archivo 5.1: Consulta de las detecciones de la familia installmonster

```
1  use cuckoo;
2  db.getCollection("analysis").find(
3    {
4      "signatures.families" : "installmonster"
5    },
6    {
7      "target.file.md5" : 1,
8      "signatures" : 1
9    }
10 );
```

Para obtener todos los reportes de las muestras analizadas, utilizamos el mismo script del archivo 5.1 eliminando la linea *4*, para que no restrinja los resultados por familia y nos devuelva los resultados de todos los análisis. Dado que solo necesitamos el md5 de cada archivo analizado y los resultados de las firmas, mantendremos la proyección de la consulta sobre los dos campos *signatures* y *target.file.md5*. Estos resultados los exportaremos a un archivo JSON.

Mediante el script de python que se presenta en el archivo 5.2 se parsea el archi-

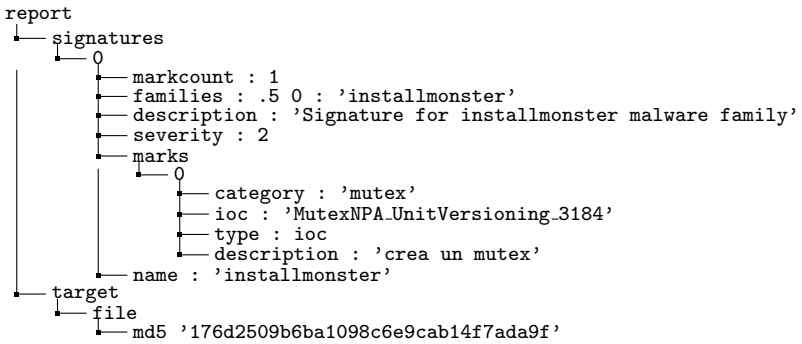

Figura 5.1: Reporte de ejemplo de la detección exitosa de la firma de la familia *installmonter*.

vo JSON de mongodb y obtenemos los resultados de la clasificación en otro archivo JSON. Estos resultados agrupan a los md5 de cada archivo en su respectiva familia detectada. Estos resultados se presentan en la tabla 5.4 y se analizan en la siguiente sección. Para evaluar nuestros resultados utilizamos las métricas presentadas en la sección 5.3, utilizando nuestra clasificación obtenida con el script del archivo 5.2 que se encuentra ya separada en clusters e familias y la clasificación que nos proporciona AVClass siendo esta la clasificación de referencia. Para el cálculo de las métricas se usa el proceso seguido por el autor en un trabajo similar [40].

Archivo 5.2: Script para obtener los resultados de la detección

```
1  #!/usr/bin/env python
2  import json
3  from collections import defaultdict
4
5  RESULTS = "results.json"
6
7  def main():
8    families = defaultdict(list)
9
10   with open(RESULTS) as f:
11     results = json.load(f)
12
13   for elem in results:
14     if elem["signatures"]:
15       for fam in elem["signatures"]["families"]:
16         families[elem["signatures"]].append(elem["target"]["file"]["md5"])
17
```

```
18    json.dump(families, open("families.json","w"))
19
20  if __name__ == '__main__':
21  main()
```

5.5 Análisis de los resultados

En este apartado se presenta de forma consolidada los resultados obtenidos en éste trabajo de investigación. El objetivo de este trabajo es evaluar la exactitud con la que clasifican el malware las firmas desarrolladas para la SandBox Cuckoo.

En la tabla 5.4 se muestra el numero de muestras clasificadas por AVClass y por las firmas de Cuckoo que desarrollamos en este proyecto. Por cada familia se presenta el número de muestras y el porcentaje con respecto a las muestras de AVClass. En general podemos observar que las tasas de detección de nuestras firmas son buenas, 4 familias *vittalia, starsurf, somoto y speedingupmypc* alcanzan un 100 %. Por el contrario tenemos 2 familias donde los porcentajes son bajos entre de menos del 30 % para las familias *delf y midie*.

Tabla 5.4: Familias clasificadas con las firmas desarrolladas.

	virushare_308				**virushare_309**		
Top	**Familia**	**AVClass**	**Firmas**		**Familia**	**AVClass**	**Firmas**
1	installmonster	2076	1835 (88 %)		installmonster	2517	2234(89 %)
2	razy	1494	824 (55 %)		fairet	1544	1338(87 %)
3	fairet	1293	1111 (85 %)		razy	1052	516(49 %)
4	zusy	186	92 (49 %)		zusy	186	74(40 %)
5	mywebsearch	85	71 (83 %)		mikey	51	36(71 %)
6	poison	81	65 (80 %)		winwrapper	50	19(38 %)
7	vittalia	58	58 (100 %)		emotet	42	31(74 %)
8	starsurf	54	54 (100 %)		somoto	39	39(100 %)
9	gamarue	41	37 (90 %)		midie	37	10(27 %)
10	delf	34	6 (18 %)		speedingupmypc	36	36(100 %)

En la tabla 5.5 se presenta el análisis de los resultados que evalúan los resultados

Tabla 5.5: Análisis de resultados.

Dataset	**Precision**	**Recall**	**F-measure**
VirusShare_308	84 %	100 %	91.3 %
VirusShare_309	81 %	100 %	89.5 %

de las firmas tomando como la clasificación de referencia la de AVClass. Debido a que nuestras firmas nos entregan directamente el nombre de la familia en la que una muestra fue clasificada. La métrica *Recall* es la ideal, pues por cada familia solo se puede crear un clúster o grupo. En el caso de la métrica *Precision* en el primer conjunto de datos obtenemos un 84 % y con el segundo un 81 %. Esto indica que hay muestras que nuestras firmas las clasifican como una familia, pero de acuerdo a la clasificación de AVClass estas muestras corresponden a otra familia. Estos casos deben ser analizados en detalle, ya que esto indica que algunas muestras están mal clasificadas. Aunque esto esta fuera del alcance del proyecto sería interesante conocer las causas de estos falsos positivos, lo cuál se podría utilizar para mejorar las firmas que hemos creado, haciendo que sean más especificas para cada familia y así aumentar la métrica *Precision*.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1 Conclusiones

Actualmente el malware sigue siendo uno de los desafíos más grandes, para los analistas, investigadores y la industria. El lanzamiento frecuente de nuevas variantes de muestras de una misma familia de malware, debido a las técnicas de ofuscación que utilizan los autores del software malicoso, hacen que la detección y clasificación del malware sea cada vez un problema más complejo. Por lo que proponer técnicas que ayuden a mitigar la brecha que existe entre los millones de muestras que aparecen cada día y las muestras que logran ser detectadas y clasificadas.

A lo largo de este trabajo analizamos el software malicioso desde varios puntos de vista. Por una parte estudiamos como trabajan las herramientas de antivirus para detectar el malware, las técnicas que utilizan y los métodos de análisis. Estas herramientas en muchos casos no se preocupan por clasificar el malware, puesto que su objetivo principal es la detección. Por otra parte estudiamos las técnicas que se han propuesto en investigaciones científicas, para analizar y clasificar el malware. Sobre esto es importante recalcar que la clasificación del malware tiene igual importancia que la detección, pues las mejoras en las técnicas de clasificación permiten tener mayores tasas de detección y por el lado contrario las mejoras en las técnicas de detección permiten clasificar con mayor exactitud nuevas variantes de malware de familias conocidas.

Las contribuciones de este trabajo se representan mediante la consecución de los objetivos planteados. A continuación se describe brevemente los pasos realizados para alcanzar el objetivo principal(OP) y los objetivos secundarios(OS).

- **OP** *El objetivo principal de este trabajo era evaluar la exactitud que presenta un enfoque de clasificación de malware utilizando las firmas de Cuckoo.* Para esto hemos evaluado la utilización de las firmas de Cuckoo SandBox, para directamente detectar y clasificar muestras de malware. Hemos desarrollado un conjunto de 16 firmas de malware de familias conocidas que han sido evaluadas con dos conjuntos de datos que comprenden un total de 16K muestras de software malicioso. Para algunas familias de malware podemos ver que la tasa de detección no fue tan alta, esto puede deberse a que las firmas deben considerar más comportamientos comunes de una familia. Con nuestros resultados el índice de clasificación evaluado por la métrica *F-measure* alcanza un 91 %.

- **OS1.** *Este objetivo pretende realizar un estudio sobre el estado del arte de la detección y clasificación de malware.* Para esto se revisaros estudios tanto académicos como de la industria de los últimos años, para poner en contexto los tipos de análisis que se realiza del software malicioso, las categorías o tipos de malware que podemos encontrar actualmente, las técnicas de clasificación y los entornos de análisis de malware más utilizado actualmente, tanto de software libre como privativo.

- **OS2.** *Este objetivo pretende diseñar e implementar una plataforma de para la detección y clasificación de malware, utilizando Cuckoo SandBox y firmas de familias de malware.* Para esto se analizó las necesidades de nuestro piloto experimental, diseñando la plataforma de acuerdo a nuestras requerimientos, para después implementar la plataforma de detección y clasificación de malware utilizando Cuckoo SandBox, detallando los pasos necesarios para ponerla en marcha. Y Se crearon 16 firmas de familias de malware que serán compartidas con la comunidad [10].

- **OS3.** *Este objetivo pretende evaluar y analizar los resultados con un conjunto de referencia.* Para esto utilizamos la herramienta externa AVClass, la que nos proporciono una clasificación de referencia de nuestros conjuntos de muestras, la cual se utilizó para evaluar la exactitud de nuestro modelo de clasificación con las firmas creadas para Cuckoo SandBox. Finalmente se realiza una breve discusión sobre los resultados obtenidos

6.2 Trabajo Futuro

Esta sección se presenta los trabajos futuros que se puede seguir investigando o implementando a partir de este trabajo.

- Mejorar las firmas creadas, e integrarlas con las firmas creadas por la comunidad de Cuckoo SandBox.
- Evaluar este modelo de clasificación con conjuntos de datos más grandes de muestras de malware.
- Realizar un estudio sistemático del estado del arte de las actuales herramientas de análisis de malware, para conocer los beneficios que proporciona cada una de estas herramientas.

Bibliografía

[1] P. Security, "Panda Security as a Visionary in Gartner Magic Quadrant," 2018. https://www.pandasecurity.com/gartner-magic-quadrant/.

[2] AVcomparatives, "File Detection Test September 2016 — AV-Comparatives," 2016. https://www.av-comparatives.org/tests/file-detection-test-september-2016/.

[3] Cuckoo Foundation, "Cuckoo sandbox book," 2018. https://cuckoo.sh/docs/.

[4] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "Avclass: A tool for massive malware labeling," in *Symposium on Research in Attacks, Intrusions, and Defenses*, 2016.

[5] OPSWAT, "cyber security report 2017," 2018. https://www.f-secure.com/documents/996508/1030743/cyber-security-report-2017.

[6] A. K. Lab, "IT threat evolution Q1 2018. Statistics - Securelist," 2018. https://securelist.com/it-threat-evolution-q1-2018-statistics/85541/.

[7] M. Vasilescu, L. Gheorghe, and N. Tapus, "Practical malware analysis based on sandboxing," in *RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, 2014*, pp. 1–6, IEEE, 2014.

[8] A. Balakrishnan and C. Schulze, "Code obfuscation literature survey," *CS701 Construction of compilers*, vol. 19, 2005.

[9] W. Wright, D. Schroh, P. Proulx, A. Skaburskis, and B. Cort, "The sandbox for analysis: concepts and methods," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 801–810, ACM, 2006.

[10] R. Rivera, "Cuckoo-classification," 2018. https://github.com/rich7mcs/Cuckoo-Classification.

[11] M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario, "Automated Classification and Analysis of Internet Malware," in *RAID*, September 2007.

[12] U. Bayer, A. Moser, C. Kruegel, and E. Kirda, "Dynamic analysis of malicious code," *Journal in Computer Virology*, vol. 2, no. 1, pp. 67–77, 2006.

[13] T. Micro, "Malware: 1 million new threats emerging daily," 2015. https://blog.trendmicro.com/malware-1-million-new-threats-emerging-daily.

[14] M. Christodorescu, J. Kinder, S. Jha, S. Katzenbeisser, and H. Veith, "Malware normalization," tech. rep., University of Wisconsin, 2005.

[15] Wikipedia, "Antivirus software," 2018. https://en.wikipedia.org/wiki/Antivirus_software.

[16] Gartner, "Cybersecurity Research – Gartner," 2018. https://www.gartner.com/en/insights/cybersecurity.

[17] AVTEST, "El mejor software antivirus para Windows Usuarios Finales," 2018. https://www.av-test.org/es/antivirus/usuarios-finales-windows/.

[18] AVcomparatives, "Real-World Protection Test April 2018 – Factsheet," 2018. https://www.av-comparatives.org/tests/real-world-protection-test-april-2018-factsheet/.

[19] S. Sen, E. Aydogan, and A. I. Aysan, "Coevolution of mobile malware and anti-malware," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2563–2574, Oct 2018.

[20] A. Mohaisen and O. Alrawi, "AV-Meter: An Evaluation of Antivirus Scans and Labels," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, July 2014.

[21] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pp. 113–120, IEEE, 2013.

[22] K. Griffin, S. Schneider, X. Hu, and T.-C. Chiueh, "Automatic generation of string signatures for malware detection," in *International workshop on recent advances in intrusion detection*, pp. 101–120, Springer, 2009.

[23] J. O. Kephart, "Automatic extraction of computer virus signatures," in *Proc. 4th Virus Bulletin International Conference, Abingdon, England, 1994*, pp. 178–184, 1994.

[24] Y. Project, ": a malware identification and classification tool," 2013. http://virustotal.github.io/yara/.

[25] OPSWAT, "Understanding Heuristic-based Scanning vs. Sandboxing," 2015. https://www.opswat.com/blog/understanding-heuristic-based-scanning-vs-sandboxing.

[26] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, Behavior-Based Malware Clustering," in *Network and Distributed System Security*, 2009.

[27] E. Carrera, "Pefile," 2018. https://github.com/erocarrera/pefile.

[28] G. Díaz and J. R. Bermejo, "Static analysis of source code security: Assessment of tools against samate tests," *Information and Software Technology*, vol. 55, no. 8, pp. 1462 – 1476, 2013.

[29] J. Bermejo, "Apuntes del profesor - Seguridad en el Software: Análisis de Malware," *Universidad Internacional de La Rioja*, 2014.

[30] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, “Large-Scale Malware Classification using Random Projections and Neural Networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[31] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, “Learning and Classification of Malware Behavior,” in *Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008.

[32] K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic Analysis of Malware Behavior using Machine Learning,” *Journal of Computer Security*, vol. 19, no. 4, 2011.

[33] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, “Automated Classification and Analysis of Internet Malware,” in *International Symposium on Recent Advances in Intrusion Detection*, 2007.

[34] R. Perdisci, W. Lee, and N. Feamster, “Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces,” in *USENIX Symposium on Networked Systems Design and Implementation*, 2010.

[35] M. Z. Rafique and J. Caballero, “FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors,” in *International Symposium on Research in Attacks, Intrusions and Defenses*, 2013.

[36] J. Canto, M. Dacier, E. Kirda, and C. Leita, “Large scale malware collection: lessons learned,” in *IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*, Citeseer, 2008.

[37] F. Maggi, A. Bellini, G. Salvaneschi, and S. Zanero, “Finding Non-Trivial Malware Naming Inconsistencies,” in *International Conference on Information Systems Security*, 2011.

[38] J. Niemala, “It’s signed, therefore it’s clean, right?,” May 2010. Presentation at the CARO 2010 Workshop.

[39] D. Beck and J. Connolly, “The Common Malware Enumeration Initiative,” in *Virus Bulletin Conference*, 2006.

[40] R. Rivera, “Análisis de características estáticas de ficheros ejecutables para la clasificación de malware,” 2014. http://oa.upm.es/34343/.

[41] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, “A survey on automated dynamic malware-analysis techniques and tools,” *ACM computing surveys (CSUR)*, vol. 44, no. 2, p. 6, 2012.

[42] “Anubis.” http://anubis.iseclab.org/.

[43] J. Koret, “Zero Wine: Malware Behavior Analysis,” 2009. http://zerowine.sourceforge.net/.

[44] JOE SECURITY LLC, “Joe Sandbox,” 2018. http://www.joesecurity.org.

[45] AutoIt Consulting Ltd., “Autoit scripting language,” 2018. https://www.autoitscript.com/site/autoit/.

[46] C. Willems, T. Holz, and F. Freiling, “Toward Automated Dynamic Malware Analysis Using CWSandbox,” *IEEE Security & Privacy*, vol. 5, no. 2, 2007.

[47] S. GFI, “GFI SandBox,” 2011. http://www.gfi.com/malware-analysis-tool.

[48] FireEye Inc., “Advanced malware protection and analysis fireeye,” 2018. https://www.fireeye.com/solutions/malware-analysis.html.

[49] Palo Alto Networks, Inc., “Wildfire malware analysis,” 2018. https://www.paloaltonetworks.com/products/secure-the-network/subscriptions/wildfire.html.

[50] “Virustotal,” 2018. https://virustotal.com/.

[51] P. Kotzias, S. Matic, R. Rivera, and J. Caballero, “Certified PUP: Abuse in Authenticode Code Signing,” in *ACM Conference on Computer and Communication Security*, 2015.

[52] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Álvarez, "Puma: Permission usage to detect malware in android," in *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*, pp. 289–298, Springer, 2013.

[53] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pp. 15–26, ACM, 2011.

[54] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. G. Bringas, and G. Álvarez Marañón, "Mama: Manifest analysis for malware detection in android," *Cybernetics and Systems*, vol. 44, no. 6-7, pp. 469–488, 2013.

[55] R. Rivera, P. Kotzias, A. Sudhodanan, and J. Caballero, "Costly freeware: A systematic analysis of abuse in download portals," *IET Information Security*, 2018.

[56] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current android malware behaviors," in *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014 Third International Workshop on*, pp. 3–17, IEEE, 2014.

[57] J. A. Morales, A. Al-Bataineh, S. Xu, and R. Sandhu, "Analyzing and exploiting network behaviors of malware," in *International Conference on Security and Privacy in Communication Systems*, pp. 20–34, Springer, 2010.

[58] R. Rivera, P. Kotzias, A. Sudhodanan, and J. Caballero, "Detecting PUP through Installation and Uninstallation Behaviors," 2019.

[59] Cuckoo Foundation, "Malwr - the online malware analysis and research platform," 2018. https://malwr.com/.

[60] "Cuckoo Sandbox," 2018. https://cuckoosandbox.org.

[61] ORACLE, “Guests hang in r̈estoring snapshotör d̈eleting snapshot,̈” 2017. https://www.virtualbox.org/ticket/17141.

[62] ORACLE, “Oracle vm virtualbox,” 2018. https://www.virtualbox.org/.

[63] Fabrice Bellard, “Qemu,” 2018. https://www.qemu.org/.

[64] VMware, Inc, “Vmware – official site,” 2018. https://www.vmware.com/.

[65] VMware, Inc, “Software de virtualización de servidores — vsphere — vmware,” 2018. https://www.vmware.com/es/products/vsphere.html.

[66] Citrix Systems, Inc, “Xenserver — open source server virtualization,” 2018. https://xenserver.org/.

[67] VMware, Inc, “Hipervisor nativo vsphere esxi,” 2018. https://www.vmware.com/es/products/esxi-and-esx.html.

[68] Open Virtualization Alliance, “Kvm,” 2018. https://www.linux-kvm.org/page/Main_Page.

[69] FOG Project, “Fog project,” 2018. https://fogproject.org/.

[70] StatCounter, “Desktop operating system market share worldwide — statcounter global stats,” 2017. http://gs.statcounter.com/os-market-share/desktop/worldwide.

[71] “Hardening Cuckoo Sandbox against VM aware malware.” https://www.alienvault.com/blogs/labs-research/hardening-cuckoo-sandbox-against-vm-aware-malware.

[72] O. Ferran, “How to detect the Cuckoo Sandbox and hardening it?,” in *EICAR Annual Conference*, 2013.

[73] M. Corporation, “Microsoft portable executable and common object file format specification,” 1999. http://www.openwatcom.org/ftp/devel/docs/pecoff.pdf.

[74] Microsoft, "Softwarebundler:win32/installmonster," 2017. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=SoftwareBundler:Win32/InstallMonster.

[75] Symantec, "Adware.installmonster," 2016. https://www.symantec.com/security-center/writeup/2016-072113-4836-99.

[76] Lavasoft, "Application.bundler.installmonster," 2016. http://secure.lavasoft.com/mylavasoft/malware-descriptions/blog/GenVariantApplicationBundlerInstallMonster1a5f1709b88.

[77] Trend Micro, "Emerging threat on fareit," 2017. https://success.trendmicro.com/solution/1114357-emerging-threat-on-fareit.

[78] Microsoft, "Ws:win32/fareit," 2017. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=PWS:WIN32/FAREIT.

[79] Cisco Blog, "Down the rabbit hole: Botnet analysis for non-reverse engineers," 2017. https://blogs.cisco.com/security/talos/fareit-analysis.

[80] "Virusshare.com repository," 2018. http://virusshare.com/.

[81] J. R. Bermejo and G. Díaz, "Codificación segura de aplicaciones: Análisis, diseño y desarrollo sin vulnerabilidades de seguridad," *UNED*, 2009.