MACHINE LEARNING APPLIED TO THREAT INTELLIGENCE

Joseph Nikolay Arboleda Díaz

Leslie Tatiana Medina Ruiz

Director de Proyecto

Gerardo Ospina Hernández

Escuela Colombiana de Ingeniería Julio Garavito

Proyecto de Grado

Ingeniería de Sistemas

Bogotá, 2017.

HOJA DE APROBACIÓN

Este trabajo de grado, intitulado "**Machine Learning applied for Threat Intelligen**ce" presentado por los estudiantes **Joseph Nikolay Arboleda Díaz** y **Leslie Tatiana Medina Ruiz**, ha sido aprobado de acuerdo a los reglamentos de la Escuela Colombiana de Ingeniería Julio Garavito, por el jurado integrado por los profesores:

Nombres:	Firmas:
Gerardo Ospina Hernández	_____
Daniel Orlando Díaz	_____
Claudia Patricia Santiago	_____

_____

Oswaldo Castillo Navetty

Decano Programa de Ingeniería de Sistemas

Bogotá, Diciembre de 2017.

Contenido

Resumen....4
Introducción....7
Justificación....9
Objetivos....10
Objetivo General....10
Objetivos Específicos....10
Marco Teórico....11
1. Machine Learning....11
A. General.....11
B. Algoritmos.....13
2. Seguridad....14
A. General.....14
Estado del Arte....16
1. Machine Learning y Seguridad....16
A. Aplicaciones.....16
Desarrollo del Proyecto....29
1. Investigación y Pruebas Básicas....29
A. Instalación y Configuración de Cuckoo Sandbox.....31
B. Instalación y Configuración de Weka.....49
2. Resultados de Investigación y Pruebas....56
3. Elección y creación del algoritmo de Aprendizaje Automático....57
A. Búsqueda y Selección de Información....57
B. Extracción de Información....58
C. Normalización de Datos....59
D. Aplicando Aprendizaje Automático (Entrenamiento)....59
E. Aplicando Aprendizaje Automático (Clasificador)....59
4. Resultados Aplicación de Algoritmo....60
Conclusiones....61
Recomendaciones....62
Glosario....63
Bibliografía....65

Tabla de Ilustraciones

Ilustración 1 Comando actualización de distribución 31
Ilustración 2 Instalación de dependecias....31
Ilustración 3 Instalación de librerías....32
Ilustración 4 Actualización librería PIP 32
Ilustración 5 Comando de Instalación de Cuckoo....32
Ilustración 6 Instalación VirtualBox....32
Ilustración 7 Ejecución de Máquina Virtual....32
Ilustración 8 Características de Guest OS 33
Ilustración 9 Comando de instalación MongoDB 34
Ilustración 10 Comando de Instalación tcpdump 34
Ilustración 11 Comando instalación librería tcpdump....34
Ilustración 12 Ejecución de comando 1 34
Ilustración 13 Ejecución de comando 2 34
Ilustración 14 Instalación librería GIT 34
Ilustración 15 Comando para instalación de Volatilidad 35
Ilustración 16 Página de descarga Distorm 35
Ilustración 17 Instalación Distorm....35
Ilustración 18 Comando de instalación Autoreconf 35
Ilustración 19 Instalación librería Libtool-bin 35
Ilustración 20 Página de instalador Yara 36
Ilustración 21 Página de descarga Pycrypto 36
Ilustración 22 Instalación Pycrypto....36
Ilustración 23 Instalación de Openpyxl....37
Ilustración 24 Instalación UJSON 37
Ilustración 25 Instalación de Jupyter/IPython 37
Ilustración 26 Instalación librerías Mitmproxy 37
Ilustración 27 Archivos de configuración Cuckoo Sandbox 37
Ilustración 28 Visualización VirtualBox con Snapshot 40
Ilustración 29 Inicialización del Proxy....41
Ilustración 30 Creación de Snapshot Máquina Virtual....41
Ilustración 31 Máquina virtual en ejecución 42
Ilustración 32 Cuckoo Sandbox en ejecución 42
Ilustración 33 Ejecución interfaz gráfica Cuckoo Sandbox....43
Ilustración 34 Página de análisis Cuckoo Sandbox....43
Ilustración 35 Elección de archivo para prueba....44
Ilustración 36 Configuración de análisis 44
Ilustración 37 Configuración para análisis 2 45
Ilustración 38 Restauración de máquina virtual 45
Ilustración 39 Estado de tareas 46
Ilustración 40 Estado de ejecución de la herramienta 46

Ilustración 41 Página de reportes y puntuación de archivos 47
Ilustración 42 Página de reporte 1 47
Ilustración 43 Página de reporte 2 48
Ilustración 44 Página de reporte 3 48
Ilustración 45 Página principal Weka 49
Ilustración 46 Archivos estables de instalación 50
Ilustración 47 Instalador Weka 50
Ilustración 48 Elección de componentes Weka 51
Ilustración 49 Finalización instalación Weka 51
Ilustración 50 Ventana principal Weka 52
Ilustración 51 Weka Explorer 52
Ilustración 52 Weka Explorer - Iris.arff 53
Ilustración 53 Elección de algoritmo 53
Ilustración 54 Resultados algoritmo J48 54
Ilustración 55 Visualización de distribución por atributos 55
Ilustración 56 Visualización de dos atributos 55

Resumen

El uso de nuevos almacenamientos como la nube, han mejorado el rendimiento de varios algoritmos de aprendizaje y no solo esto, los algoritmos que aprenden reglas de clasificación, patrones de reconocimiento, árboles de decisión entre otros, en conjunto pueden generar un conjunto de posibilidades infinitas [21]. Y como la tecnología avanza rápidamente, la seguridad debe avanzar de igual manera, para la mejor protección de la información como un valioso activo para la sociedad. Así es como los algoritmos de aprendizaje automático pueden convertirse en una ayuda para esta protección, logrando analizar y predecir de manera efectiva actividad sospechosa o un comportamiento inadecuado de un archivo dentro de un sistema, que un humano podría pasar por alto por diversas situaciones. Lo que se busca con este proyecto es introducir los términos de *Machine Learning* e investigar los diferentes algoritmos que se pueden desarrollar actualmente, con herramientas como Weka y Cuckoo Sandbox, además de conocer diferentes aplicaciones teóricas y prácticas de los diferentes aspectos de la Seguridad Informática, con el fin de conocer aquellas necesidades aún no cubiertas por el *Machine Learning*. También se dará a conocer el procedimiento para la creación de una Red Neuronal Artificial, con la capacidad de clasificar archivos en tres tipos de familia de malware, producto final del proyecto.

Introducción

Desde la aparición de las computadoras dentro de la cotidianidad de las personas y procesos tecnológicos, ha sido una constante pregunta si estas poseen la capacidad de aprendizaje sin ser guiadas por un ser humano en su quehacer. Para resolver esta pregunta nace el *Machine Learning*, que es definido como "field of study that gives computers the ability to learn without being explicitly programmed" por Arthur Samuel en 1959 [7]. Esto puede ser de gran utilidad dentro de la sociedad, por su capacidad de predecir futuros entornos con la información recopilada a lo largo del tiempo. El *Machine Learning* podría desarrollarse en muchos entornos, pero quizás el más relevante de ellos sea la seguridad de datos informáticos.

El propósito de este documento introductorio se divide en varias partes. En primer lugar se introduce el término de *Machine Learning*, en el cual de forma concreta se busca definir su función y aplicación sobre el campo de la seguridad y la inteligencia de amenazas. En este punto también se analizaron los diferentes tipos de algoritmos de Aprendizaje Automático y cómo podrían funcionar en los campos antes mencionados. En segundo lugar, se hablará del proyecto para móviles SeraphimDroid y de su surgimiento y orientación a las amenazas y los riesgos para los usuarios del sistema operativo Android [22]. Además, se investigó a fondo esta aplicación en la cual se analizó su código para entender de forma práctica el *Machine Learning* y cómo este algoritmo funciona en la detección temprana de ataques a dispositivos móviles.

Además de esta aplicación, se conocerán más formas de ejecución de *Machine Learning* dentro del ámbito de seguridad, generando un estado del arte de los algoritmos más utilizados en

este campo de la informática y así poder conocer los diferentes aportes que se han generado en torno a esta rama de la inteligencia artificial. También se realizaron diferentes aplicaciones prácticas de algunos algoritmos de Aprendizaje Automático, generando nuevo conocimiento para nuevos proyectos de grado y aportando al proyecto de Investigación en el que se encuentra actualmente incluido nuestro proyecto de grado. También se realizaron diferentes análisis de aplicación de aprendizaje automático a un sistema de análisis de malware para tener diferentes ideas de aplicación en un futuro a este sistema.

Justificación

La gestión de la seguridad de la información busca resolver diferentes retos asociados a la gran cantidad de activos a proteger y la aparición constante de nuevas vulnerabilidades y amenazas. Estos retos han sido abordados generalmente desde una perspectiva determinística, sin embargo, en los últimos años se han destacado las soluciones de seguridad que incorporan técnicas de aprendizaje automático obteniendo resultados más rápidos y precisos. A pesar de la utilidad de estas soluciones, sigue habiendo una brecha importante en la aplicación de dichas técnicas en escenarios reales. Por medio del presente proyecto se busca explorar la aplicación de técnicas de aprendizaje automático (*machine learning*) para análisis forense y para el manejo de grandes volúmenes de eventos de seguridad.

Objetivos

Objetivo General

Realizar la revisión, implementación, aplicación y prueba de diferentes algoritmos de aprendizaje automático a problemas de seguridad de la información que permitan detectar y predecir ataques de seguridad, apoyar labores de ciberinteligencia y anticipar situaciones de alto impacto organizacional debidas a incidentes informáticos.

Objetivos Específicos

- Identificar los algoritmos de aprendizaje automático y lenguajes más utilizados actualmente para resolver problemas de seguridad de la información tales como identificación y predicción de ataques, análisis forense, gestión de incidentes, etc.

- Definir un escenario de seguridad, realizar una prueba funcional y generar un prototipo de aplicación en la cual se apliquen diferentes algoritmos de aprendizaje automático a una base de datos de múltiples eventos de seguridad para extraer conocimiento.

- Conocer los comportamientos maliciosos de tres diferentes familias de malwares, para la creación de una herramienta de extracción de información desde la herramienta Cuckoo Sandbox.

Marco Teórico

1. Machine Learning

A. General.

Antes de poder guiar el *Machine Learning* hacia un escenario específico, lo mejor es comenzar con la definición general de *Machine Learning*. *Machine Learning* se refiere a la detección automatizada de patrones significativos en los datos. El tema lleva años en vigencia, pues se ha convertido en una herramienta útil en cualquier tarea que requiere obtener información relevante de un conjunto de datos [46]. Que los algoritmos logren aprender de los eventos de su entorno es algo que puede ser muy útil en la sociedad actualmente, donde se puede realizar una gran automatización de procesos que hoy en día son más que necesarios. En algunos de los procesos donde se puede aplicar es el procesamiento de lenguaje, robótica, procesamiento de imágenes, seguridad informática, minería de datos, entre otros. [1, 20] Desde este último, aunque es un gran tema a tratar por la cantidad de escenarios variados que contiene, puede ser de mayor utilidad dentro de la predicción de ataques de malware. Y, en realidad, no sólo en el ámbito informático han probado ser útiles los algoritmos de *Machine Learning*. Su uso varía desde la medicina hasta la biología y redes sociales.

Con respecto a la medicina, dos de las investigaciones más recientes hechas a partir del *Machine Learning* son: *Evaluation of machine learning algorithms and structural features for optimal MRI-based diagnostic prediction in psychosis*, que utilizó algoritmos de *Machine Learning* para la predicción de diagnósticos “sobre las características principales de la resonancia magnética estructural (sMRI) incluyendo morfometría basada en voxel de materia gris y blanca (VBM), espesor y volumen cortical basado en vértices, medidas volumétricas de región de interés y mapas de morfometría basada en ondulaciones (WBM). Todas las posibles combinaciones de

algoritmos y características de los datos se consideraron en las clasificaciones de dos parejas de controles sanos, pacientes con esquizofrenia y pacientes con trastorno bipolar*"* [3].

La segunda toma como título *Testing a machine-learning algorithm to predict the persistence and severity of major depressive disorder from baseline self-reports*. En esta investigación se desarrollaron modelos de aprendizaje de *Machine Learning* a partir de informes sobre episodios del trastorno depresivo mayor, mostrando que aquellos modelos lograron predecir algunas características del trastorno con una buena precisión. Después de hacer la comparación con modelos de regresión que se utiliza normalmente, se pudo concluir que los algoritmos de *Machine Learning* fueron útiles en la generación de modelos de clasificación del riesgo del trastorno, además de mejorar los modelos actualmente usados ya que los síntomas son muy diversos entre sí, lo que complica en gran medida la toma de decisiones clínicas con respecto a su tratamiento [4].

El *Machine Learning* también se puede encontrar en las redes sociales, como se pudo encontrar en el artículo *Validating Machine Learning Algorithms for Twitter Data Against Established Measures of Suicidality*[17] realizada con una aplicación de *Machine Learning*, alimentada por diferentes textos de participantes escogidos por medio de una encuesta. Cada texto fue analizado por un software LIWC (o Linguistic Inquiry and Word Count) para extraer información y señalar aquellos que mostraran señales de angustia.

En la botánica, con respecto al análisis de las hojas de plantas, evidenciado en el artículo *Machine Learning for Plant Leaf Analysis*[16] que se basó en la aplicación de *Machine Learning* para el análisis de las hojas de las plantas, pues estas varían dependiendo de su desarrollo, y

difieren entre ellas incluso en la forma de sus hojas desde que la planta es muy joven, haciendo que su clasificación fuera aún más complicada; o sobre la clasificación y predicción de Péptidos Anti fúngicos en *An Evaluation on Different Machine Learning Algorithms for Classification and Prediction of Antifungal Peptides*[18], donde se logró desarrollar un modelo de predicción para los péptidos anti fúngicos. Se decidió utilizar SVM, quién entregó los mejores resultados en su aplicación, y se consideró una gran herramienta para el descubrimiento de nuevos agentes de focalización de hongos.

B. Algoritmos.

El *Machine Learning*, debido a que se basa en el hallazgo de patrones dentro de la información que tiene, utiliza algoritmos que le puedan ayudar en diferentes tareas. Estos algoritmos se clasifican en aprendizaje supervisado, semi-supervisado, por refuerzo, multi-tarea y transducción [19]. El aprendizaje supervisado es un conjunto de reglas de instancias. Para ser más específicos, crea un clasificador que puede ser usado para generalizar desde nuevas instancias [26]. El siguiente algoritmo es el aprendizaje semi supervisado, que igual al anterior, es una forma de clasificación pero en este se toma en cuenta el problema de abordar los datos sin etiquetar y lo hace utilizando una gran cantidad de datos no etiquetados, juntos con los etiquetados para construir mejores clasificadores [27]. El aprendizaje por refuerzo, contiene un agente conectado con el entorno y al recibir una entrada, se retroalimenta para generar una salida [28]. Uno de los algoritmos, el de aprendizaje multitarea, es donde permite el aprendizaje de manera independiente con respecto a parámetros de tareas para generar conocimiento y así poder enfrentarse a tareas semejantes [29]. Para finalizar, está el algoritmo de transducción que, al igual que el primero, construye un clasificador pero no de manera explícita, y predice las categorías por medio de las entradas, categorías y ejemplos nuevos del sistema.

Supervisado	Decision Tree, kNN, Random Forest, Logistic Regression.
No supervisado	Apriori Algorithm, k-Means, Hierarchical Clustering
Refuerzo	Markov Decision Process, Q Learning

Tabla 1. Algunos algoritmos para tres tipos de aprendizaje encontrado en *Machine Learning*.

2. Seguridad

A. General.

Desde el 2015 se ha presentado un auge de los dispositivos conectados a internet, estos cada vez manejan más información sensible y es necesario pensar ¿Que tanta seguridad garantizan al usuario? ¿Es seguro dar mis datos a estos dispositivos? Estas incertidumbres han sido un tema delicado que se ha venido tratando durante los últimos años y han implicado que se investiguen diferentes territorios inexplorados que los atacantes pueden aprovechar. En el 2016 se presentaron varios ataques y fallas de seguridad entre los cuales se presentaron el ataque DDoS (ataque distribuido de denegación de servicio, en español), Botnet (Red de equipos infectados y controlados de manera externa por un tercero, de modo imperceptible para el usuario), y los exploit kit (software diseñado para identificar vulnerabilidades del software en las máquinas cliente y explotar esas vulnerabilidades). [20]

Dentro de la seguridad informática, se ha destacado el ransomware, que es un tipo de virus capaz de encriptar los archivos de la máquina infectada y exige el pago de una suma, a cambio de la clave para descifrar sus documentos.[2] Aunque este software no es para nuevo, ya que su nacimiento se remonta hace 30 años, con un malware que ocultaba directorios y cifraba nombres de archivos[43], este ha evolucionado para no solo cifrar u ocultar archivos o directorios si no "secuestrar" por completo dispositivos. Por estos diferentes software maliciosos es necesario

destacar el papel importante que ha tomado el *Machine Learning* sobre estos ataques y como ha avanzado en diferentes sectores a través de la gran capacidad de “aprender” para poder detener estos ciberataques [34].

Dentro de las investigaciones realizadas en los últimos dos años, se encuentra un artículo que presenta varios métodos de aprendizaje automático para aplicaciones de seguridad cibernética. En el mismo, se discuten distintos algoritmos, su complejidad y una comparación de cada uno de estos en cuanto a problemas de intrusiones cibernéticas, con el fin de ofrecer un conjunto de características que debe cumplir un problema concreto para que pueda resolverse mediante un algoritmo de aprendizaje automático específico [13]. Es por esto que se ha decidido tratar en este artículo las diferentes aplicaciones del aprendizaje automático junto con la seguridad en los últimos años.

Estado del Arte

1. Machine Learning y Seguridad

A. Aplicaciones.

La capacidad de predicción de los algoritmos de *Machine Learning* puede ser de gran ayuda para las amenazas a la seguridad que han ido en aumento a pasos agigantados en los últimos años, pues permite detectar comportamientos sospechosos [25]. Prueba de esto es la cantidad de ataques sufridos durante el último año y que han sido de gran escala, llegando a atacar infraestructuras críticas como entes gubernamentales y hospitales. Por supuesto, sin olvidar que no sólo se han hecho ataques a computadores de mesa o equipos portátiles, sino también a dispositivos IoT y dispositivos móviles, estos últimos haciendo peligrar tanta información valiosa como cualquier otro dispositivo. Para mejorar el contexto frente a las amenazas que continuamente se desarrollan y mejoran, se han creado aplicaciones y artículos donde estos dos temas han sido utilizados ya sea como una idea a realizarse o como una puesta en marcha sobre el tema.

Uno de los artículos investigados presenta un acercamiento a *Machine Learning* mediante la detección de malware en Android con ayuda del algoritmo de aprendizaje supervisado llamado Support Vector Machine. Durante las pruebas realizadas, se logró probar el sistema creado contra 2081 aplicaciones benignas y 91 aplicaciones maliciosas de Android, para así entrenarlo. Aunque estas pruebas arrojaron una gran efectividad al evitar clasificar aplicaciones malware como benignas, los autores reconocían la necesidad de mejorar el reconocimiento de falsos positivos (aplicaciones benignas como malware). Además de esto, mencionan el cambio de modelo de One-Class Support Vector Machine a un algoritmo de aprendizaje semi supervisado, que sería

mucho más poderoso al tener en cuenta que suelen existir siempre aplicaciones maliciosas y tendría mayor habilidad para discriminar entre características [4].

Pero no debemos olvidar que no sólo Android puede recibir ataques de malware dentro de sus aplicaciones, por lo que la aplicación de *Machine Learning* dentro de iOS es también importante de mencionar. Gracias a una investigación, se construyó una aplicación con 20 diferentes algoritmos de *Machine Learning*, entre los cuales se destaca Support Vector Machine, Partial Decision Tree, Random Forest y Random Tree. Para la evaluación de estos algoritmos, se utilizaron las características de 50 aplicaciones confiables (obtenidas gracias a la App Store de Apple) y 50 aplicaciones malware, conseguidas de la página de Contagio Mobile. De esta evaluación, se obtuvo que el algoritmo One Rule había dado los mejores resultados con respecto a las aplicaciones iOS. Se propuso añadir la detección de ocurrencias bajo el uso del análisis dinámico para identificar familias de malware dentro del sistema de iOS [5].

Aunque en estas dos muestras de unión entre *Machine Learning* y Seguridad Móvil se utiliza Support Vector Machine para identificar las características de la aplicación potencialmente peligrosa, también se pueden identificar mediante los patrones de acceso a la memoria virtual del dispositivo. Dentro de una investigación, se utilizó *Machine Learning* para monitorear y clasificar patrones de acceso a la memoria virtual, usando esta información para clasificar las aplicaciones en potencialmente peligrosas. El resultado de esta investigación fue un *framework online* que permite revisar cualquier comportamiento malicioso según los patrones de acceso, enfocándose en los *kernel rootkits* y los ataques de corrupción de memoria. Se hizo uso de tres algoritmos de *Machine Learning* de aprendizaje supervisado: logistic regression, SVM y random

forest. Con el framework se consiguió una detección del 99%, con menos de 5% de falsos positivo. [31]

Continuando con las investigaciones que han producido frameworks para hacer análisis con *Machine Learning*, podemos encontrar la investigación *StormDroid*. Este framework utilizó seis métodos de clasificación como Support Vector Machines, Decision Tree, Artificial Neural Networks, Naive Bayes, K-Nearest Neighbors y Bagging predictor. Para poder revisar cuál algoritmo podría ser el mejor de los presentados, se realizaron pruebas en dos categorías, siendo la primera aquella que tuviera mejor rendimiento en cuanto a los permisos de las aplicaciones y de llamados a la API (teniendo en cuenta una comparación de patrones entre aplicaciones benignas y maliciosas), y la segunda categoría, añadiendo secuencias y comportamientos dinámicos. Gracias a este último análisis de exactitud de los algoritmos dentro del framework, se pudo encontrar que el mejor clasificador había sido K-Nearest Neighbors, que requirió menos recursos para completar la clasificación.[37]

Uno de los casos particulares en el cual *Machine Learning* se ha aplicado ha sido en *Hidost*, el primer sistema estático de detección de malware, diseñado para operar en múltiples formatos de archivo, este innovador sistema realiza órdenes de magnitud a gran velocidad, combinando la estructura lógica de los archivos con su contenido para tener una precisión bastante efectiva en la detección de malware. Los formatos en los cuales se implementó el sistema fueron PDF y SWF (Flash) y se evaluó con 440.000 archivos PDF y 40.000 archivos SWF superando a muchos de los antivirus famosos en esta área por medio del *Machine Learning*. Es capaz de aprender entre malware malicioso y benigno basado en la estructura lógica del archivo, por ende maneja un tipo de algoritmo de Aprendizaje supervisado el cual maneja seis etapas principales las cuales son: la

extracción de la estructura, Consolidación de trayectoria estructural, selección de características, vectorización, aprendizaje y por último la clasificación [36].

También es importante resaltar un caso que no es tan explorado en el área del *Machine Learning* como lo es DoS (denial-of-service attack) y en tiempo real y este es a menudo encontrado en los servicios de VoIP (voice over IP: 'voz por IP')) en especial los basados en SIP estos modelos son simples y de bajo costo pero por ende son inseguros y son propensos a múltiples ataques. Al ver este problema Zisis Tsiatsikas y un equipo de trabajo mostraron el rendimiento y precisión de detección de DoS en 17 experimentos, en los cuales, realizando varios ataques simulados en los clasificadores, se comportaban muy eficientes incluso con incidentes de baja tasa, llegando a máximo 3,5 ms en promedio con una desviación estándar media de 7,7 ms, todo esto gracias a cinco clasificadores llamados SMoO, Naive Bayes, Neural Networks, Decision Trees y por último Random Forest [42].

Pero no sólo en estas áreas puede ayudar el *Machine Learning*, pues también puede ser aplicado para la computación forense. Aunque en la investigación no se produjo ningún tipo de resultado como en anteriores artículos, pero si se realizó un análisis de la aplicación de *Machine Learning* como método para mejorar el desarrollo forense, concluyendo que la computación forense necesita adaptarse a cada escenario, siendo así necesaria la intervención del ser humano y que el algoritmo elegido tenga la capacidad de aprender de las retroalimentaciones del analista encargado, a diferencia de otras aplicaciones donde la obtención de esta información es mediante bases de datos de eventos previamente recogidos por similitudes.[6]

Uno de los problemas más grandes que ha perjudicado en los últimos años a varias empresas ha sido el ransomware, pero debido a que muchas de estas no están protegidas en lo básico, como lo es el correo electrónico de phishing, se crea la herramienta Phishing Alerting System (PHAS) la cual está en la capacidad de detectar y alertar todo tipo de correos electrónicos engañosos para así ayudar al usuario a tomar decisiones. Con la ayuda de *Machine Learning*, la herramienta extrae y aprende la información de una gran base de datos de e-mail con la cual obtiene un 93.11% de precisión en la detección de correos engañosos usando j48 Decision Tree y KNN [44].

Uno de los métodos más comunes para realizar grandes ataques, es el desarrollo de Botnets debido a las vulnerabilidades de diferentes equipos alrededor del mundo. El entorno empresarial es uno de los objetivos más frecuentes de estos ataques, debido a la cantidad de información importante que manipula, y es por esto que *The Role of Machine Learning in Botnet Detection* es una investigación relevante. Aquí no sólo se probaron algoritmos de aprendizaje supervisado sino también sin supervisión, pero debido a las diferentes limitaciones que aún se poseen (como el monitoreo en tiempo real) se mostró que era efectivo, pero no lo que se esperaba. Por parte de los algoritmos de aprendizaje supervisado no se mostró algún avance significativo, debido a que este tipo de aprendizaje solo funcionaría en casos donde las características de los bots son conocidas, dejando de lado la naturaleza camaleónica de los bots. Por otro lado, los algoritmos sin supervisar pueden causar gran cantidad de falsos positivos por la similitudes que un grupo puede compartir con los ejemplos cargados en él, pero esto último ha sido solucionado por algunos investigadores al presentar características específicas de los grupos formados de bots.[8]

Como uno de los inconvenientes del artículo anterior fue la detección en tiempo real, se probó un algoritmo de Redes Neuronales Artificiales, conocido como Regulación Bayesiana para lograr

un mejor reconocimiento de las características de botnets usados por medio de P2P aunque estos no hayan sido utilizados para su entrenamiento antes, permitiendo una detección de tráfico en tiempo real. Este algoritmo resultó siendo muy eficiente gracias a la generalización que puede realizar y que era necesaria para este tipo de amenazas, donde cada día se descubren nuevos tipos de botnet que difieren bastante de sus antecesores. Para comprobar que su capacidad de analizar la información fuera la más óptima, se comparó con Cyberoam, un sistema de gestión de amenazas unificado, mostrando que el mismo paquete de prueba que Cyberoam encontró como malicioso, también lo identificó el framework BR-ANN. [9]

A diferencia de la última investigación, donde se usó un algoritmo de Redes Neuronales, existe otra forma de realizar una detección de botnets con otro tipo de algoritmo como lo son K-medoids (programado en Java) y K-means, usado desde Weka. Para ambos se utilizaron conjuntos de datos sobre Botnets para entrenar estos algoritmos, usando Botnets centralizados y descentralizados. Bajo un conjunto de reglas (clústeres) para detectarlos en la fase "Comando y Control", se pudo encontrar que K-means es bastante útil en la detección de botnets tipo Storm y Neris que K-medoids, sin embargo, este último era capaz de detectar botnets tipo Rbot, Waledac y botnets descentralizados con un 90% de aciertos y 7% de falsos positivos, necesitando de muy pocas reglas, permitiendo que K-medoids se convierta en una alternativa al algoritmo K-means bajo condiciones específicas.[10]

Los botnet atacan por medio de la red a las diferentes víctimas, dejando actividad sospechosa por donde pasen. Esto también sucede con ataques de otro tipo que deban usar la red como medio por lo que ¿es posible mejorar la detección de anomalías dentro de las conexiones de red con ayuda de *Machine Learning*? Gracias a esta pregunta, se logró dar con una investigación donde

se comenzó con la identificación las ventajas y desventajas que traería aplicar algoritmos supervisados y no supervisados al ambiente de la detección de intrusos en la red, notando que el mejor a aplicar sería un algoritmo semi-supervisado debido a que los conjuntos de datos que se usarían para entrenarlo, no podrían ser clasificados en etiquetas y la forma de conseguirlas podría ser un gran problema. Finalmente se aplicó una Red Neural con peso variado como clasificador por el alto rendimiento que tiene con conjuntos de datos con y sin etiquetas. [14]

Como es bien conocidos los servidores SSH son muy utilizados pero hay que tener en cuenta lo precavido que se debe ser a la hora de usar este servicio ya que existen varios ataques que pueden poner peligro información importante como los ataques de DoS, ataques de phishing, correo electrónico de spam entre otros. Y de esto se encargaron un grupo de colaboradores en el 2017, quienes muestran cómo detectan una sesión SSH comprometida que está llevando a cabo actividades maliciosas, basado en un enfoque de flujo (Los paquetes individuales no se examinan), los datos se extraen de un honeypot y se usa *Machine Learning* con el algoritmo de árbol de decisión J48 el cual se desempeñó muy bien en la detección de sesiones SSH comprometidas, deduciendo por último que el tiempo entre la llegada entre los paquetes y el tamaño de la carga útil de un paquete desempeña un papel importante en la detección de compromisos[40].

Los ataques de Cross-VM han surgido últimamente como una amenaza que preocupa a muchos de los usuarios y empresas que usan nubes comerciales, estos ataques explotan fugas a nivel de hardware en los servidores físicos compartidos. Este ataque inclusive ha tomado las arquitecturas de caché compartidas de alto nivel (LLC), explotando estas vulnerabilidades se pueden recuperar claves criptográficas de populares librerías de software, haciendo que sea

esencial verificar las plataformas que manejan datos sensibles. En el artículo, se muestran cómo por medio de una máquina basada en *Machine Learning* usando un algoritmo supervisado se puede clasificar las aplicaciones de acuerdo a sus perfiles de acceso de caché. Usando vectores de características es posible entrenar modelos usando máquinas vectoriales para una clasificación de alto grado de éxito, demostrando que es posible entrenar modelos para predecir exitosamente aplicaciones que se ejecutan en instancias co-localizadas [11].

La gran mayoría de ataques se realizan para extraer información personal, robo de dinero o llegar a infectar a la mayor cantidad posible de equipos para que sean estos últimos los que lleven a más lugares la contaminación con virus. Una de las razones por las que un computador termina infectado, son las visitas a páginas web maliciosas por lo cual se podría usar *Machine Learning* para realizar una identificación de cuáles páginas son peligrosas para el usuario por ser de URL maliciosa. Aunque se ha usado más de un algoritmo para resolver este problema, la investigación muestra cómo podría usarse el algoritmo C4.5 para la detección y clasificación correcta de páginas maliciosas. Debido a que la clasificación depende de características y factores externos, un enfoque hacia la clasificación, obteniendo buenos resultados por medio de URLs. [12]

Uno de los fenómenos que ha atacado a los teléfonos inteligentes ha sido el botnet, esta red de ordenadores, ordenadores portátiles, dispositivos móviles o tabletas que son controlados remotamente por los ciberdelincuentes para iniciar varios ataques coordinados distribuidos incluyendo correos electrónicos no deseados, fraude en clics de anuncio, minería de Bitcoin, Denegación Distribuida de Servicios (DDoS), difusión de otros malwares y mucho más. En el artículo se propone un SMARTbot, que es un marco de análisis dinámico mejorado con el aprendizaje automático, para detectar automáticamente los binarios de la red de bots maliciosos.

Este genera un modelo de aprendizaje de botnet mediante la inducción del método de retro propagación de las redes neuronales artificiales, creando un conjunto de datos de botnet móvil que puede ser el punto de partida para futuros estudios [15].

Como ya se ha hablado, el ransomware es un tema delicado que no solo ha afectado a múltiples computadores, esta amenaza ha llegado a las plataformas móviles y en particular a Android. En el artículo, se propone una máquina de aprendizaje para la detección de ransomware, aprovechando la información extraída de paquetes del API del sistema que permite mediante un algoritmo supervisado, caracterizar aplicaciones sin conocimiento específico de contenido definido por el usuario. Al probar la máquina resaltaron que los datos muestran que es posible detectar ransomware de Android y distinguirlo de malware genérico con una precisión muy alta. Además de esto los resultados se marcan y son enviados a R-PackDroid para detectar aplicaciones que fueron detectadas como ransomware. Por último concluyen que la máquina fue capaz de distinguir correctamente ransomware verdadero de falsos positivos, proporcionando así una valiosa ayuda para el análisis de estas aplicaciones maliciosas [41].

Una muestra sobre el tema de malware en móviles es M0Droid, que se creó bajo la premisa de un modelo de detección de Malware basado en su comportamiento. Aquí se utiliza el coeficiente de correlación de Spearman, después de normalizar las firmas (o comportamientos de la aplicación) para aumentar la exactitud del resultado. Con la información del coeficiente, se hace una identificación con malwares de comportamientos similares generando una lista negra de firmas, logrando un puntaje de detección del 60.16% con un 39.43% de falsos positivos y un 0.4% de falsos negativos. [30].

Para comprender mejor cómo actúan los algoritmos para llevar a cabo sus funciones, es necesario ver las diferentes aplicaciones prácticas del *Machine Learning*. Hasta el momento hemos visto frameworks realizados por motivos académicos pero ¿es posible encontrar *Machine Learning* en software comercial de seguridad? A continuación veremos el uso de *Machine Learning* en algunos productos de empresas como Cylance y CounterTack, especializadas en la seguridad, que han ganado terreno en el mercado gracias a su uso de *Machine Learning*.

CounterTack posee tres herramientas: Endpoint threat platform, Responder Pro y Threat Scan Pro. Nos enfocaremos específicamente en Endpoint Threat Plataform, encargada de revisar las conexiones de dispositivos a la red y si estos representan una amenaza para la misma, siendo la única de las tres herramientas en usar *Machine Learning* para realizar sus funciones [32]. Actualmente se dedican a la protección de activos empresariales, especialmente los equipos finales de la red (como equipos de cómputo y servidores conectados a la red). No realiza una detección basada en firmas o un análisis binario. CounterTack realiza un análisis comportamental observado dentro del sistema operativo y la memoria, examinando el origen y el efecto en el estado del mismo, alertando de intrusiones maliciosas en ese momento, a diferencia del análisis binario que sólo notificaría de actividad de alto riesgo para los usuarios.[33]

Cylance principalmente desarrolla un antivirus llamado CylancePROTECT, después se encuentran tres herramientas CylanceOPTICS, ThreatZERO™ y por último CylanceV® las cuales son independientes una de la otra pero en conjunto conforman una gran protección teniendo compatibilidad entre ellas [35]. Esta solución de antivirus principalmente se basa en la detección y protección de malware mientras y antes que se ejecute en los computadores. El aprendizaje automático funciona con una precisión inigualable evitando el 99% del malware

existente, analizando bloques estadísticamente similares de código de archivo para identificar archivos maliciosos, para lo cual usa reconocimiento de patrones y análisis predictivo. En lugar de firmas reactivas, las amenazas se bloquean automáticamente en tiempo real [35].

Cybereason por su parte es una compañía de seguridad informática la cual se dedica a la detección de punto final y software de respuesta. Principalmente manejan un producto llamado Cybereason Deep Detect & Respond (EDR), también manejan otros dos productos Deep Prevent (NGAV) y RansomFree y presta tres servicios Deep Monitoring, Deep Hunting y Deep Incident Response. El activo tecnológico que protegen en general son los archivos y redes ya que desarrollan herramientas y técnicas de detección y respuesta ante una amenaza. Se despliegan en 24 horas y todos los datos de los sensores se retransmiten constantemente al motor de caza Cybereason. Recuerda, relaciona y conecta actividades pasadas y presentes haciéndose más inteligente y más eficaz, usando algoritmos de aprendizaje el motor de caza reconoce comportamientos, incluyendo el malware sin archivos y el movimiento lateral. Conecta eventos aparentemente no relacionados o benignos para revelar el alcance completo del ataque [45].

SentinelOne desarrolló un software que remplazaría el antivirus convencional con ayuda de *Machine Learning*. Conocida como "SentinelOne Endpoint Protection Platform", utiliza *Machine Learning* para predecir amenazas de cualquier tipo, comparando el comportamiento de la misma con un comportamiento normal. Los activos que tienen prioridad en su software son los endpoint como son las estaciones de trabajo y servidores conectados a la red, aunque también ofrecen su seguridad a infraestructuras críticas en sectores como salud, educación, energía y finanzas. Esta aplicación unifica prevención, detección y respuesta para predecir un comportamiento peligroso, eliminar amenazas automáticamente, integrando capacidades de respuesta ante cualquier evento,

y adaptándose a los ataques que más aparecen, realizando un monitoreo continuo sin disminuir la capacidad de procesamiento de la máquina donde se realice. [38]

Uno de los problemas financieros más relevantes es el fraude financiero, siendo Sift Science una empresa que decidió proponer una solución a este problema mediante *Machine Learning*. Debido a la cantidad de información que se debe analizar en la parte financiera como infraestructura crítica, Sift Science ha optado por un *Machine Learning* a gran escala, recolectando información de patrones de fraudes en tiempo real y realizando reglas gracias a señales de fraude específicas por acciones realizadas. Pero todo esto no sería de ayuda si los estafadores logran ocultar su rastro, por lo que Sift Science ha logrado que, gracias al *Machine Learning*, se pueden encontrar más señales de fraude, incluyendo que actos de evasión como la desactivación de JavaScript en algunos sitios o el cambio de IP para que esta aparezca como aquella a la que hayan robado, sean indicados como señales sospechosas. [39]

Como podemos ver a través de todos los ejemplos tanto teóricos e implementados, el *Machine Learning* ha abarcado una gran cantidad de problemas no solo en la medicina e infraestructura sino también dentro del área de seguridad, revolucionando los conceptos pasados y ayudando a tomar mejores decisiones haciendo pruebas con los diferentes algoritmos. Uno de los temas que más ha abordado el *Machine Learning* con respecto a la seguridad es el tema de malware ya sea en archivos, redes, servidores e inclusive en celulares, demostrando que cada vez más los algoritmos de aprendizaje son necesarios para el desarrollo de problemas no solo actuales sino de gran amenaza. A través de la investigación hecha podemos deducir que uno de los temas en los cuales se necesita más desarrollo en el área de *Machine Learning* son las amenazas que presentan ya daños. Siendo más específicos, muchos de los algoritmos detectan y prevén las

amenazas pero cuando una amenaza ya infringió algún tipo de daño a activos como archivos, servidores, redes, entre otros, aún no se ha contemplado bien cómo podrían estos algoritmos ayudar en esto, y sería un tema de profundo interés que podría ser analizado para futuros proyectos.

Desarrollo del Proyecto

1. Investigación y Pruebas Básicas

Según vimos, el sistema operativo Android es uno de los que más recibe atención para la aplicación de Machine Learning, haciendo uso de Android Studio para la realización de aplicaciones. Uno de los ejemplos más relevantes de la unión de ambos es SeraphimDroid, creado por Nikola Milošević quién también generó los 10 riesgos en móviles como un proyecto afín a SeraphimDroid [23]. Como herramienta, es una aplicación creada bajo Android Studio, de código abierto y bajo constante actualización gracias a una comunidad internacional de voluntarios que realizan actualizaciones y nuevas funciones para revisión antes de ser agregadas directamente al programa oficial[22]. Adicionalmente, su creador ha decidido aportar un nuevo marco educativo donde puede ayudar a fomentar las buenas prácticas en cuanto a la seguridad de datos gracias a diversos artículos añadidos sobre este tema [24]. Actualmente la aplicación cuenta con diferentes características pensadas para el cumplimiento de sus dos objetivos como es el "Permission Scanner" que permite revisar los permisos de las aplicaciones instaladas y predecir, bajo uso de Machine Learning, si la aplicación podría resultar siendo un virus, troyano, gusano, rootkit u otro tipo de amenaza, para ser notificada al usuario posteriormente[22].

Para utilizar Machine Learning en la aplicación, se utilizó la librería de Weka que permitió utilizar el algoritmo de Support Vector Machine ya implementado en la herramienta. La herramienta Weka es un conjunto completo de bibliotecas de clases Java que implementan muchos algoritmos avanzados de aprendizaje automático y minería de datos. [2] Esto nos permitió identificar la librería de Weka para Java y aplicarla en otro contexto distinto a SeraphimDroid y Android, con la ayuda del software NetBeans. Dentro de este archivo, se

ejecutaron 4 algoritmos como lo son J48, PART, Decision Table y OneR. Para cada clasificador se realizó un grupo de predicciones, con la ayuda de un grupo de datos usados para entrenamiento y otro para pruebas, todos estos datos obtenidos del conjunto de datos previamente almacenados en formato ARFF. Para saber el porcentaje de aciertos de cada predicción en el vector donde se guardaban, se comparó con los porcentajes ya predichos y si coinciden, se podía considerar un acierto. Finalmente, de este último dato, se definió la exactitud a mostrar por algoritmo definido.

J48	94%
PART	90.67%
Decision Table	92.67%
OneR	96%

Tabla 2. Resultados de diferentes algoritmos sobre "iris.arff", un dataset que provee Weka.

Pero, hasta el momento, no se tenía ninguna prueba sobre eventos de seguridad que hubieran sucedido en un ambiente real, por lo que se utilizó la base de datos de eventos que generó OSSIM, un sistema de gestión de eventos e información de seguridad perteneciente a AlienVault, para utilizarla dentro de Weka con el fin de conocer los resultados que arrojaría una herramienta como Weka sobre estos datos. Siendo que OSSIM tiene la capacidad de exportar estos eventos a un formato CSV, y Weka la función para convertir este tipo de archivos en ARFF para su análisis, se tomaron los eventos de baja y media prioridad de 3 días, debido a que hasta el momento de la exportación de esta información, no se encontraron eventos de mayor prioridad. Pero primero, se debió crear un programa en el lenguaje Python que modifica algunos aspectos del archivo exportado para facilitar su lectura y conversión a ARFF, y después ejecutar su importación a Weka. Por último, se ejecutaron dos herramientas para el análisis de malware llamadas Cuckoo Sandbox y CuckooML dentro de dos máquinas físicas para su ejecución.

A. Instalación y Configuración de Cuckoo Sandbox.

a. Requerimientos del Sistema

Los requerimientos mínimos para la instalación de Cuckoo Sandbox son:

Sistema Operativo	Ubuntu 16.02
Memoria RAM	2 GB (para Virtualización)
Disco Duro Disponible	40 GB
CPU	Quad-Core

b. Proceso de Instalación

*Al iniciar la instalación verifique que está en ROOT con el comando "sudo su" después de esto ponga la clave de administrador.

*Durante la instalación varios softwares dirán la opción de Yes/No después de correr ciertos comandos en todos daremos "Y" y presionamos enter.

*Para los archivos descargados de internet descomprimir con la instrucción "tar -xvzf nombredelarchivo"(Sin las comillas).

1. Instale Linux como sistema operativo principal, para este caso se usó Ubuntu 16.04.02.
2. Corra el comando "sudo apt-get update" para actualizar la distribución de Linux.

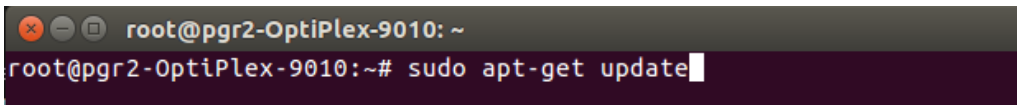

Ilustración 1 Comando actualización de distribución

3. Librerías de Python
    a. Instale las dependencias "sudo apt-get install python python-pip python-dev libffi-dev libssl-dev"

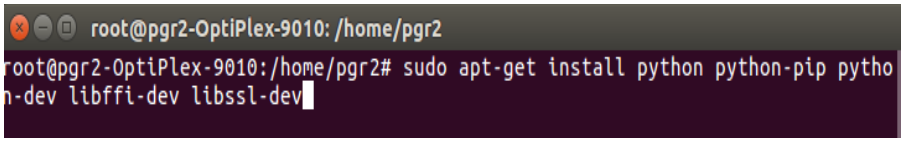

Ilustración 2 Instalación de dependecias

b. Instale “libxml2-dev” y “libxslt-dev” con el comando “sudo apt-get install libxml2-dev libxslt-dev”

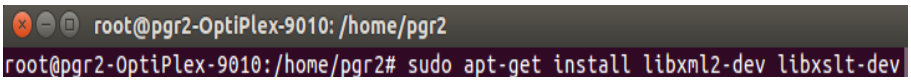

Ilustración 3 Instalación de librerías

4. Descargue y actualice “pip” con el comando “sudo -H pip install --upgrade pip”

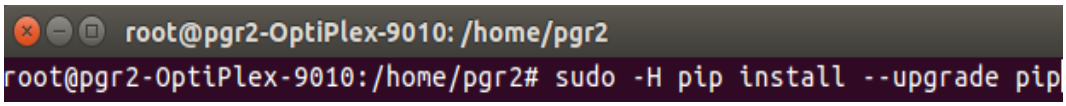

Ilustración 4 Actualización librería PIP

5. Descargue Cuckoo Sandbox “pip install -U cuckoo”

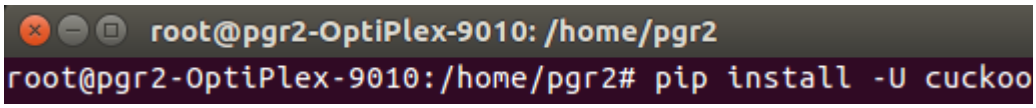

Ilustración 5 Comando de Instalación de Cuckoo

6. Descargue un iso Windows 7

7. Descargue e instale Virtualbox “sudo apt-get install virtualbox-5.2”

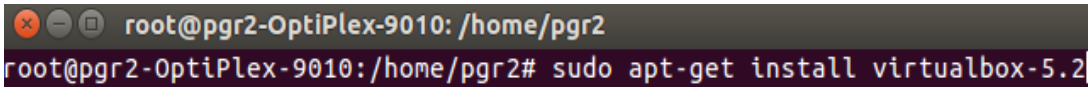

Ilustración 6 Instalación VirtualBox

8. Corra virtualbox con el comando “sudo virtualbox”

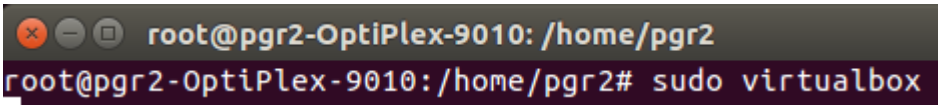

Ilustración 7 Ejecución de Máquina Virtual

9. Cree una máquina virtual con la iso antes descargada con las siguientes especificaciones

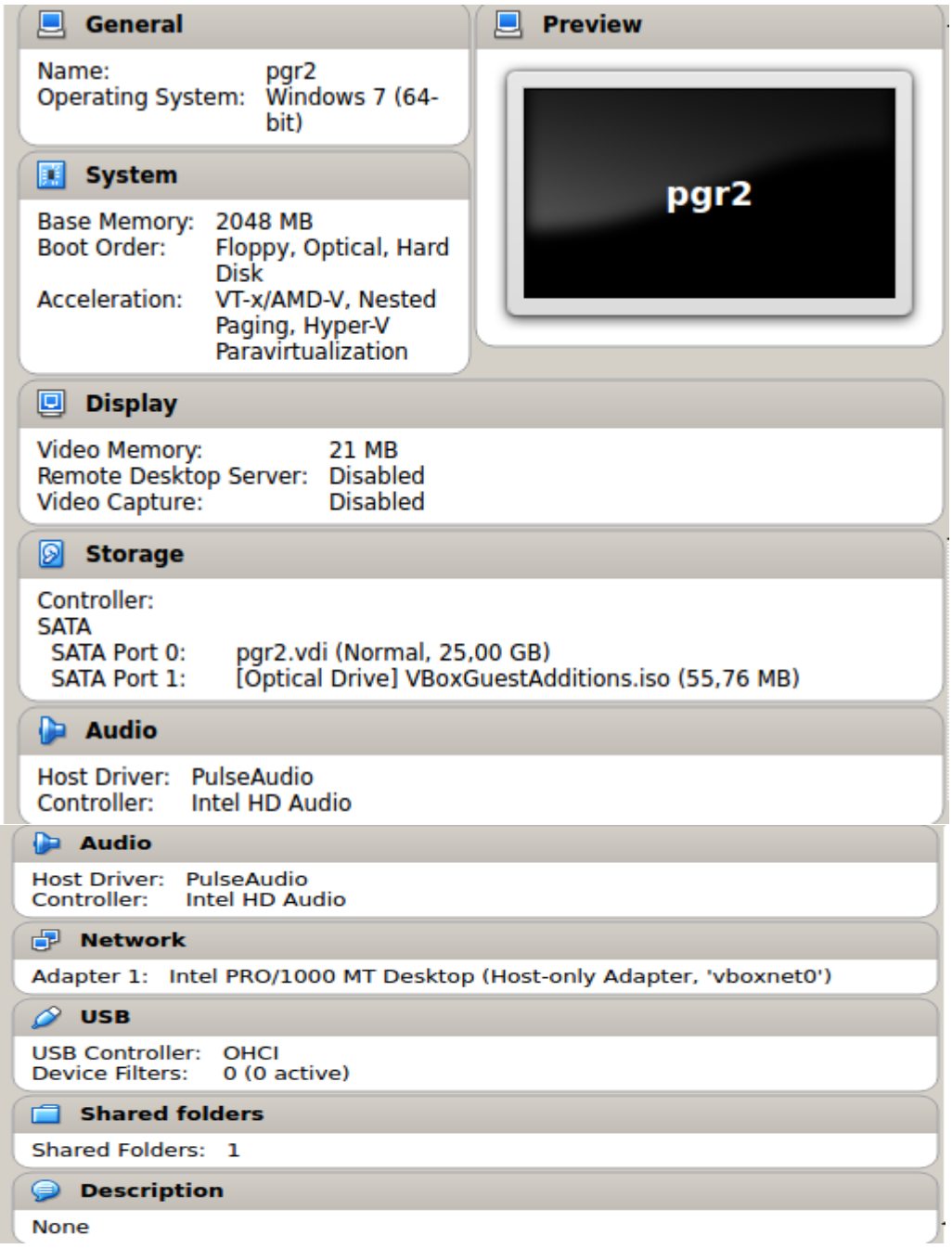

Ilustración 8 Características de Guest OS

a. Siga los pasos de instalación básico de Windows 7

10. Instalación de base de datos MongoDB con el comando “sudo apt-get install mongodb”

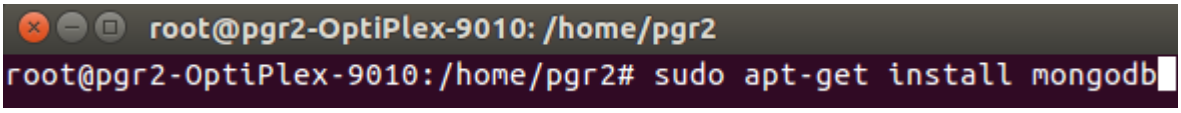

Ilustración 9 Comando de instalación MongoDB

11. Instale TCPdump

a. Ejecute el siguiente comando “sudo apt-get install tcpdump”

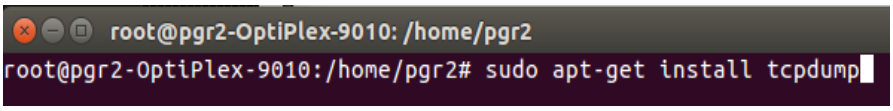

Ilustración 10 Comando de Instalación tcpdump

b. Ejecute el siguiente comando “sudo apt-get install libcap2-bin”

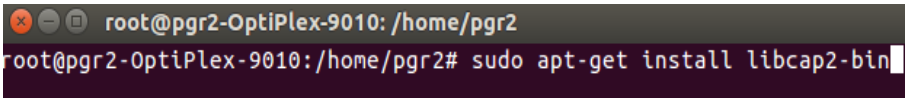

Ilustración 11 Comando instalación librería tcpdump

c. Ejecute el siguiente comando “sudo setcap cap_net_raw,cap_ _net_admin=eip /usr/sbin/tcpdump”

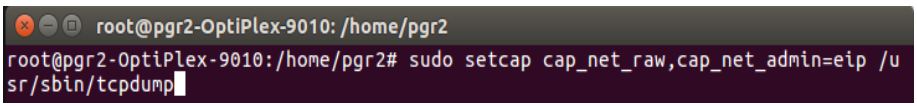

Ilustración 12 Ejecución de comando 1

d. Ejecute el siguiente comando “getcap /usr/sbin/tcpdump”

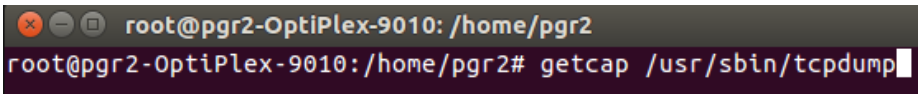

Ilustración 13 Ejecución de comando 2

12. Instale la Volatilidad

a. sudo apt install git

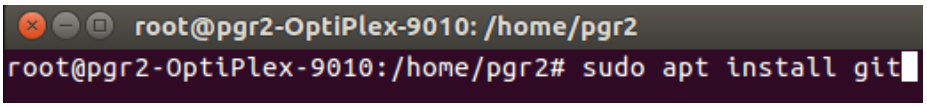

Ilustración 14 Instalación librería GIT

b. git clone https://github.com/volatilityfoundation/volatility.git

c. Navegue a la carpeta de volatilidad y ejecute el siguiente comando "sudo python setup.py install"

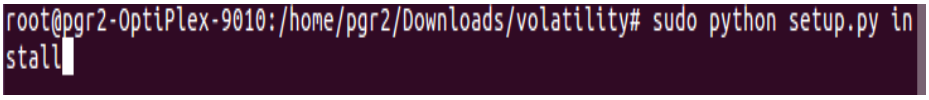

Ilustración 15 Comando para instalación de Volatilidad

13. Instale los plug-ins de volatilidad

a. De la página https://github.com/gdabah/distorm/releases descargue Distorm "Source code (tar.gz)"

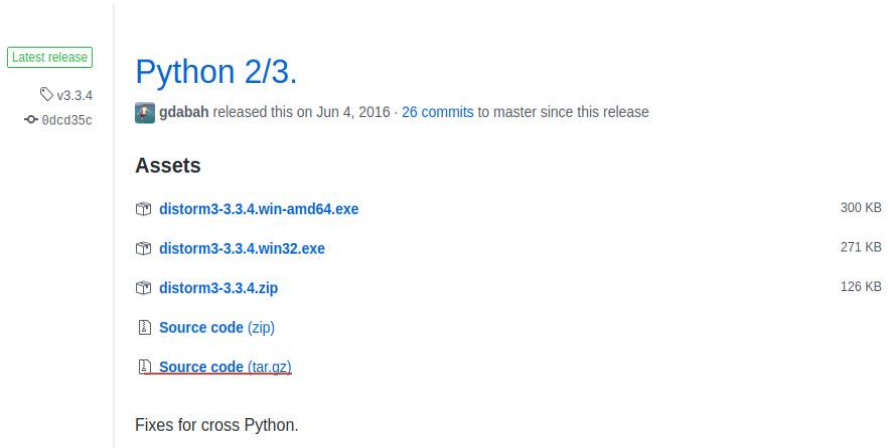

Ilustración 16 Página de descarga Distorm

b. Navegue hasta la carpeta distorm y ejecute el comando "sudo python setup.py install"

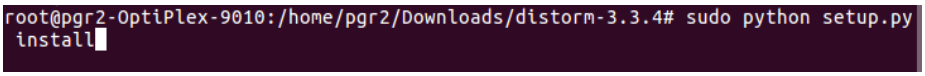

Ilustración 17 Instalación Distorm

3. Instalar Yara

a. Instale autoreconf

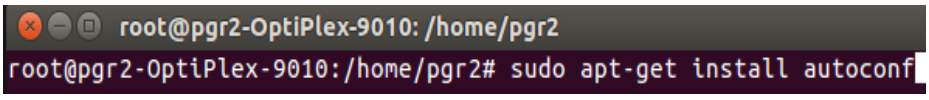

Ilustración 18 Comando de instalación Autoreconf

b. Instale libtool-bin

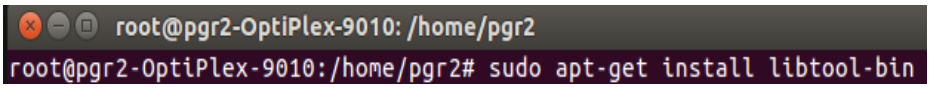

Ilustración 19 Instalación librería Libtool-bin

c. Descargue Yara de la siguiente página (tar)

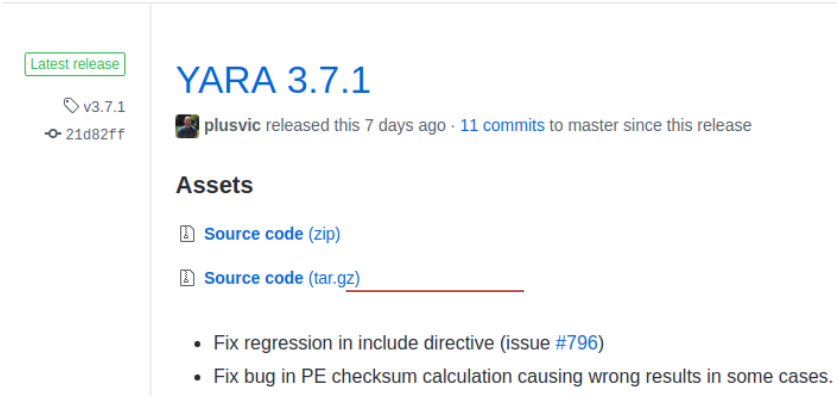

Ilustración 20 Página de instalador Yara

d. Navegue hasta la carpeta yara y ejecute "./bootstrap.sh" seguido "./configure --with-crypto --enable-magic --enable-cuckoo" seguido "make" seguido "sudo make install" seguido "sudo -H pip install yara-python"

15. Instalar PyCrypto

a. Descargue PyCrypto de la siguiente página https://pypi.python.org/pypi/pycrypto

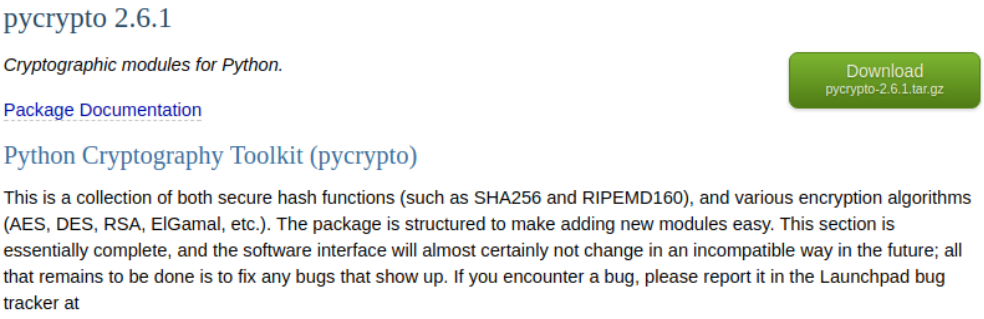

Ilustración 21 Página de descarga Pycrypto

b. Navegue hasta la carpeta pycrypto y ejecute el siguiente comando "python setup.py build" y después "sudo python setup.py install"

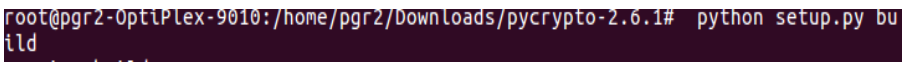

Ilustración 22 Instalación Pycrypto

16. Instale Openpyxl con el siguiente comando “sudo -H pip install openpyxl”

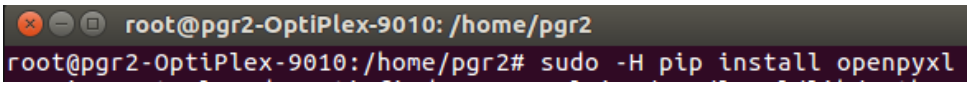

Ilustración 23 Instalación de Openpyxl

17. Instale UJSON

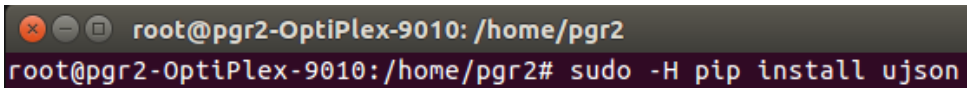

Ilustración 24 Instalación UJSON

18. Instale IPython

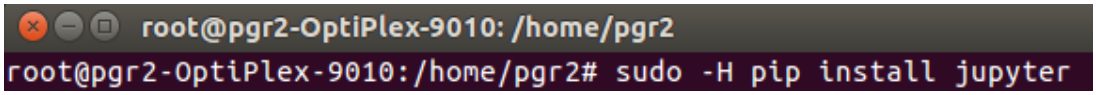

Ilustración 25 Instalación de Jupyter/IPython

19. Instalar Mitmproxy

a. Ejecute “sudo apt-get install python3-pip python3-dev libssl-dev libtiff5-dev libjpeg8-dev zlib1g-dev libwebp-dev”

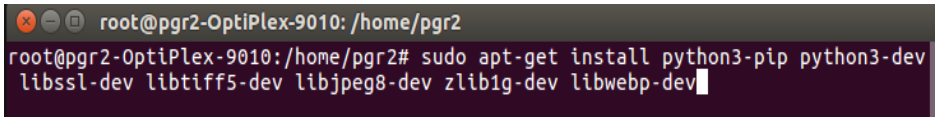

Ilustración 26 Instalación librerías Mitmproxy

b. Ejecute “sudo pip3 install mitmproxy” seguido de “mitmproxy” seguido de
“cd ~/.mitmproxy” seguido “cp mitmproxy-ca-cert.p12 /home/YourUserName/Downloads/cuckoo/analyzer/windows/bin/cert.p12 ” y por último “mitmdump = /usr/local/bin/mitmdump”

20. Ahora vamos a configurar los archivos de Cuckoo en el siguiente path

Ilustración 27 Archivos de configuración Cuckoo Sandbox

- cuckoo.conf
  Ejecute “nano cuckoo.conf” en la consola
  [cuckoo]

memory_dump = on
machinery = virtualbox o vmware

[resultserver]
ip = dirección ip de la máquina host(Ubuntu).

- auxiliary.conf
  Ejecute “nano auxiliary.conf” en la consola:
  [mitm]
  enable = yes
  [sniffer]
  interface = Interfaz de red de la máquina virtual
- vmware.conf
  Ejecute “nano vmware.conf” en la consola:
  [vmware]
  machines = name of virtual machine
  interface = name of the network interface for the virtual machine
  [Nombre_de_la_máquina_virtual]
  vmx_path = ../Nombre_de_la_maquina virtual/
  Nombre_de_la_maquina virtual.vmx
  ip = Direccion ip de la máquina virtual
- processing.conf
  Ejecute “nano processing.conf” en la consola:
  [memory]
  enable = yes
- memory.conf
  Ejecute “nano memory.conf” en la consola:
  [basic]
  guest_profile = Nombre del guest (Profile según el sistema operativo)

A continuación se deja una lista de los sistemas operativos Windows

```
$ python vol.py --info

VistaSP0x64                 - A Profile for Windows Vista SP0 x64
VistaSP0x86                 - A Profile for Windows Vista SP0 x86
VistaSP1x64                 - A Profile for Windows Vista SP1 x64
VistaSP1x86                 - A Profile for Windows Vista SP1 x86
VistaSP2x64                 - A Profile for Windows Vista SP2 x64
VistaSP2x86                 - A Profile for Windows Vista SP2 x86
Win10x64                    - A Profile for Windows 10 x64
Win10x86                    - A Profile for Windows 10 x86
Win2003SP0x86               - A Profile for Windows 2003 SP0 x86
Win2003SP1x64               - A Profile for Windows 2003 SP1 x64
Win2003SP1x86               - A Profile for Windows 2003 SP1 x86
Win2003SP2x64               - A Profile for Windows 2003 SP2 x64
Win2003SP2x86               - A Profile for Windows 2003 SP2 x86
Win2008R2SP0x64             - A Profile for Windows 2008 R2 SP0 x64
Win2008R2SP1x64             - A Profile for Windows 2008 R2 SP1 x64
Win2008SP1x64               - A Profile for Windows 2008 SP1 x64
Win2008SP1x86               - A Profile for Windows 2008 SP1 x86
Win2008SP2x64               - A Profile for Windows 2008 SP2 x64
Win2008SP2x86               - A Profile for Windows 2008 SP2 x86
Win2012R2x64                - A Profile for Windows Server 2012 R2 x64
Win2012x64                  - A Profile for Windows Server 2012 x64
Win7SP0x64                  - A Profile for Windows 7 SP0 x64
Win7SP0x86                  - A Profile for Windows 7 SP0 x86
Win7SP1x64                  - A Profile for Windows 7 SP1 x64
Win7SP1x86                  - A Profile for Windows 7 SP1 x86
Win81U1x64                  - A Profile for Windows 8.1 Update 1 x64
Win81U1x86                  - A Profile for Windows 8.1 Update 1 x86
Win8SP0x64                  - A Profile for Windows 8 x64
Win8SP0x86                  - A Profile for Windows 8 x86
Win8SP1x64                  - A Profile for Windows 8.1 x64
Win8SP1x86                  - A Profile for Windows 8.1 x86
WinXPSP1x64                 - A Profile for Windows XP SP1 x64
WinXPSP2x64                 - A Profile for Windows XP SP2 x64
WinXPSP2x86                 - A Profile for Windows XP SP2 x86
WinXPSP3x86                 - A Profile for Windows XP SP3 x86
```

- reporting.conf
  Ejecute “nano reporting.conf” en la consola:
  [reporthtml]
  enable = yes
  [mongodb]
  enable = yes

21. Configuración de la máquina guest Windows

a. Deshabilite el Firewall de Windows
b. Deshabilite el Defender de Windows
c. Instale python 2.7.13 en la página https://www.python.org/downloads/release/python-2713/
d. Transfiera mediante una carpeta compartida el archivo de Cuckoo “Agent” y ejecútelo
e. Descargue pafish desde esta página https://github.com/a0rtega/pafish y ejecútelo
f. Ahora se realizará un Snapshot de la máquina virtual (*Importante antes de realizar alguna prueba verificar que la conexión entre el guest y el host sea HostOnly y siempre se trabaje sobre el Snapshot*)

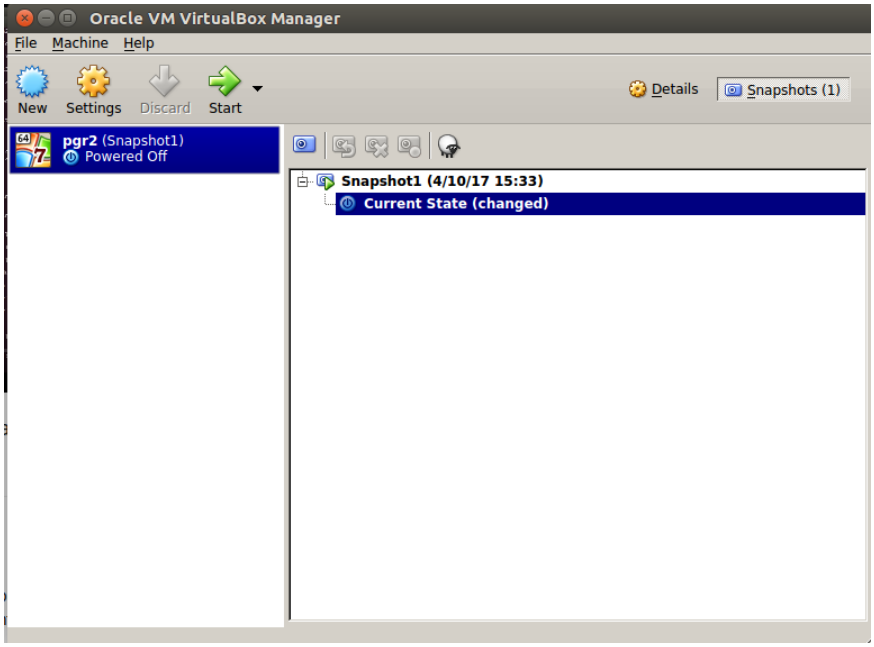

Ilustración 28 Visualización VirtualBox con Snapshot

22. Configuración de la red

- Se asigna a la máquina virtual creada anteriormente una interfaz de red Host-Only

```
vboxmanage modifyvm MaquinaVirtual --hostonlyadapter1
vboxnet0
vboxmanage modifyvm MaquinaVirtual --nic1 hostonly
```

- Para que la máquina virtual pueda tener acceso virtual se añaden reglas iptables. Estos comandos harán que esto sea posible.

```
iptables -A FORWARD -o eth0 -i vboxnet0 -s 192.168.56.0/24 -
m conntrack --ctstate NEW -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate
ESTABLISHED,RELATED -j ACCEPT
iptables -A POSTROUTING -t nat -j MASQUERADE
```

- Para permitir el "Ip forwarding" en el host, se aplica el siguiente comando con permisos de root:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

23. Ejecución de Cuckoo

a. Proxy

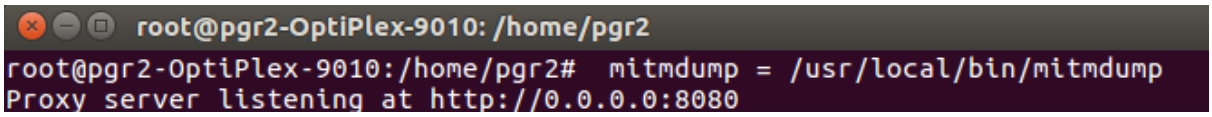

Ilustración 29 Inicialización del Proxy

b. Haga un Snapshot de la máquina virtual

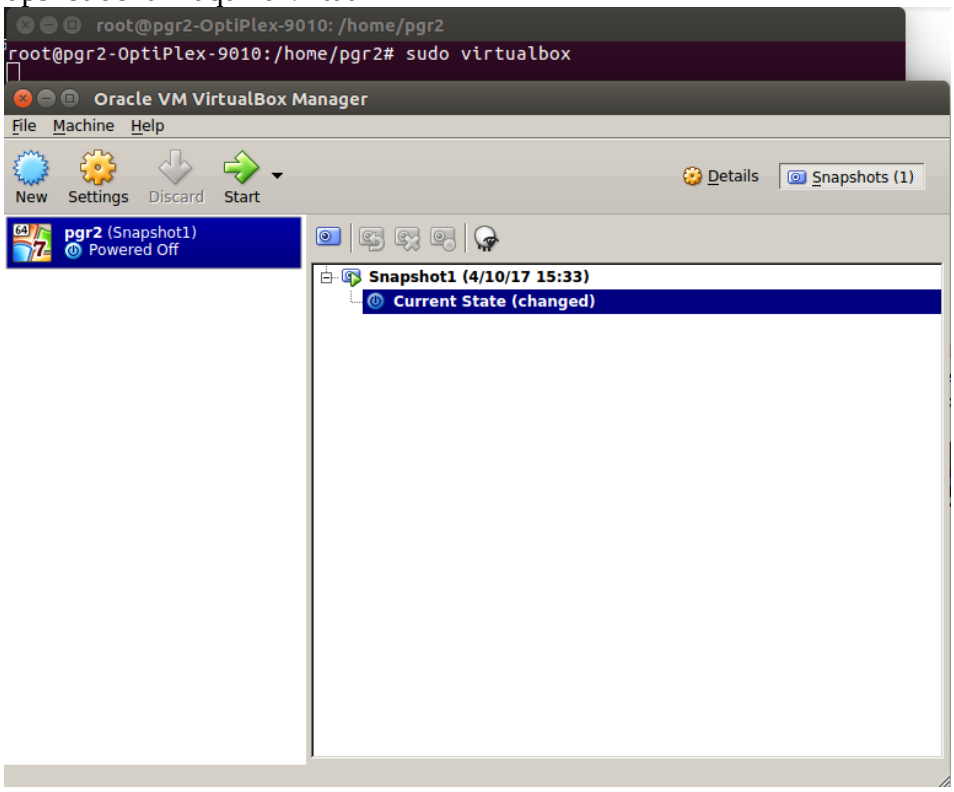

Ilustración 30 Creación de Snapshot Máquina Virtual

c. Verifique que la máquina virtual este corriendo sin ningún problema

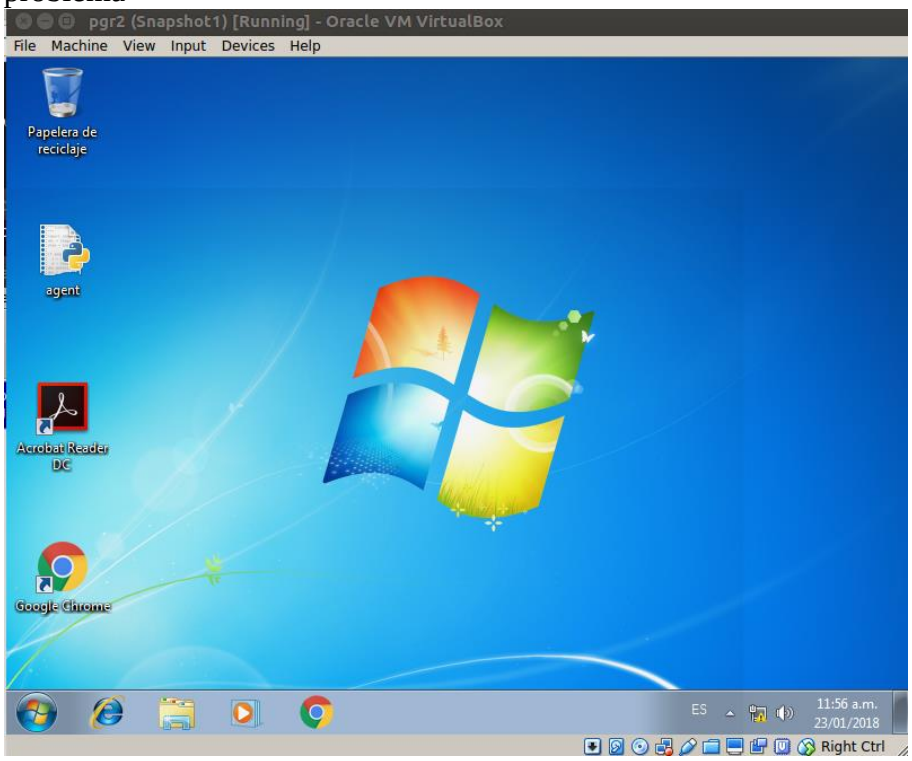

Ilustración 31 Máquina virtual en ejecución

d. Corra Cuckoo con el comando "sudo cuckoo" verifique que Cuckoo corra sin ningún problema y haya cargado correctamente la imagen como se logra visualizar en la imagen

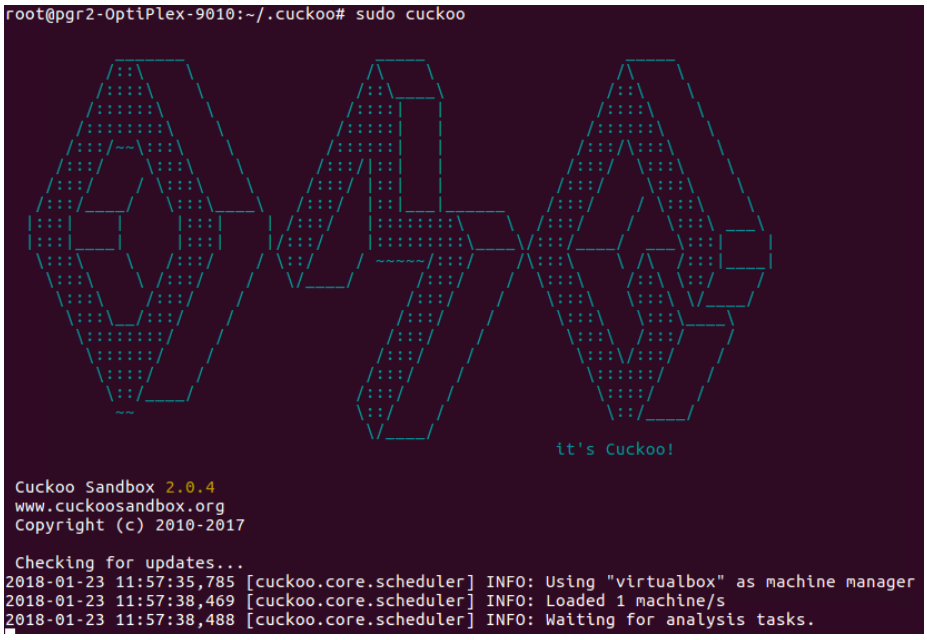

Ilustración 32 Cuckoo Sandbox en ejecución

e. Dentro de la carpeta de Cuckoo corra el siguiente comando “cuckoo web runserver” para habilitar el servidor

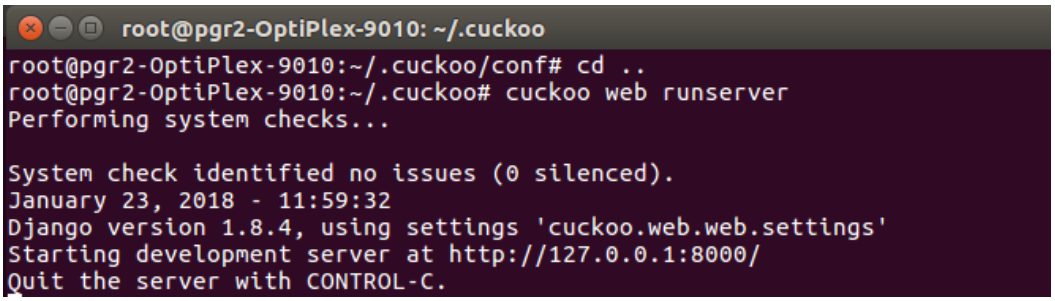

Ilustración 33 Ejecución interfaz gráfica Cuckoo Sandbox

f. En el navegador, corra el LocalHost con el puerto 8000 y verifique que la pagina cargue completamente y de click en “SUBMIT A FILE FOR ANALYSIS”

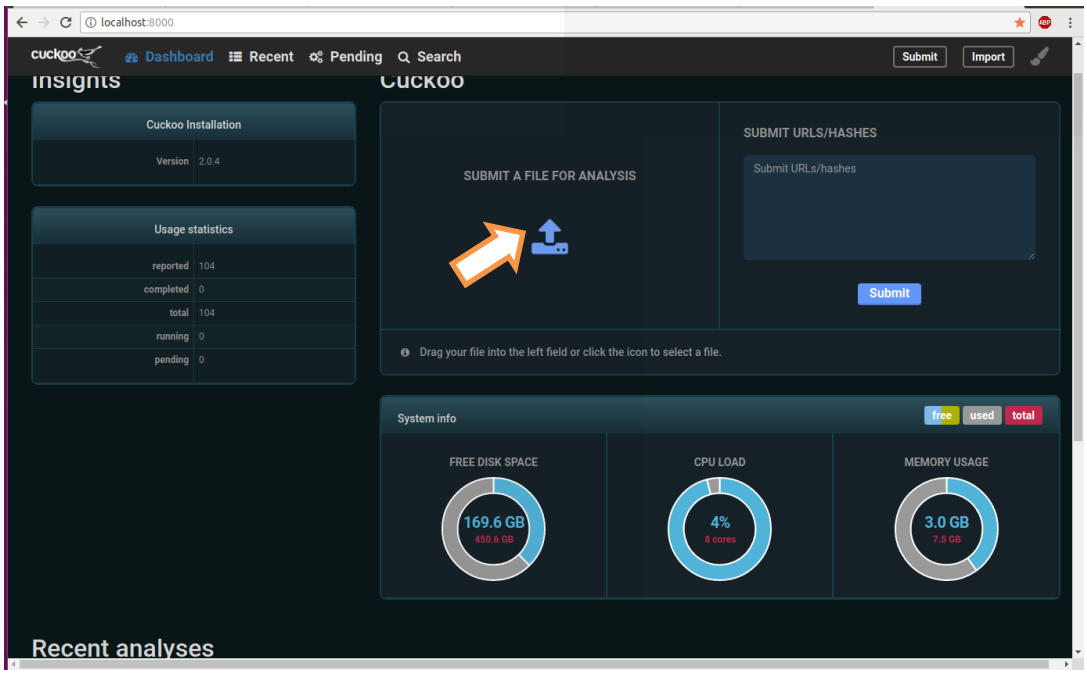

Ilustración 34 Página de análisis Cuckoo Sandbox

g. Descargue de Undercode(https://malwares.underc0de.org) o TheZoo(https://github.com/ytisf/theZoo) un malware para ser analizado en esta caso se escogerá “Locky”

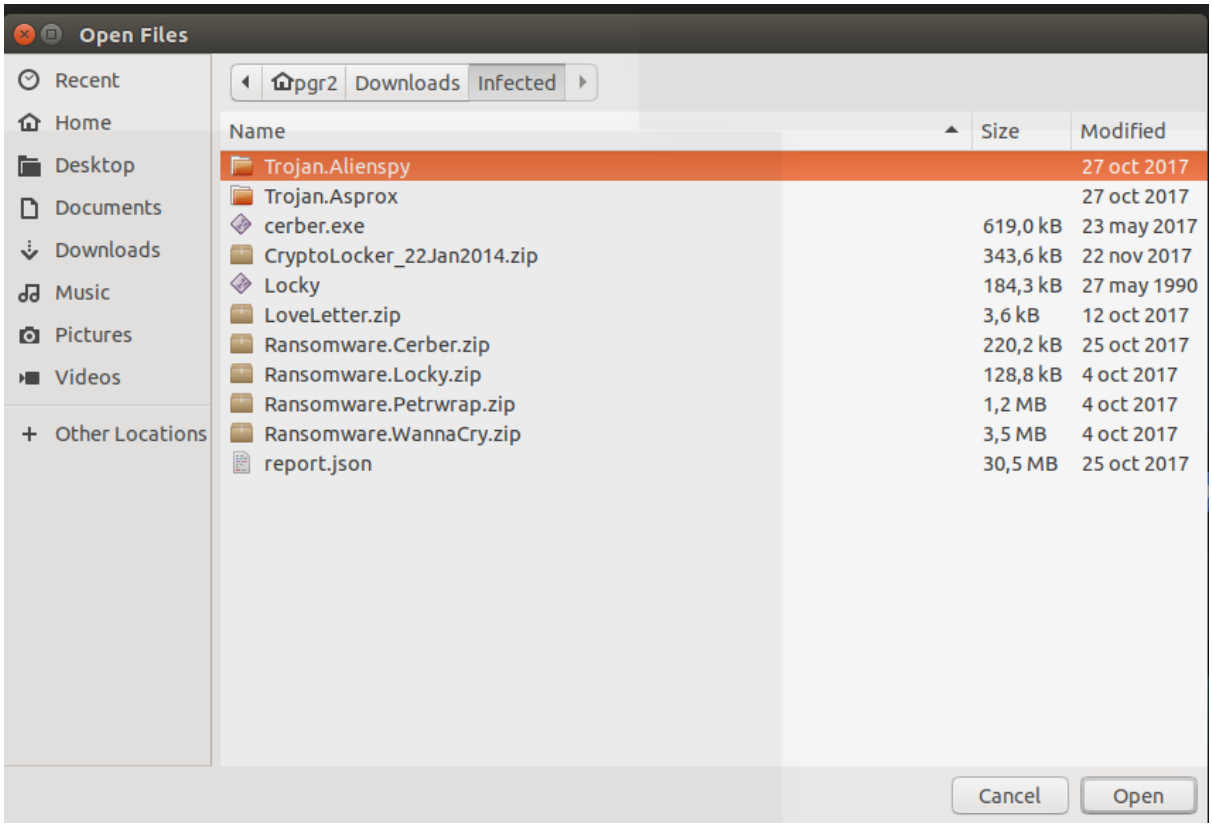

Ilustración 35 Elección de archivo para prueba

h. Una vez seleccionado verifique que la siguiente página salga

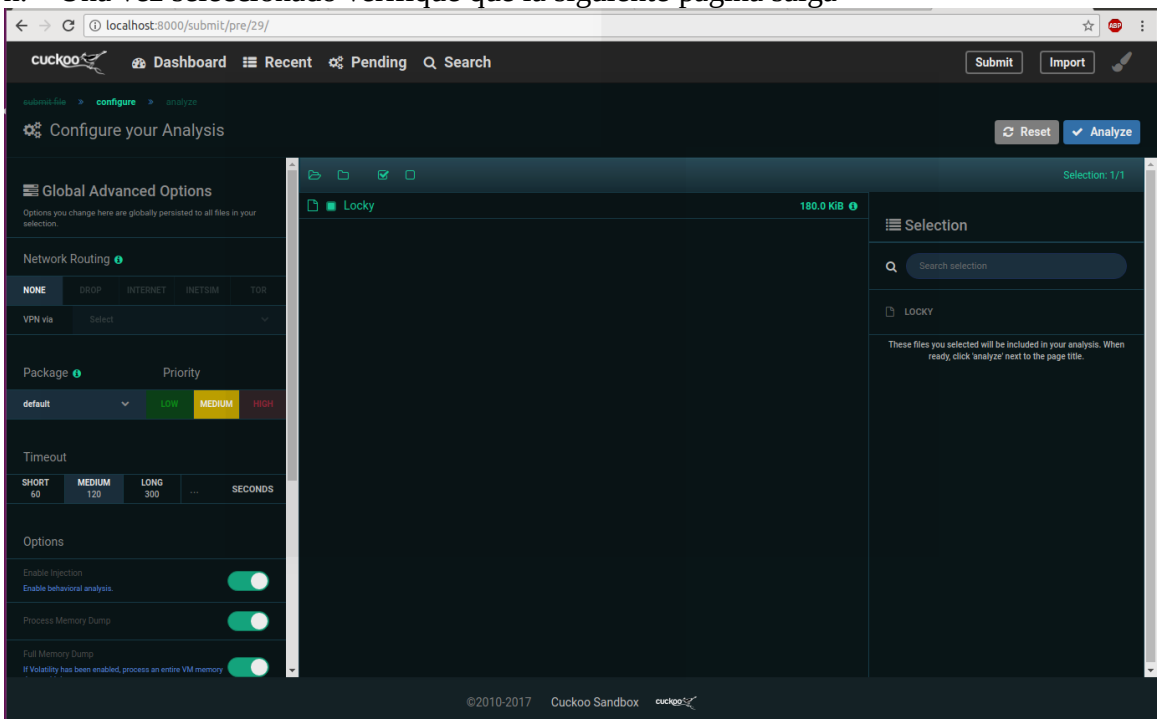

Ilustración 36 Configuración de análisis

i. Verifique que en opciones quede tal cual se presenta en la siguiente imagen y también que en “Machine” reconozca el nombre de la máquina virtual y presione Analyze

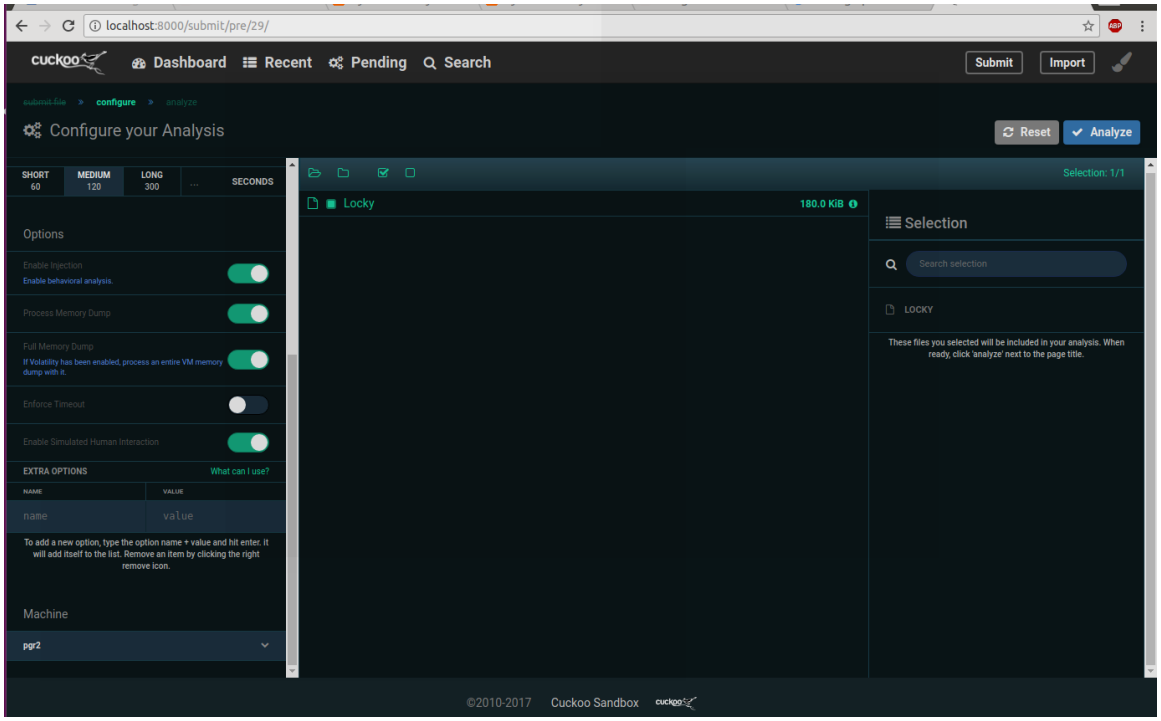

Ilustración 37 Configuración para análisis 2

j. Verifique que la imagen de la máquina virtual se restaure

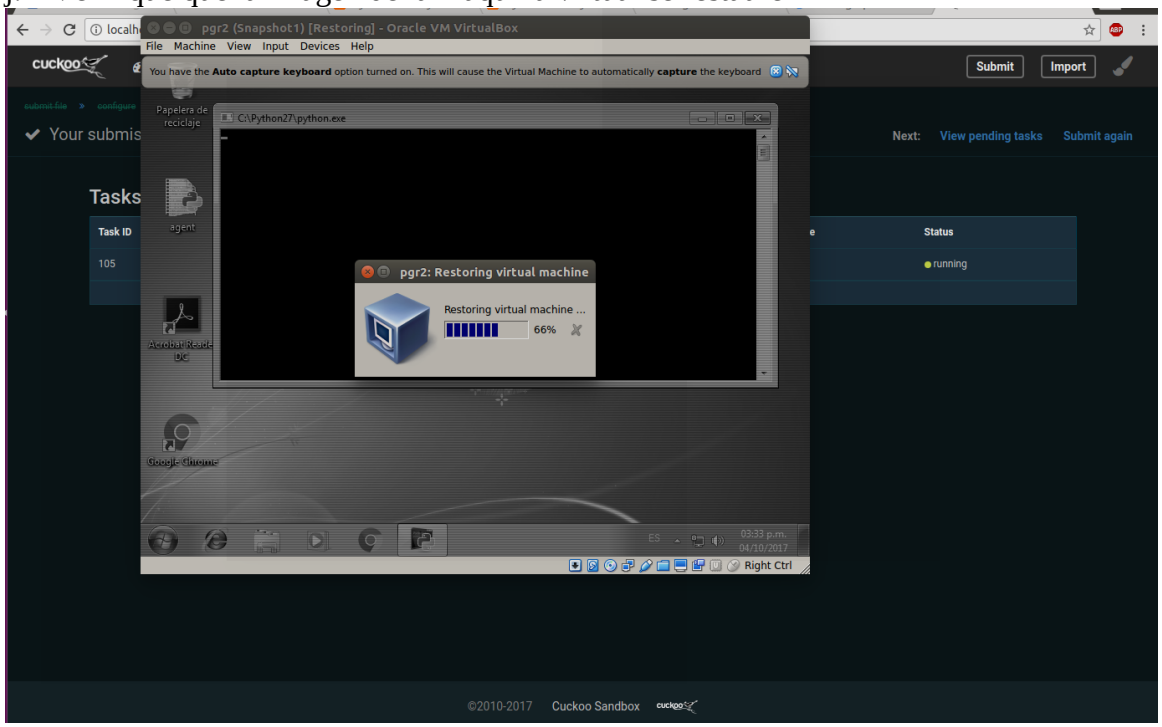

Ilustración 38 Restauración de máquina virtual

k. Después del anterior paso el malware que se ejecutó debería estar corriendo

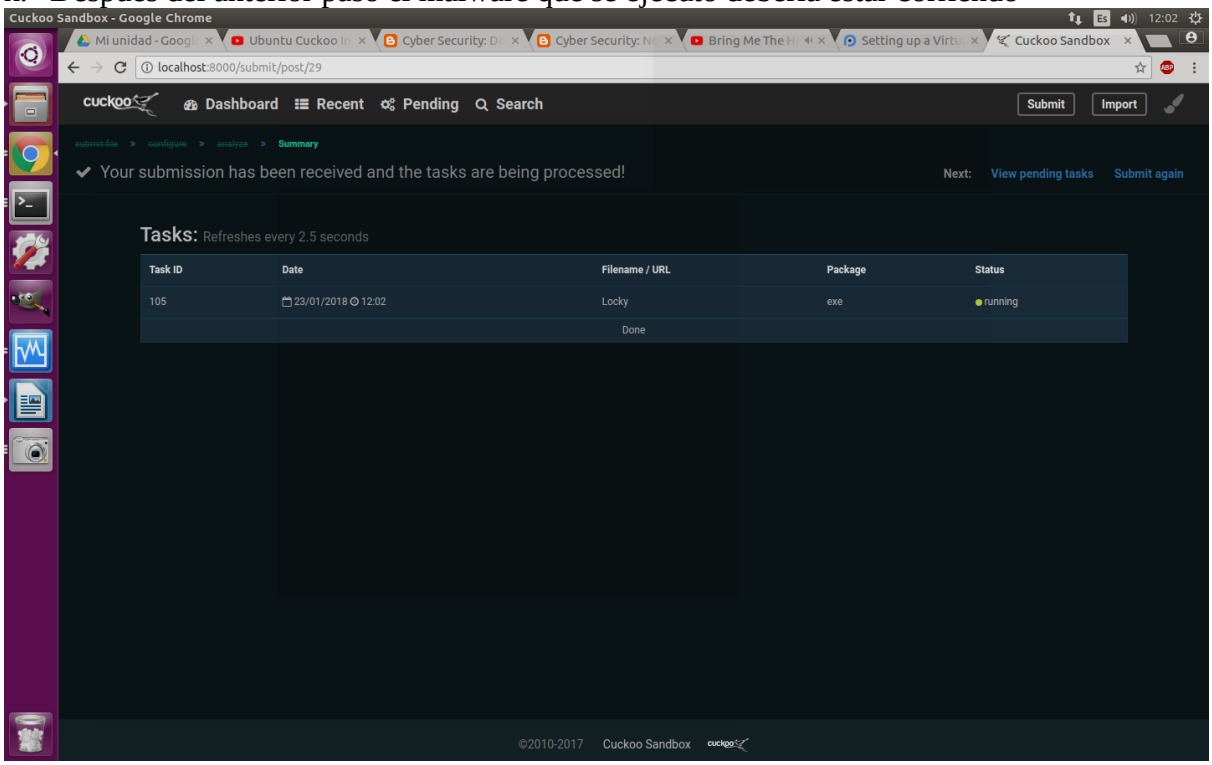

Ilustración 39 Estado de tareas

l. Verifique en la consola donde se está corriendo Cuckoo el "Agent" este corriendo y en el servidor que esté recibiendo peticiones get y post

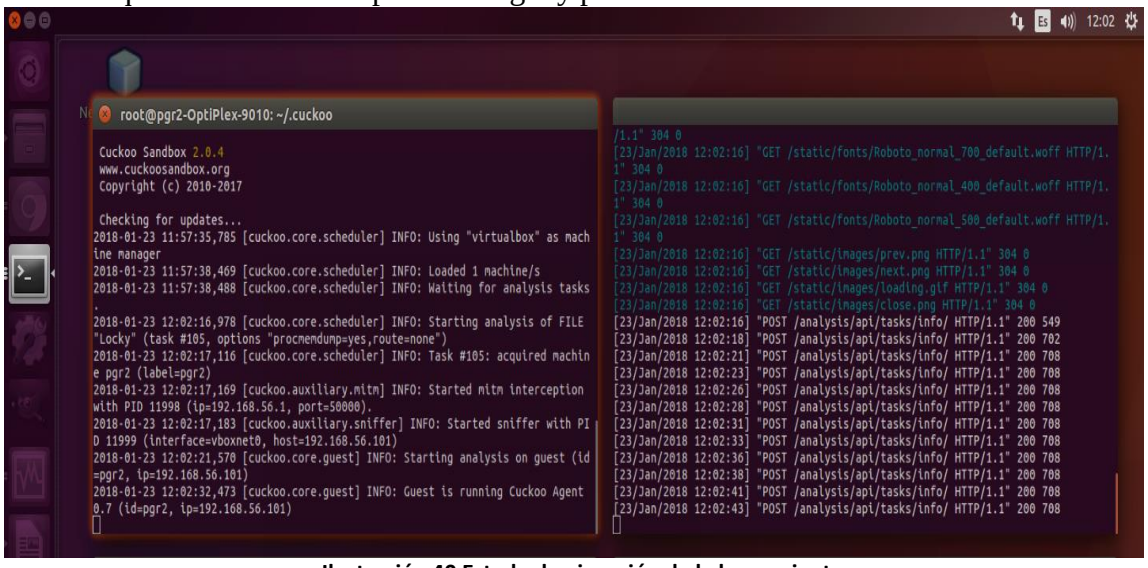

Ilustración 40 Estado de ejecución de la herramienta

m. En la pestaña de reciente verifique que el malware ingresado haya sido reportado

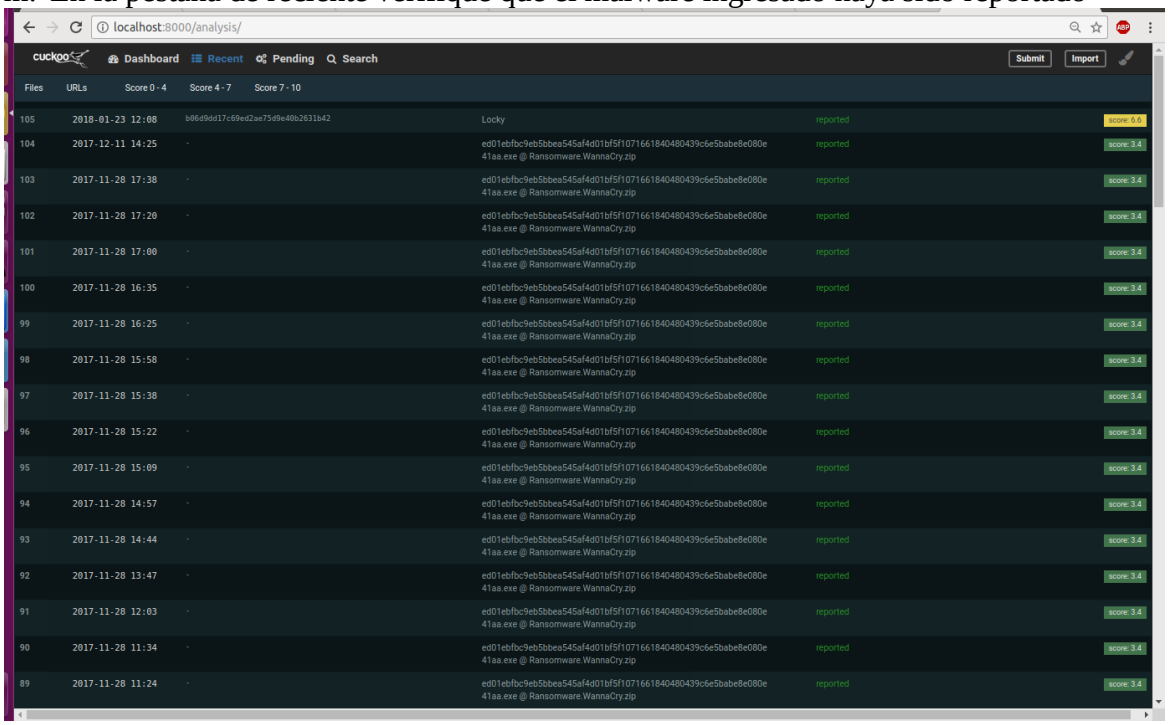

Ilustración 41 Página de reportes y puntuación de archivos

n. Por último verifique que el reporte este completo

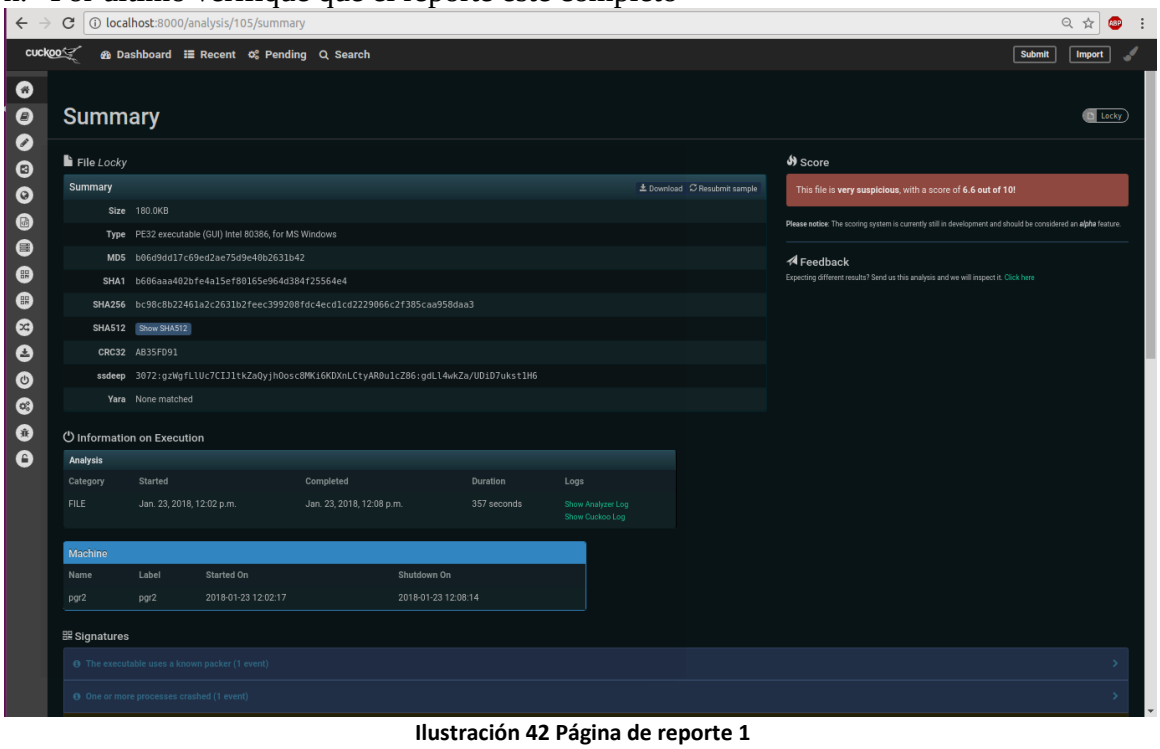

Ilustración 42 Página de reporte 1

o.

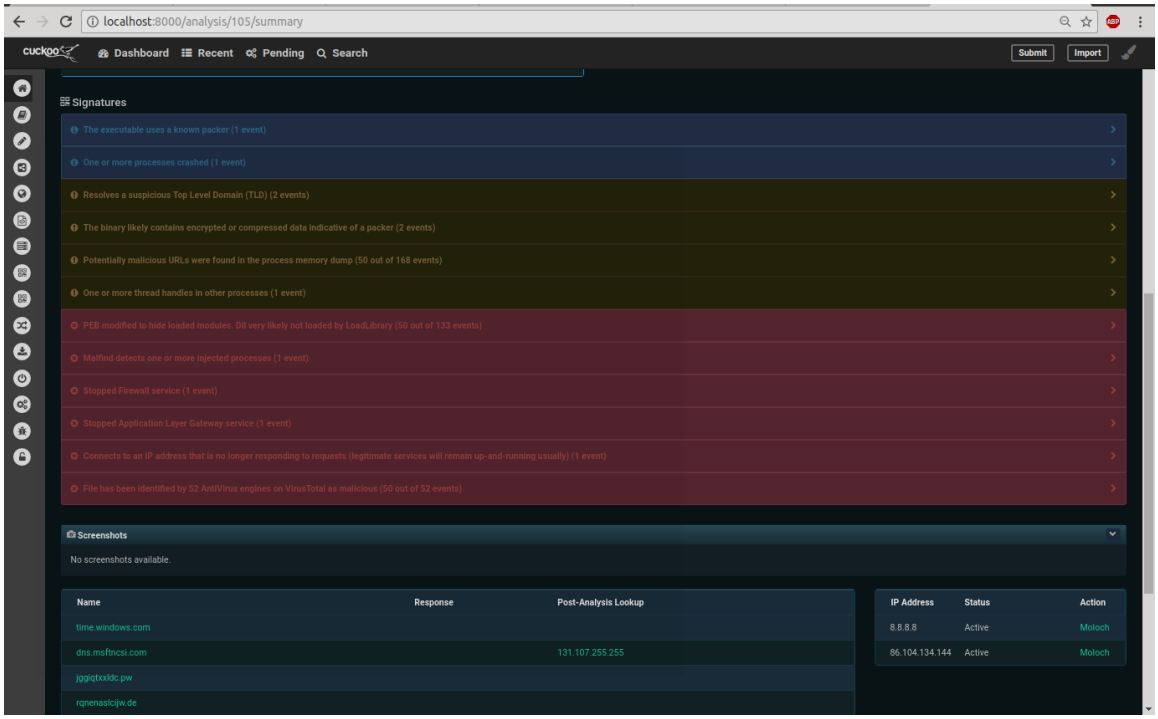

Ilustración 43 Página de reporte 2

p.

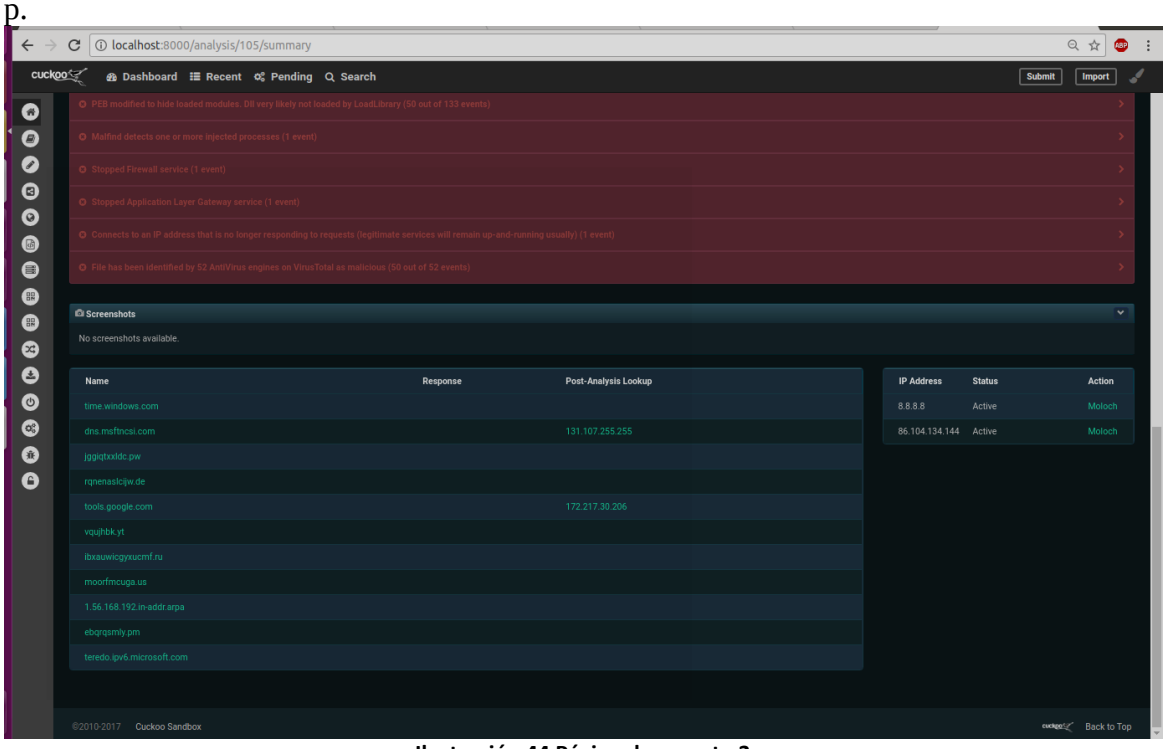

Ilustración 44 Página de reporte 3

B. Instalación y Configuración de Weka.

a. Requerimientos del Sistema

El requerimiento mínimo para instalar Weka es Java. A continuación se muestra la versión mínima necesaria de Java que permitiría ejecutar una versión de Weka correctamente.

		Java				
		1.4	**1.5**	**1.6**	**1.7**	**1.8**
WEKA	**<3.4.0**	X	X	X	X	X
	3.4.x	X	X	X	X	X
	3.5.x	3.5.0-3.5.2	>3.5.2	X	X	X
	3.6.x		X	X	X	X
	3.7.x		3.7.0	>3.7.0	>3.7.13	X
	3.8.x				3.8.0-3.8.1	>3.8.1
	3.9.x				3.9.0-3.9.1	>3.9.1

b. Proceso de Instalación

Para comenzar con Weka, primero debe dirigirse a la página de la Universidad de Waikato (https://www.cs.waikato.ac.nz/ml/weka/), donde aparecerá la siguiente página sobre Weka. Se elige la opción "Download" en la lista "Getting Started".

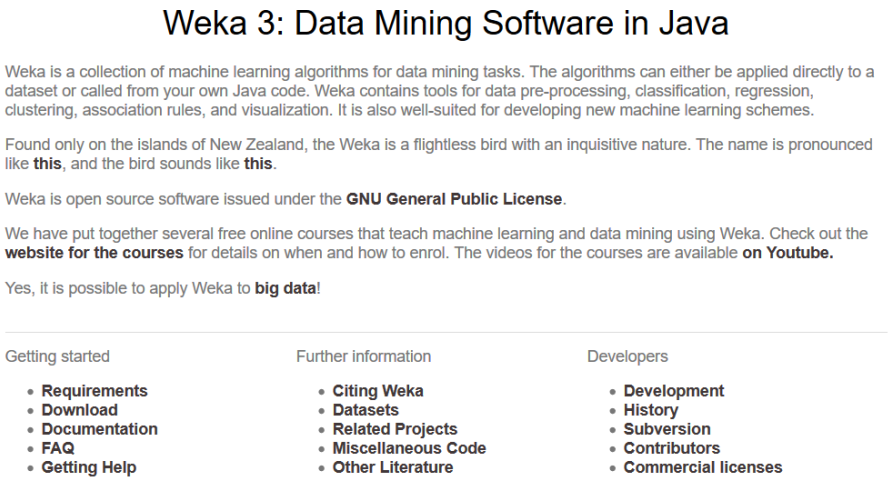

Ilustración 45 Página principal Weka

Cuando la siguiente página aparezca, se podrá elegir un ejecutable para la instalación de la herramienta. La misma cubre diferentes sistemas operativos como Windows, Mac OSX y Linux. En esta ocasión, se descargará el ejecutable para 64 bits de Windows sin Java.

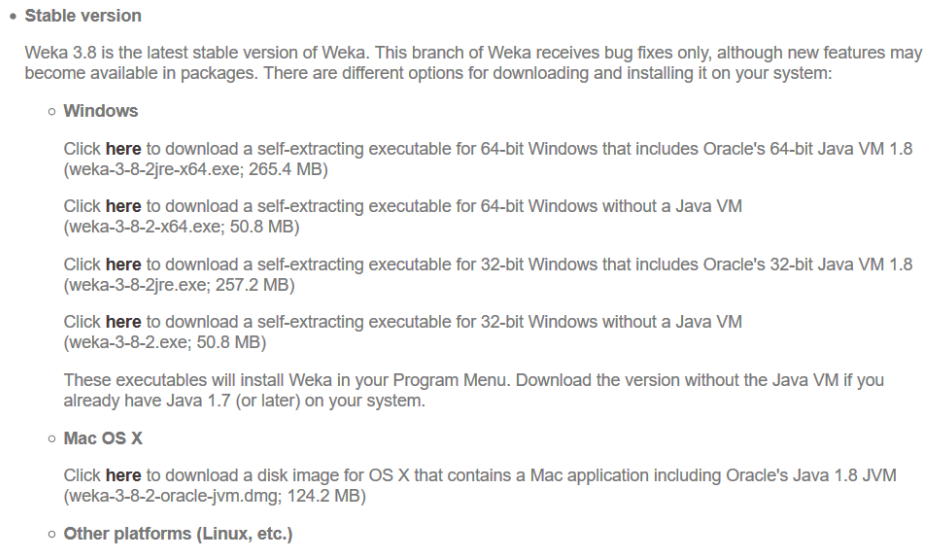

Ilustración 46 Archivos estables de instalación

En cuanto finalice la descarga, procederemos a dar doble clic sobre el archivo para ejecutarlo y abrir la siguiente ventana, se oprime el botón “Next”.

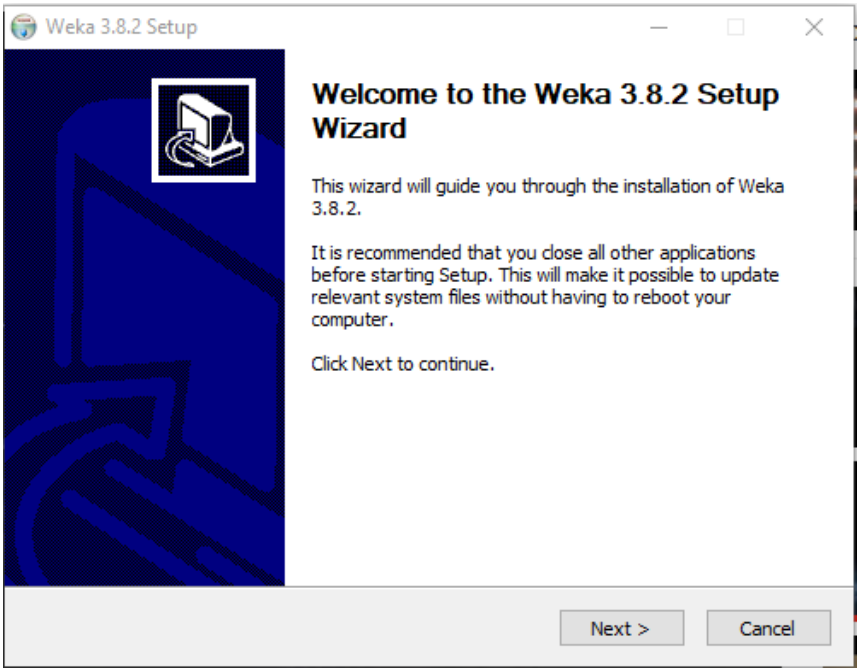

Ilustración 47 Instalador Weka

Se acepta la licencia de la herramienta, que es de tipo GNU. Y a continuación, se elige el tipo de instalación a realizar. Para evitar que falte algún archivo importante para la demostración, se elige la instalación “Full”.

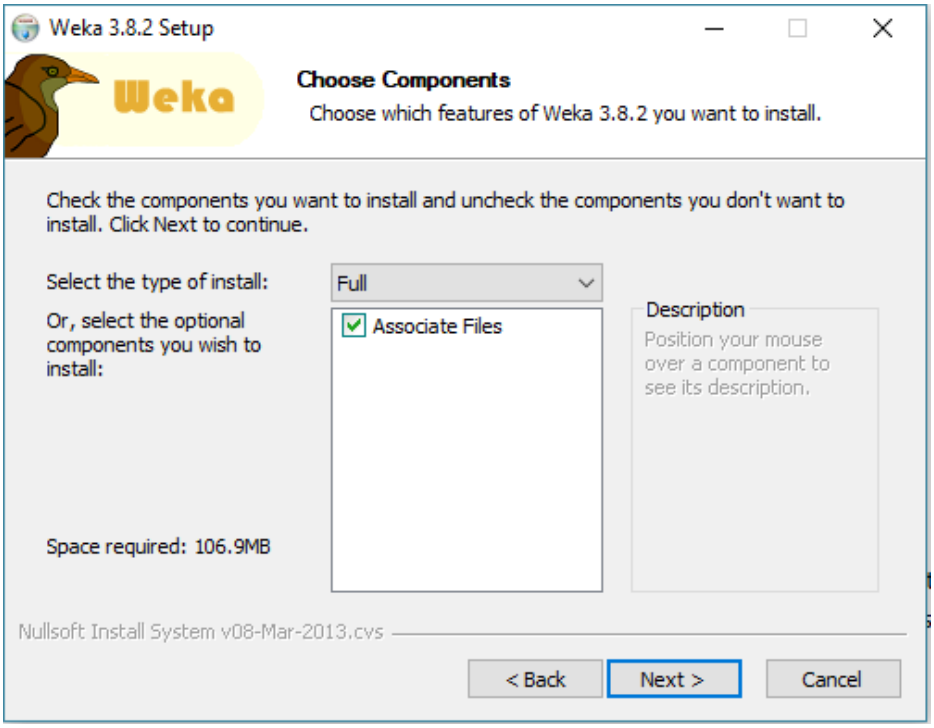

Ilustración 48 Elección de componentes Weka

La ruta de instalación puede ser modificada, pero para evitar perder más adelante esta ubicación, se deja predeterminada por el programa. Se oprime el botón “Next” para pasar a la ventana de accesos directos y ahora “Install” para comenzar. Al terminar, se podrá oprimir “Next” y nos llevará a una nueva ventana para abrir la aplicación.

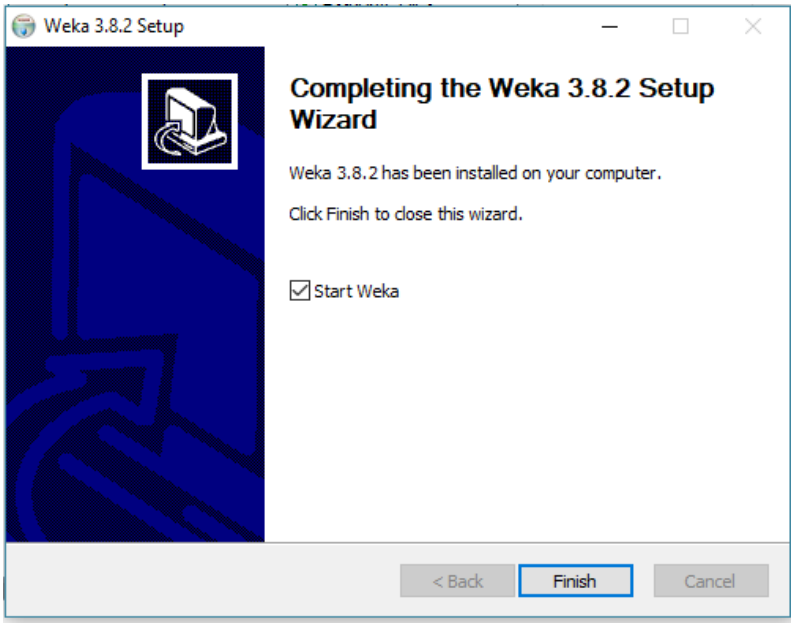

Ilustración 49 Finalización instalación Weka

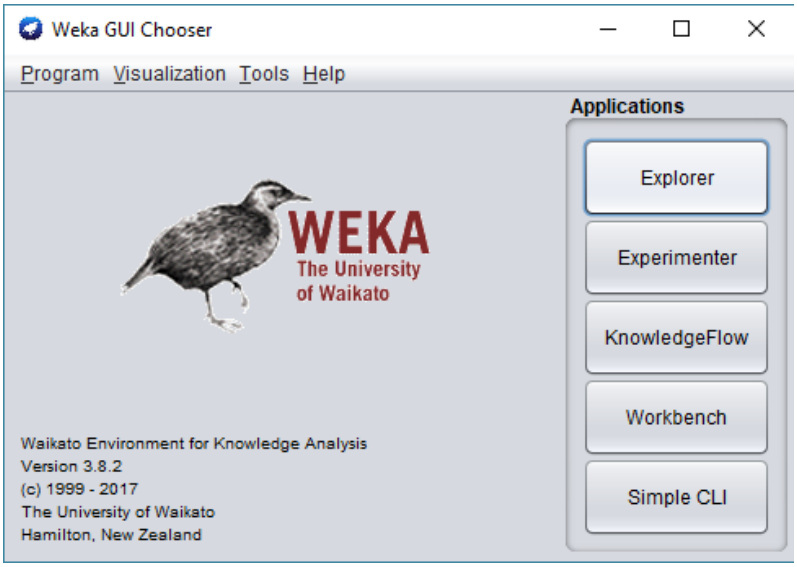

Ilustración 50 Ventana principal Weka

c. Demostración

Para la demostración, se elige la opción "Explorer" de la pantalla principal de Weka. Que nos llevará a una nueva ventana. En esta, oprimiremos la opción "Open file" que nos permitirá buscar un conjunto de datos en formato ARFF o añadir uno propio. Para la demostración, se usará un conjunto de datos predeterminado de Weka llamado "iris.arff".

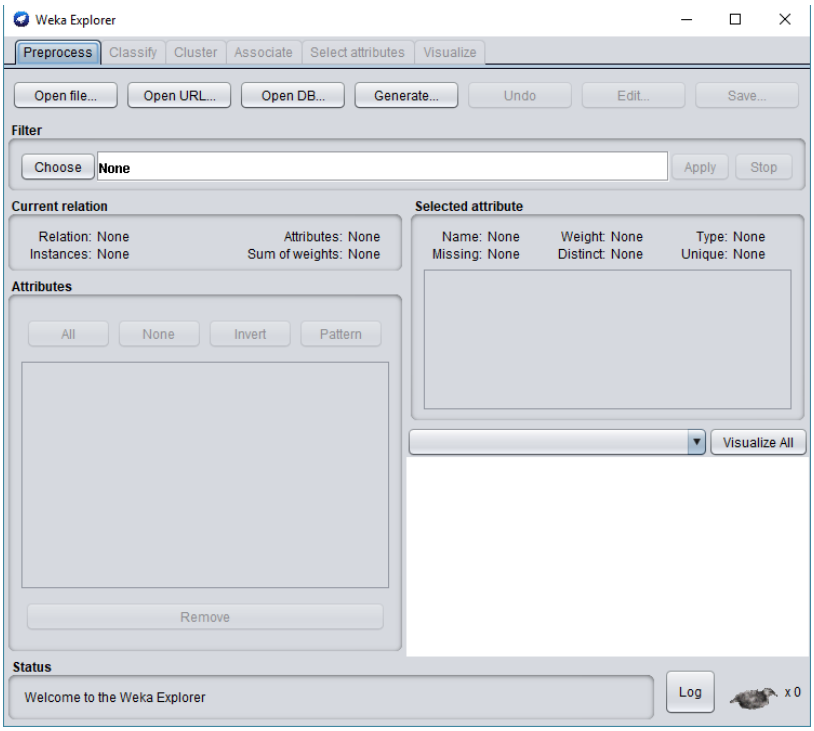

Ilustración 51 Weka Explorer

Al abrir el conjunto de datos, la ventana Explorer cambiará, mostrando la información de los datos ingresados.

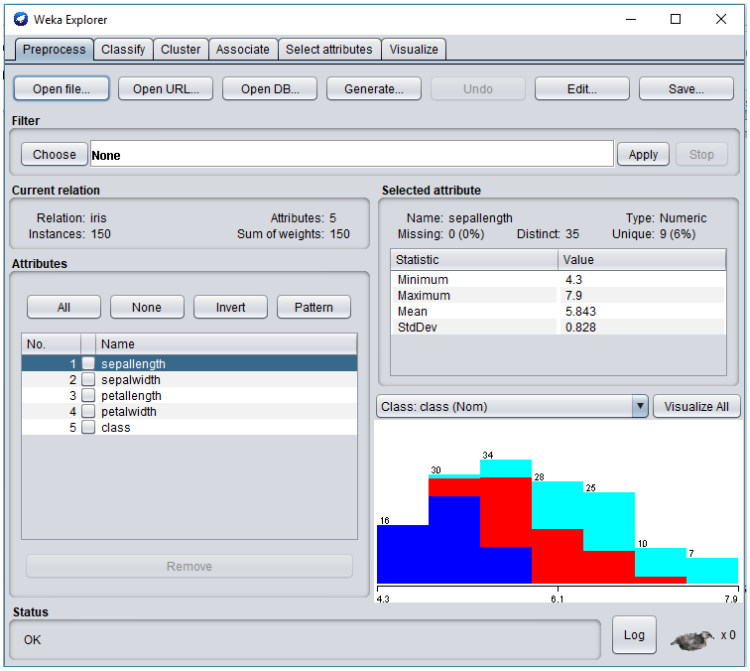

Ilustración 52 Weka Explorer - Iris.arff

Con los datos actuales, se usará un algoritmo de clasificación J48, que es un algoritmo de árbol como se puede ver en la parte lateral del menú de elección.

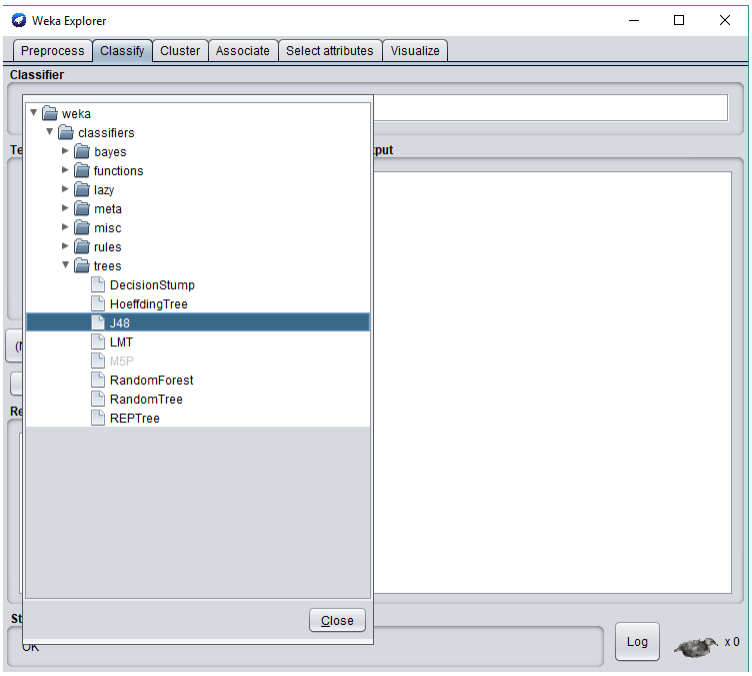

Ilustración 53 Elección de algoritmo

Se deja la configuración predeterminada para J48, y se ejecuta la prueba para una validación cruzada de 10 hojas, calculando así el porcentaje de aciertos esperados. Después de oprimir Start, la herramienta realiza los cálculos necesarios y los presenta en el espacio en blanco junto a su configuración.

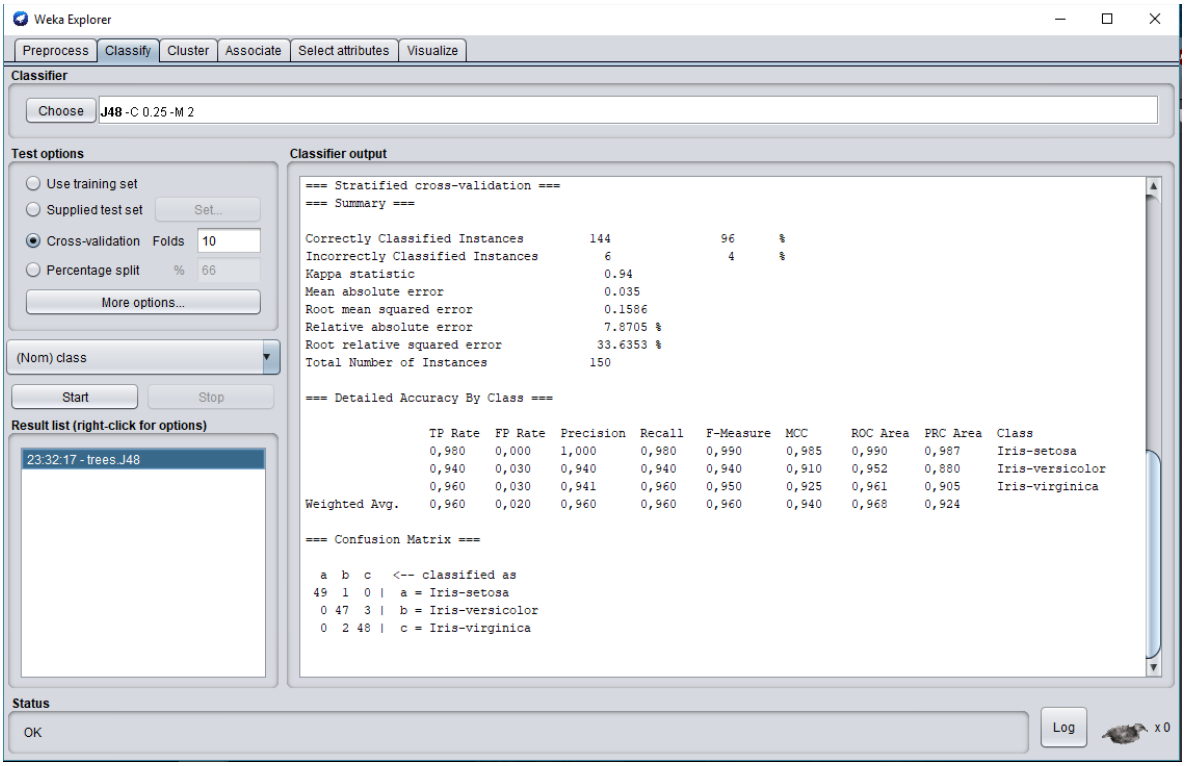

Ilustración 54 Resultados algoritmo J48

Aquí podemos observar que logró clasificar correctamente 144 datos de 150, siendo un 96% de acierto para este algoritmo. En la parte inferior de la pantalla, observaremos la matriz de confusión y cómo el algoritmo los clasificó según su configuración predeterminada.

Para visualizar mejor la clasificación realizada por el algoritmo, podemos dar clic sobre la pestaña Visualize, que nos permitirá ver la forma como están dispersos los datos según los atributos que posee el archivo.

Ilustración 55 Visualización de distribución por atributos

En este caso, veremos que la mejor distribución se encuentra bajo los atributos Petallength y Petalwidth.

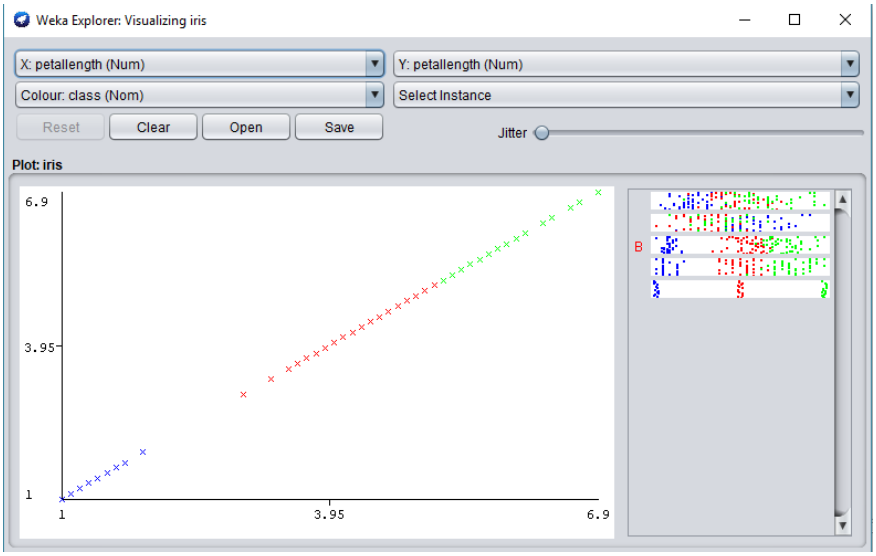

Ilustración 56 Visualización de dos atributos

Como podemos observar en la gráfica, la clasificación 100% acertada sería difícil debido a que los datos de Iris-versicolor e Iris-virginica se encuentran muy cerca y sin una separación demarcada como la separación con Iris-setosa. Esta es una explicación gráfica para el porcentaje faltante de clasificación en la prueba teórica.

2. Resultados de Investigación y Pruebas

Se logró realizar diferentes pruebas con el software Weka para poder comprender más profundamente cómo actuaban los diferentes algoritmos supervisados y no supervisados dentro de la herramienta, entre los cuales se encontraban J48, PART, Decision Table y OneR. Se utilizaron conjuntos de datos que proveía la herramienta, aplicando varios algoritmos y así conseguir diferentes resultados según el algoritmo usado, aprendiendo cuales servían para un conjunto de datos con determinadas características, dependiendo de lo que se deseara realizar con la información. Para mostrar los conocimientos adquiridos en esta herramienta, se realizó una presentación para explicar su uso, como introducción a la herramienta en general.

Después de realizar estas pruebas con Weka, y lograr dar con la librería que nos permitía ejecutar los diferentes algoritmos fuera del propio software, se creó un algoritmo que tomaba los mismos conjuntos de datos que el software, pero realizaba el análisis bajo el lenguaje de programación Java y el software NetBeans. Se consiguió un resultado parecido con ambas pruebas, tanto en el Software Weka como en el programa creado en Java. Conociendo el funcionamiento de Weka, y el formato en que este manejaba los datos necesarios, se ejecutó una nueva prueba con los datos obtenidos de OSSIM, normalizando la información para poder ser leída dentro de Weka y el programa en Java. Aunque se esperaba que el archivo que se extrajera pudiera ser enviado directamente a Weka, se obtuvieron buenos resultados con respecto a pruebas con algoritmos de agrupación.

Debido a que Weka no fue construido como un software específico para el tema de seguridad que se manejó dentro de este artículo, se pasó a la siguiente prueba y resultados con un sistema de

análisis de Malware como lo es Cuckoo Sandbox. Se logró la instalación de Cuckoo Sandbox 2.0.3 en el sistema operativo Ubuntu 16.04 para tener un analizador de malware en ambientes controlados usando Windows 7 como target o guest, logrando analizar diferentes tipos de malware entre ellos Petya y Wannacry un par de los ransomware que más han afectado entre el 2016 y 2017, haciendo un volcado de memoria y viendo los diferentes tipos de DNS a los cuales se intenta conectar; también se logra observar los *signatures,* los cuales nos permiten ver que las acciones que ejecuta este tipo de ransomware en particular[1].

También se preparó un ambiente de trabajo en una segunda máquina física con Ubuntu 16.04 para poder realizar la instalación de CuckooML y realizar un análisis más detallado de la información que el programa pudiera conseguir de los archivos analizados. Debido a que este programa fue realizado basándose en una versión antigua de Cuckoo Sandbox, su instalación se dificultó y no se pudo realizar correctamente un análisis con el mismo.

3. Elección y creación del algoritmo de Aprendizaje Automático

A. Búsqueda y Selección de Información

Dada la información de las familias de malware se investigó diferentes bases de datos que contuviera malware de forma segura para el análisis de forma abierta y recopilación accesible y segura. Con base en esto se escogió Undercode [47] y TheZoo [48] por su amplia gama de malware tantos recientes como antiguos. Una vez obtenidos los ejecutables de los malware, se escogieron tres grandes familias de malware: Botnet, Troyanos y Crypters, debido a su importancia e impacto a través de los años.

Después de la revisión de los antecedentes del apoyo del aprendizaje automático en la seguridad informática, se logró encontrar que los algoritmos más usados en este ámbito

fueron KMeans, un algoritmo de clusterización, y Red Neuronal Artificial, como algoritmo de clasificación. Después de pruebas con la herramienta Weka, y con un conjunto de datos proveídos por la misma, se decidió continuar con la Red Neuronal Artificial, pues su porcentaje de acierto con la mayoría de los datos fue mayor al 97%.

B. Extracción de Información

Una vez obtenidos los ejecutables, se procede a analizarlos en el Sandbox Cuckoo. Después de esto, Cuckoo entrega un reporte con diferentes segmentos de los cuales nos enfocaremos en "Summary" e "Information on Execution"; en el primero, veremos las reglas de Yara para identificar descripciones de malware y profundizar en el análisis del malware, en el segundo nos enfocaremos en las "Signatures" o comportamientos del malware, los cuales se identificaron cuando el malware se ejecutó en el ambiente controlado.

Una vez el análisis está completo se procede a extraer la información del archivo en formato Json generado por Cuckoo. Después de analizar la estructura, y mediante un programa realizado en Java, se obtienen todas las signatures en tres diccionarios organizados por severidad. Cabe destacar que Cuckoo separa en tres segmentos la severidad de los comportamientos localizados: los azules o leves tienen un valor de uno, los amarillos o medios tienen un valor de dos y, por último, los rojos o altos tienen un valor de tres o más. Por último, se extrae todas las signatures de Cuckoo organizadas por severidad de más bajo a más alto en un vector para el proceso de normalización que se explicará más adelante.

C. Normalización de Datos

Para que los datos sean más precisos y puedan usarse en la Red Neuronal Artificial previamente escogida, se realizó un proceso de normalización y selección de datos. En el proceso de normalización, para los tres diccionarios con su severidad, se creó un vector de la misma longitud del vector de "signatures" o comportamientos lleno de ceros y unos, en el cual los "1" eran la posición exacta en donde la llave de los diccionarios coincide con el vector de las "signatures". Con esto obtendremos un vector normalizado con toda la información del comportamiento del malware en la herramienta para ser usado en la Red Neuronal Artificial.

D. Aplicando Aprendizaje Automático (Entrenamiento)

En el proceso de entrenamiento una vez establecido el conjunto de datos normalizados, los cuales fueron 150 malware distribuidos equitativamente entre las tres familias antes mencionadas, el programa empieza a probar varias combinaciones entre el porcentaje de entrenamiento y el número de neuronas en capa oculta, finalizado este proceso entrega un porcentaje de entrenamiento entre un 60 – 80 %, este valor puede variar dependiendo de la cantidad de datos.

La variable que modifica el número de neuronas en la capa oculta se establece entre 4–10 neuronas con el fin de verificar el porcentaje de acierto en la clasificación del entrenamiento de la red.

E. Aplicando Aprendizaje Automático (Clasificador)

Para la prueba de la RNA se usaron 54 malwares diferentes a los usados en el entrenamiento de la red neural dando un 98% de clasificación acertada en las familias de

malware antes expuestas; este porcentaje se mide basado en las salidas las cuales fueron 4, o de manera gráfica {0,0,0,0}. Dependiendo de la posición que tuviera el cambio de valor a 1, sería la clasificación dada por la Red Neuronal Artificial. En nuestro caso, de derecha a izquierda, la clasificación sería Botnet, Troyano, o Crypter. El último espacio es asignado al evento donde la red no logró clasificar el malware.

Como podemos ver, de este 2% de error se puede deducir dos posibles escenarios. En el primero, los datos de prueba no cumplían con las características específicas para las tres familias de malware. En el segundo, es posible que la red necesitara más entrenamiento o información para identificar patrones o comportamientos no definidos dentro de ella.

4. Resultados Aplicación de Algoritmo

Una vez obtenidos los resultados de los 50 archivos maliciosos se logró observar un 98% de acierto en la clasificación de estos malware dentro de las categorías Botnets, Troyanos y Crypters, logrando observar patrones y comportamientos en específicos en cada uno de las familias, logrando crear una base de datos con esta información en la cual podemos ver que eventos son frecuentes en cada clasificación. También se observaron dos casos particulares en los cuales la Red Neuronal Artificial clasificó en dos familias (Botnets y Troyanos) un malware, siendo este un caso particular dentro del conjunto de datos.

Conclusiones

Desde el inicio de la investigación, pudimos encontrar que la inclusión del aprendizaje automático dentro de la seguridad no es un tema nuevo, y que existen varias aplicaciones que la usan para hacer análisis más profundos y potenciar aún más la seguridad de la información que, al día de hoy, es uno de los recursos más vitales dentro de las empresas y de usuarios particulares. Esto nos ayudó bastante para enfocar nuestros esfuerzos en un ámbito menos explorado y poder utilizar herramientas ya existentes en otro tipo de ambientes.

Gracias a esta información, pudimos encontrar el software Weka, que nos ayudó a entender mucho más la información que se podía extraer y cómo se podía aplicar en los diferentes proyectos que se tenían pensados para el futuro del presente proyecto. Bajo la investigación realizada de las diferentes aplicaciones en seguridad del aprendizaje automático, también logramos ver en qué aspectos de la seguridad informática es utilizada en menor medida, para enfocar nuestros esfuerzos e ideas en estas necesidades.

Después de comprender mejor el análisis de los algoritmos de aprendizaje automático, y el entendimiento en el funcionamiento de un sistema de análisis de malwares como lo es Cuckoo Sandbox, pudimos extraer la información adecuada para la aplicación y entrenamiento de un algoritmo como lo es la Red Neuronal Artificial. Las firmas, como comportamientos de un archivo en ejecución dentro del sistema, sirvieron para la identificación y clasificación de los tres tipos de malware propuestos, cumpliendo así con un objetivo y proponiendo diferentes usos para el desarrollo realizado a lo largo del proyecto.

Recomendaciones

Como se ha visto en el desarrollo del proyecto, hay muchas nuevas oportunidades de mejora en todo el proceso. Los ataques no sólo se reducen a las tres familias de malware expuestos, por lo que una buena opción es la implementación de más familias de malware, con más datos para el entrenamiento más robusto de la Red Neuronal Artificial. Además de esto, se recomienda la demostración de los resultados obtenidos de manera gráfica, para mejorar la comprensión de los resultados obtenidos con la herramienta creada.

Como una alternativa a la nueva cantidad de información a recopilar y clasificar, se propone la implementación del algoritmo Random Forest, algoritmo supervisado, para el aumento de acierto de clasificación. Por último, se propone la mejora del algoritmo implementado (Red Neuronal Artificial o Random Forest) para la detección y/o predicción de potenciales malwares nuevos.

Glosario

Machine Learning (aprendizaje automático): Campo de estudio que da a las computadoras la habilidad de aprender sin ser explícitamente programadas.

Red Neuronal Artificial (RNA): Son un paradigma de aprendizaje automático inspirado en las neuronas de los sistemas nerviosos de los animales. Las conexiones tienen pesos numéricos que se adaptan según la experiencia y a un impulso, siendo son capaces de aprender.

Weka Software: Plataforma de software para el aprendizaje automático, escrito en Java y desarrollado en la Universidad de Waikato, Nueva Zelanda.

Cuckoo Sandbox: Es un sistema de análisis de malware, que hace uso de entornos aislados y realiza un registro de acciones ejecutadas por un archivo sospechoso dentro del entorno.

Sandbox: Un entorno de pruebas separado del entorno de producción.

Algoritmo: Es una secuencia de pasos lógicos que permiten solucionar un problema.

Botnet: Es el nombre genérico que denomina a cualquier grupo de PC infectados y controlados por un atacante de forma remota.

Troyano: Es un tipo de virus cuyos efectos pueden ser la eliminación de ficheros o destrucción de la información del disco duro. Además, son capaces de capturar y reenviar datos confidenciales a una dirección externa.

Crypter: Es un software que se utiliza para ocultar los virus, keyloggers o cualquier otra herramienta RAT del antivirus para que no se detectan y se eliminan por los antivirus.

Malware: Malware hace referencia a cualquier tipo de software malicioso que trata de infectar un ordenador o un dispositivo móvil.

Yara: Es una herramienta destinada a ayudar a los investigadores de malware a identificar y clasificar muestras de malware. Se puede crear descripciones de familias de malware basadas en patrones textuales o binarios.

OSSIM: Open Source Security Information Management. Es una colección de herramientas bajo la licencia GPL, diseñadas para ayudar a los administradores de red en la seguridad de las computadoras, detección de intrusos y prevención.

CSV: Son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

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

ARFF: Acrónimo de Attribute-Relation File Format. Es un archivo de texto ASCII que describe una lista de instancias que comparten un conjunto de atributos.

Honeypot: Es una herramienta de la seguridad informática dispuesto en una red o sistema informático para ser el objetivo de un posible ataque informático, y así poder detectarlo y obtener información del mismo y del atacante.

Bibliografía

1. Oktavianto, D. and Muhardianto, I. (2013). Cuckoo malware analysis. Birmingham, UK: Packt Pub.
2. Ali, A. Ransomware: A Research and a Personal Case Study of Dealing with this Nasty Malware.
3. Salvador, R., Radua, J., Canales-Rodríguez, E. J., Solanes, A., Sarró, S., Goikolea, J. M., ... & Moro, N. (2017). Evaluation of machine learning algorithms and structural features for optimal MRI-based diagnostic prediction in psychosis. PloS one, 12(4), e0175683.
4. Kessler, R. C., van Loo, H. M., Wardenaar, K. J., Bossarte, R. M., Brenner, L. A., Cai, T., ... & Nierenberg, A. A. (2016). Testing a machine-learning algorithm to predict the persistence and severity of major depressive disorder from baseline self-reports. Molecular psychiatry, 21(10), 1366.
5. Cimitile, A., Martinelli, F., & Mercaldo, F. (2017). Machine Learning Meets iOS Malware: Identifying Malicious Applications on Apple Environment. In ICISSP (pp. 487-492).
6. Ariu, D., Giacinto, G., & Roli, F. (2011, October). Machine learning in computer forensics (and the lessons learned from machine learning in computer security). In Proceedings of the 4th ACM workshop on Security and artificial intelligence (pp. 99-104). ACM.
7. Munoz, A. (2014). Machine Learning and Optimization. *URL: https://www. cims. nyu. edu/~ munoz/files/ml_optimization. pdf [accessed 2016-03-02][WebCite Cache ID 6fiLfZvnG]*.
8. Miller, S., & Busby-Earle, C. (2016, December). The role of machine learning in botnet detection. In Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for (pp. 359-364). IEEE.
9. Guntuku, S. C., Narang, P., & Hota, C. (2013). Real-time peer-to-peer botnet detection framework based on bayesian regularized neural network. arXiv preprint arXiv:1307.7464.
10. Alejandre, F. V., Cortés, N. C., & Anaya, E. A. (2016). Botnet Detection using Clustering Algorithms. Research in Computing Science, 118, 65-75.
11. Gulmezoglu, B., Eisenbarth, T., & Sunar, B. (2017, April). Cache-Based Application Detection in the Cloud Using Machine Learning. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (pp. 288-300). ACM.
12. Mašetic, Z., Subasi, A., & Azemovic, J. (2016). Malicious Web Sites Detection using C4. 5 Decision Tree. Southeast Europe Journal of Soft Computing, 5(1).
13. Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153-1176.
14. Ashfaq, R. A. R., Wang, X. Z., Huang, J. Z., Abbas, H., & He, Y. L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. Information Sciences, 378, 484-497.
15. Karim, A., Salleh, R., & Khan, M. K. (2016). SMARTbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications. PloS one, 11(3), e0150077.

16. Remagnino, P., Mayo, S., Wilkin, P., Cope, J., & Kirkup, D. (2017). Machine Learning for Plant Leaf Analysis. In Computational Botany (pp. 57-79). Springer Berlin Heidelberg.
17. Braithwaite, S. R., Giraud-Carrier, C., West, J., Barnes, M. D., & Hanson, C. L. (2016). Validating machine learning algorithms for twitter data against established measures of suicidality. JMIR mental health, 3(2).
18. Mousavizadegan, M., & Mohabatkar, H. (2016). An evaluation on different machine learning algorithms for classification and prediction of antifungal peptides. Medicinal Chemistry, 12(8), 795-800.
19. Sreelakshmi K P, Sayana Anil Kumar, Vanitha T, "A review on machine learning in data mining", https://www.ijltet.org/journal_details.php?id=908&j_id=3467, Special Issue - SACAIM - November 2016, 397-400, #ijltetorg
20. Los 10 incidentes de seguridad más importantes del último año. (2017). ESET Latinoamérica. Retrieved 17 July 2017, from http://www.eset-la.com/centro-prensa/articulo/2017/los-10-incidentes-seguridad-mas-importantes-del-ultimo-a%C3%B1o/4439
21. Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. Machine learning, 6(1), 37-66.
22. Milosevic, N., & Kholi, K. (2017). OWASP SeraphimDroid Project - OWASP. Owasp.org. Retrieved 12 June 2017, from https://www.owasp.org/index.php/OWASP_SeraphimDroid_Project
23. Milosevic, N. (2017). Mobile security, OWASP Mobile Top 10, OWASP Seraphimdroid. [online] Es.slideshare.net. Available at: https://es.slideshare.net/nikolamilosevic86/mobile-security-owasp-mobile-top-10-owasp-seraphimdroid [Accessed 4 Aug. 2017].
24. Milosevic, N. (2017). Educational framework added to OWASP Seraphimdroid - Inspiratron.org. Inspiratron.org. Retrieved 18 July 2017, from http://inspiratron.org/blog/2016/08/28/educational-framework-added-to-owasp-seraphimdroid/
25. Malviya, R., & Umrao, B. K. (2014). Machine Learning Security.
26. Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.
27. Zhu, X. (2006). Semi-supervised learning literature survey. Computer Science, University of Wisconsin-Madison, 2(3), 4.
28. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. Journal of artificial intelligence research, 4, 237-285.
29. Argyriou, A., Evgeniou, T., & Pontil, M. (2007). Multi-task feature learning. Advances in neural information processing systems, 19, 41.
30. Damshenas, M., Dehghantanha, A., Choo, K. K. R., & Mahmud, R. (2015). M0droid: An android behavioral-based malware detection model. Journal of Information Privacy and Security, 11(3), 141-157.
31. Xu, Z., Ray, S., Subramanyan, P., & Malik, S. (2017, March). Malware detection using machine learning based analysis of virtual memory access patterns. In 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 169-174). IEEE.
32. Endpoint Security Solutions and Enterprise Security. (2017). Countertack.com. Retrieved 16 July 2017, from http://www.countertack.com/endpoint-threat-platform

33. Technology Overview. Big Data endpoint detection, analysis and response.. (2016). Retrieved from http://cdn2.hubspot.net/hub/150964/file-1356366908-pdf/CounterTack_Tech_Overview.pdf
34. Mousavi, S. A., Rad, B. B., & Wah, T. Y. (2016, September). Systems in Danger: A Short Review on Metamorphic Computer Viruses. In The Third International Conference on Digital Security and Forensics (DigitalSec2016) (p. 20).
35. Endpoint Protection - CylancePROTECT | Cylance. (2017). Cylance.com. Retrieved 19 July 2017, from https://www.cylance.com/en_us/products/our-products/protect.html
36. Šrndić, N., & Laskov, P. (2016). Hidost: a static machine-learning-based detector of malicious files. EURASIP Journal on Information Security, 2016(1), 22.
37. Chen, S., Xue, M., Tang, Z., Xu, L., & Zhu, H. (2016, May). Stormdroid: A streaminglized machine learning-based system for detecting android malware. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (pp. 377-388). ACM.
38. Get real-time protection against advanced threats driven by machine learning and intelligent automation. (2017). [ebook] Available at: https://sentinelone.co.za/wp-content/uploads/2017/04/ds-SentinelOne-EPP-Data-Sheet-FINAL-JUNE-2016.pdf [Accessed 12 Jul. 2017].
39. What is machine learning?. (2017). Sift Science Help Center. Retrieved 18 July 2017, from https://support.siftscience.com/hc/en-us/articles/201786663-What-is-machine-learning-
40. Sadasivam, G. K., Hota, C., & Anand, B. (2017). Detection of Severe SSH Attacks Using Honeypot Servers and Machine Learning Techniques. Software Networking, 2017(1), 79-100.
41. Maiorca, D., Mercaldo, F., Giacinto, G., Visaggio, C. A., & Martinelli, F. (2017, April). R-PackDroid: API package-based characterization and detection of mobile ransomware. In Proceedings of the Symposium on Applied Computing (pp. 1718-1723). ACM.
42. Tsiatsikas, Z., Geneiatakis, D., Kambourakis, G., & Gritzalis, S. (2016, September). Realtime DDoS Detection in SIP Ecosystems: Machine Learning Tools of the Trade. In International Conference on Network and System Security (pp. 126-139). Springer International Publishing.
43. Pope, J. (2016). Ransomware: Minimizing the Risks. Innovations in clinical neuroscience, 13(11-12), 37.
44. Mbah, K. F., Lashkari, A. H., & Ghorbani, A. A. (2017). A PHISHING E-MAIL DETECTION APPROACH USING MACHINE LEARNING TECHNIQUES (Doctoral dissertation, UNIVERSITY OF NEW BRUNSWICK).
45. How It Works. (2017). Cybereason.com. Retrieved 18 July 2017, from https://www.cybereason.com/how-it-works/
46. Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
47. *Malware Data Base | Underc0de*. (2018). *Malwares.underc0de.org*. Retrieved 15 January 2018, from https://malwares.underc0de.org/index.php
48. *ytisf/theZoo*. (2018). *GitHub*. Retrieved 15 January 2018, from https://github.com/ytisf/theZoo