UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDISTRIAL

MAESTRÍA EN TELECOMUNICACIONES

TEMA: SISTEMA DE DETECCIÓN DE INTRUSOS BASADO EN MACHINE LEARNING

Trabajo de Titulación, previo a la obtención del Grado Académico de Magíster en Telecomunicaciones

Modalidad de Titulación: Proyecto de Desarrollo

Autor: Ingeniero Adrián Gabriel Vargas Machuca Del Salto

Director: Ingeniero Víctor Santiago Manzano Villafuerte, Magíster.

Ambato – Ecuador

2021

APROBACIÓN DEL TRABAJO DE TITULACIÓN

A la Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas Electrónica e Industrial.

El tribunal receptor de la Defensa del Trabajo de Titulación, presidido por la Ingeniera Elsa Pilar Urrutia Urrutia Magíster, e integrado por los señores Ingeniero Juan Pablo Pallo Noroña Magíster e Ingeniero Marco Antonio Jurado Lozada Magíster designados por la Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, para receptar el Trabajo de Titulación con el tema: "SISTEMA DE DETECCIÓN DE INTRUSOS BASADO EN MACHINE LEARNING", elaborado y presentado por el señor Ingeniero Adrián Gabriel Vargas Machuca Del Salto, para optar por el Grado Académico de Magíster en Telecomunicaciones; una vez escuchada la defensa oral del Trabajo de Titulación el Tribunal aprueba y remite el trabajo para uso y custodia en las bibliotecas de la Universidad Técnica de Ambato.

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Elsa Pilar Urrutia Urrutia, Mg.

Presidente y Miembro del Tribunal de Defensa

Firmado electrónicamente por:
JUAN PABLO PALLO NORONA

Ing. Juan Pablo Pallo Noroña, Mg.

Miembro del Tribunal de Defensa

Firmado electrónicamente por:
MARCO ANTONIO JURADO LOZADA

Ing. Marco Antonio Jurado Lozada, Mg.

Miembro del Tribunal de Defensa

AUTORÍA DEL TRABAJO DE TITULACIÓN

La responsabilidad de la opiniones, comentarios y críticas emitidas en el Trabajo de Titulación presentado con el tema: “SISTEMA DE DETECCIÓN DE INTRUSOS BASADO EN MACHINE LEARNING”, le corresponde exclusivamente a: Ingeniero Adrián Gabriel Vargas Machuca Del Salto, Autor bajo la Dirección del Ingeniero Víctor Santiago Manzano Villafuerte, Magíster, Director del Trabajo de Titulación; y el patrimonio intelectual a la Universidad Técnica de Ambato.

Firmado electrónicamente por:
ADRIAN GABRIEL VARGAS MACHUCA DEL SALTO .

Ingeniero Adrián Gabriel Vargas Machuca Del Salto

AUTOR

Firmado electrónicamente por:
VICTOR SANTIAGO MANZANO VILLAFUERTE

Ingeniero Víctor Santiago Manzano Villafuerte, Magister.

DIRECTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que el Trabajo de Titulación, sirva como un documento disponible para su lectura, consulta y procesos de investigación, según las normas de la Institución.

Cedo los Derechos de mi Trabajo de Titulación, con fines de difusión pública, además apruebo la reproducción de este, dentro de las regulaciones de la Universidad Técnica de Ambato.

Firmado electrónicamente por:
ADRIAN GABRIEL VARGAS MACHUCA DEL SALTO .

Ingeniero Adrián Gabriel Vargas Machuca Del Salto

C.C. 1803131190

ÍNDICE GENERAL

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

AGRADECIMIENTO

A Dios todo poderoso que me guía en cada camino de mi vida

Al mi tutor, Ingeniero Santiago Manzano, que me ha dado su gran apoyo y guía para hacer posible este proyecto

Al Ingeniero Patricio Córdoba que ha sido un gran amigo y apoyo incondicional

A Cisne Almeida, gracias por tu apoyo sincero y constante paciencia en mis altos y bajos

A mi madre que ha sido una pieza fundamental en mi vida y proyectos

A mi abuelito por ser mi inspiración y modelo a superar
A mi abuelita que me sigue dando sus consejos desde el cielo

A mis amigos David G., Freddy N., Homero V., Fabian R., que me impulsan con sus triunfos y derrotas
A mis docentes de la Universidad Técnica de Ambato, por su valiosa guía en mi formación

A todos mis amigos, clientes y conocidos, gracias infinitas.
Mis logros solo son la suma de todos sus esfuerzos en conjunto.

Adrián Vargas

DEDICATORIA

A mi padre celestial que me impulsa a ser un hijo fuerte y que se esfuerza

A mi madre que es mi ejemplo a seguir y motivo de superación

A Cisne que es mi inspiración para seguir adelante

A Cisne Almeida, gracias por tu apoyo sincero y constante paciencia en mis altos y bajos

A mi abuelita que me guía orgullosa desde lo más alto

Adrián Vargas

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN TELECOMUNICACIONES

TEMA:

SISTEMA DE DETECCIÓN DE INTRUSOS BASADO EN MACHINE LEARNING

AUTOR: Ingeniero Adrián Gabriel Vargas Machuca Del Salto

DIRECTOR: Ingeniero Víctor Santiago Manzano Villafuerte, Magíster

LÍNEA DE INVESTIGACIÓN:

- TECNOLOGÍA, SEGURIDAD Y GESTIÓN DE REDES DE COMUNICACIONES

FECHA: 19 de octubre de 2021

RESUMEN EJECUTIVO

En base a la masificación de los servicios en la nube y las redes de datos, actualmente toda la información valiosa de una empresa está interconectada a la red para agilitar el trabajo y gestionar los diferentes procesos, sin embargo esto la pone en riesgo ante un ataque a nivel de red, dejando no solo expuesto sus datos confidenciales, sino también parando el ritmo de trabajo dependiendo de sus servicios en la nube, como pueden ser servicio de correo electrónico, página web, base de datos, entre otros. Estos servicios en red son susceptibles a ataques como pueden ser denegación de servicio (DoS) y todas sus variantes, envío masivo de correo spam, escaneo de puertos o ataques de fuerza bruta. Todos estos ataques se pueden detectar a nivel de red usando sistemas de detección de intrusos (IDS), el problema es la necesidad de actualizar constantemente su base de datos que detecta ataques en base a una lista negra, de manera similar a como trabaja un antivirus convencional. Con machine learning se propone levantar un sistema de detección de intrusos basado en patrones de comportamiento, para detectar ataques de fuerza bruta y reportarlo en una página web.

Anteriores investigaciones ya han sentado las bases para aplicar Machine Learning en este campo, usando algoritmos como arboles de decisión, el cual es un algoritmo supervisado muy

efectivo para la clasificación booleana. La investigación planteó de manera similar la aplicación de bosques aleatorios que es la combinación iterativa de árboles de decisión, lo que mejora en la mayoría de los casos el error en clasificación.

El sistema propuesto cursa por dos fases principales, la primera es la fase de entrenamiento donde se captura todo el tráfico malicioso usando el honeypot Cowrie para generar un modelo entrenado de clasificación, lo que se realiza una única vez. Luego en la fase de prueba, el algoritmo detecta en tiempo real los ataques recibidos a la IP pública de la empresa Icono Sistemas y los clasifica como malicioso o no.

Al final y de manera experimental se identificó mediante la matriz de confusión generada por el algoritmo en WEKA que el sistema basado en bosques aleatorios es capaz de detectar satisfactoriamente un ataque de fuerza bruta, sin importar si la amenaza apunta a un puerto o ip específicos.

Este Sistema de bajo coste será capaz de adaptarse a los ataques básicos y sus variaciones, para disparar una alerta en caso de detectarlos y facilitar una posterior toma de medidas por parte del administrador, como bloqueo de puertos de entrada o salida específicos, limitar el tráfico de una interfaz, etc. en caso de tráfico sospechoso.

Descriptores: Ataques de red, Machine Learning, Bosques Aleatorios, Ataque de fuerza bruta, Weka, Cowrie, IDS.

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN TELECOMUNICACIONES

THEME:

INTRUDER DETECTION SYSTEM BASED ON MACHINE LEARNING

AUTHOR: Ingeniero Adrián Gabriel Vargas Machuca Del Salto

DIRECTED BY: Ingeniero Víctor Santiago Manzano Villafuerte, Magíster

LINE OF RESERCH:

- TECHNOLOGY, SECURITY AND MANAGEMENT OF COMMUNICATIONS NETWORKS

DATE: October 19, 2021

EXECUTIVE SUMMARY

Based on the massification of cloud services and data networks, currently all the valuable information of a company is interconnected to the network to speed up the work and manage the different processes, however this puts it at risk in the face of an attack at the network level, leaving not only your confidential data exposed, but also stopping the pace of work depending on your cloud services, such as email service, website, database, among others. These network services are susceptible to attacks such as denial of service (DoS) and all its variants, mass spamming, port scanning or brute force attacks. All these attacks can be detected at the network level using intrusion detection systems (IDS), the problem is the need to constantly update its database that detects attacks based on a black list, in a similar way to how an antivirus works. conventional. With machine learning, it is proposed to build an intrusion detection system based on behavior patterns, to detect brute force attacks and report it on a web page.

Previous research has already laid the foundations to apply Machine Learning in this field, using algorithms such as decision trees, which is a very effective supervised algorithm for Boolean classification. The research similarly raised the application of random forests, which is the iterative combination of decision trees, which improves classification error in most cases.

The proposed system goes through two main phases, the first is the training phase where all malicious traffic is captured using the Cowrie honeypot to generate a trained classification model, which is done only once. Then in the testing phase, the algorithm detects in real time the attacks received on the public IP of the company Icono Sistemas and classifies them as malicious or not.

In the end and experimentally, it was identified by the confusion matrix generated by the WEKA algorithm that the system based on random forests is capable of successfully detecting a brute force attack, regardless of whether the threat is targeting a specific port or IP.

This low-cost system will be able to adapt to basic attacks and their variations, to trigger an alert in case of detection and facilitate subsequent action by the administrator, such as blocking specific input or output ports, limiting traffic of an interface, etc. in case of suspicious traffic.

Descriptors: Network Attacks, Machine Learning, Random Forests, Brute Force Attack, Weka, Cowrie, IDS

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. Introducción

La falta de seguridad en internet y las constantes amenazas que circulan con el fin de tener acceso no autorizado hacen que la implementación de sistemas de seguridad y detección de amenazas, sea una prioridad a la hora de levantar servicios en la nube disponibles para los usuarios de la red.

Ecuador ocupó en 2017 según Kaspersky el primer lugar en América Latina con un 2.8% de los ataques recibidos por fuerza bruta, y el quinto lugar a nivel mundial en lo que respecta a amenazas de red. Los principales métodos de detección de intrusos IDS utilizan un algoritmo de clasificación de amenazas que se basa en una lista negra de comando e IP's, que requieren de constantes actualizaciones para mantenerse eficaces y efectivos. [1]

Se propone la implementación del aprendizaje de máquina para detectar intrusos a nivel de red, lo que le aporta flexibilidad al algoritmo basado en actualizaciones, permitiéndole adaptarse a las nuevas amenazas para detectar y aprender patrones de intrusión recientes. El paquete de software Weka desarrollado por la Universidad de Waikato contiene los algoritmos más conocidos de Machine Learning que se aplicará al estudio mediante su código fuente en Java. [2]

El algoritmo de Árboles de decisión es una técnica de aprendizaje supervisado de máquina para clasificar datos, con eficacia comprobada al trabajar con data sets de ataques en red. Por su parte los bosques aleatorios son la mejora del método anterior, ya que hacen uso de iteraciones consecutivas de árboles de decisión con lo que se mejora la eficacia del método y se reduce el error, se busca aplicarlo de manera similar a la detección de intrusos y evidenciar su mejora en detección de ataques de fuerza bruta.

Finalmente, el sistema en Ubuntu Server con el algoritmo de Machine Learning precargado con el modelo de clasificación, es capaz de identificar amenazas en un entorno real y notificar la incidencia por medio de una página web que se actualiza automáticamente de manera periódica cada 30 segundos.

1.2. Justificación

Los sistemas de detección de intrusos sirven para clasificación y detección de amenazas en una red, basado en el comportamiento o huellas que dejan los intrusos como IP's de origen, tipo de peticiones o escaneo de puertos, lo que condiciona al algoritmo de detección a conocer previamente el comportamiento de los atacantes, por lo que no es capaz de detectar amenazas más recientes. [6]

Comúnmente los honeypots se encargan de recolectar información del comportamiento de un atacante, haciéndose pasar por un servidor real, y registrando todos los métodos y comandos utilizados por los bots o scripts de ataques de fuerza bruta que se ejecutan en el internet, para luego mediante intervención de un programador, extraer la lista de comportamientos sospechosos que servirá de actualización a la base de datos de un IDS, de manera muy similar a como trabaja un antivirus convencional de computadora. [7]

Los IDS basados en comportamientos tienen el principal inconveniente que requieren constantemente estar actualizando para mantenerse medianamente eficaces ante los ataques de red codificados por hackers más recientes. [8]

La presente investigación plantea el uso de técnicas de Machine Learning para entrenar a un modelo de detección de amenazas, lo que le permitiría prescindir de constantes actualizaciones y aprender de manera automática los patrones de comportamiento de las nuevas amenazas de red que surgen a cada segundo.

1.3.Objetivos

1.3.1. Objetivo General

Implementar un sistema de detección de Intrusos mediante Machine Learning.

1.3.2. Objetivos Específicos

- Analizar los principales ataques a nivel de Red.
- Implementar un modelo de aprendizaje de ataques a nivel de red con Machine Learning.
- Desarrollar un Sistema de detección de intrusos en un ambiente real de trabajo mediante Machine Learning.

CAPÍTULO II

ANTECEDENTES INVESTIGATIVOS

2.1. Estado del Arte

Naveen Bindra y Manu Sood en el año 2018, en la Universidad de Shimla de India, desarrollaron el trabajo: "Detectando ataques DDoS usando técnicas de Machine Learning y data sets contemporáneos de detección de intrusos", donde se ha evidenciado a lo largo del tiempo que los ataques que se presentan en mayor medida son los de denegación de servicio distribuido a la red, donde identificar y eliminar este tipo de tráfico constituye un verdadero reto de desarrollo e implementación dada la variedad de sistemas, protocolos y tecnologías en los Routers presentes en el mercado, por lo que se centraron en el uso de machine learning, específicamente usando el algoritmo de aprendizaje Random Forest, con una precisión del 96% en detección de amenazas de red. [9]

En el instituto indú de tecnología, Mayank Agarwal desarrolló su investigación en Machine Learning en el año 2016 para detectar ataques de flooding en redes 802.11 y localizar al atacante, centrado en el campo de los ataques de Denegación de servicio de inundación (Flooding) dirigidos a Routers Wifi del estándar 802.11, por su baja seguridad en la capa MAC de control de acceso al medio, debido a la poca exploración e investigación desarrollada en esta área de estudio. Agrawal y compañía proponen un método de bajo coste y fácil implementación usando Machine Learning para detectar el tráfico de inundación que sobrecarga los equipos AP y bloquearlo, y posteriormente identificar la posición del atacante usando el algoritmo de localización según el ángulo de llegada de la onda para ignorar toda señal que proviene de esa dirección, logrando una tasa de detección y precisión superior al 95% usando Adaboost. [10]

En el año 2018 Raouf Boutaba desarrolla una encuesta exhaustiva sobre Machine Learning para redes, donde ha revisado exhaustivamente un compendio bibliográfico importante para dar una visión global del estado del arte, desde el punto de vista de calidad de servicio, optimización de tráfico de red y seguridad de la red, en lo que se refiere a networking. Así mismo se hace énfasis en la necesidad de avanzar en el estudio de Machine Learning para determinar su factibilidad de crear redes autónomas en el contexto actual de alta demanda de

datos, creciente número de usuarios y aplicaciones. Además, el estudio encontró que la mayoría de investigaciones previas de seguridad de redes se enfocan en una sola capa y un solo usuario, por lo que la proyección debe ser a multitud de capas y redes. [11]

Wang en colaboración con Xu, Wu, Zheng, Xu y Qiao en el año 2018, hicieron su estudio en el departamento de educación y ciencia en China mediante Machine Learning para determinar la robustez de la red de sistemas de múltiples agentes bajo ataque, donde se enfocaron en una red MAS con grandes nodos para determinar su solidez, usando Redes Neuronales NN que es básicamente la percepción de múltiples capas MLP para entender la estructura de redes de múltiples agentes MAS, con softmax como clasificador. Los datos se contrastaron con dos métodos de red neuronal convolucional CNN que se usan comúnmente en datos estructurados con datos artificiales generados por un modelo gráfico aleatorio de Erdos-Renyi, con lo que lograron una mejora ligeramente superior con respecto a los métodos basado en redes neuronales convolucionales CNN [12]

Las investigaciones aportan al trabajo con el estudio de métodos de machine learning como bosques aleatorios y árboles de decisión, captura de datos a nivel de red y análisis comparativo de eficacia entre machine learning y redes neuronales, para aplicarlo en detección de ataques de fuerza bruta en la red.

2.2. Marco Teórico

Aplicaciones Web

Las aplicaciones Web son programas realizados en un lenguaje específico para ejecutarse en un navegador web a través de internet o intranet y que se alojan en un servidor web para acceder en cualquier momento.

Seguridad Informática

Seguridad informática es el proceso de evitar el acceso no autorizado a un sistema de información privada con fines maliciosos, para obtener provecho de la información o simplemente por error, mediante la prevención y detección por software como antivirus, firewalls, IDS, etc.

2.2.1. Riesgos de seguridad en aplicaciones web

Una empresa que oferta sus servicios informáticos a redes de acceso tendrá la necesidad de que su información y recursos estén protegidos, ya que internet es un evidente riesgo del mal uso de los servicios e información disponibles por lo que todas las aplicaciones web deben estar protegidas ante posibles ataques.

En la figura 2.1 se tiene una comparación de las tres metodologías principales para análisis de ciberseguridad con sus pros y contras. [13]

Metodología	Pros	Contras
ISO 27001	Ofrece una buena guía para la gestión de toda la documentación referente a la seguridad.	La complejidad de su lectura se asemeja a la de textos legales. No cubre la seguridad de un proyecto, solo a nivel de organización.
OSSTMM	Presenta diferentes módulos que se encargan de cubrir diferentes áreas de seguridad en una organización. Ofrece una métrica sobre el nivel de seguridad de la organización, así como la forma de utilizarla.	Algunos de los controles de seguridad propuestos pueden resultar excesivos para la mayoría de organizaciones. No se cubre la seguridad de un proyecto, solo a nivel de organización.
OWASP Testing Guide	Cubre la seguridad de un proyecto. Es la metodología que más se acerca a nuestras necesidades.	Algunas fases del ciclo de vida no se profundizan.

Figura 2.1:Comparación de metodologías de Ciberseguridad

Fuente [14]

Inyección

Una inyección consiste en insertar un código malicioso alterando así el funcionamiento normal de una base de datos. Los ataques de inyección más comunes son SQL, LDAP O CRFL.

Riesgos de inyección SQL. Es un ataque que puede ocurrir cuando una página utiliza los datos dados por un usuario como parte de una consulta. La inyección SQL permite al atacante eliminar o modificar datos que han sido almacenados en una base de datos. [15]

Ataques DoS y DDoS

Los ataques de denegación de servicio DoS y denegación de servicio distribuido DDoS buscan privar a los usuarios de un recurso o servicio en red. No suele darse un robo de información,

pero puede resultar muy costoso, representando una importante pérdida de dinero y tiempo para el usuario u organización afectada.

Los ataques de negación de servicio son una de las amenazas más potentes a las que tienen que hacer frente las empresas ya que los hackers pueden utilizar este colapso de redes como arma para pedir un rescate de tipo económico. En la figura 2.2 se visualiza el esquema de ataque DoS a un servidor Web. [16]

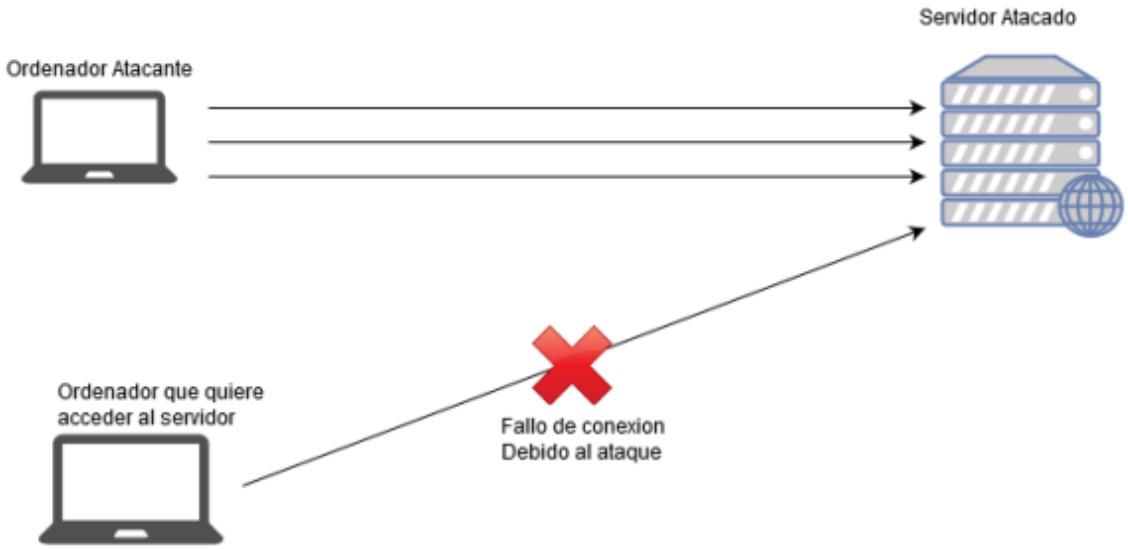

Figura 2.2:Ataque DoS a un servidor web

Fuente [17]

1. Ping de la muerte: El ping de la muerte es una variante de ataque DDoS se basa en una alteración de un protocolo IP, esto sucede cuando se manda un ping (conocido como herramienta de medición de latencia) a un sistema, colapsando así al destinatario.

2. Slowloris: Slowloris es una amenaza que ocurre cuando colapsa un servidor mediante un envío masivo de conexiones HTTP reduciendo al mínimo sus recursos, siendo así uno de los tipos DDoS más difíciles de superar.

3. Inundación Syn: Inundación Syn es un tipo de ataque DDoS que consiste en suplantar la identidad de un usuario para así inundar servidores con paquetes Syn provocando el colapso total del servicio en red.

4. Inundación de puertos sin servicio: La inundación de puertos sin servicio ocurre cuando los puertos son inundados por un envío masivo de paquetes TCP/UDP dejando sin acceso a los servidores.

5. Inundación por fragmentos: La inundación por fragmentos es un ataque que ocurre cuando innumerables paquetes diseñados mediante fragmentación ingresan a los servidores evadiendo así cualquier tipo de firewall.

6. Inundación mediante paquetes anómalos: La inundación mediante paquetes anómalos ocurre con la presencia de paquetes con anomalías provocando una sobrecarga y haciendo que los servidores queden inutilizados. Este ataque puede evitarse con una configuración correcta del firewall.

7. Inundación por mezcla: Inundación por mezcla se da por una combinación de varios tipos de ataques lanzados al mismo tiempo generando el colapso y caída de los servidores o a su vez una desconfiguración total del sistema.

8. Inundación de puertos de servicio: Inundación de puertos de servicio ataca los puertos colapsándolos por completo. La inversión para recuperarse de este ataque puede suponer un gasto enorme para recuperar todos los servicios.

9. Inundación ICMP: La inundación ICMP genera una ralentización de los servidores por el envío de grandes volúmenes de paquetes ICMP dañando severamente el sistema y ocupando el ancho de banda, extorsionando así a sus víctimas.

10. Inundación zombi: La inundación zombi utiliza conexiones auténticas, dificultando así detección. Como resultado la red queda inutilizada y el ancho de banda copado. [18]

Ataque de fuerza bruta

El ataque de fuerza bruta es un tipo de amenaza de red que escanea servicios como SSH y Telnet para probar distintos usuarios y contraseñas mediante un diccionario de credenciales de seguridad más comunes e ingresar sin autorización a un servidor o servicio. [19]

En la mayoría de los casos los ataques de fuerza bruta se ayudan de software a manera de bots automáticos para ir probando las diferentes combinaciones hasta dar con las credenciales correctas para vulnerar un sistema. [20]

2.2.2. Machine Learning

Machine Learning es una rama de la inteligencia artificial basada en la idea en que los sistemas pueden aprender de datos y tomar decisiones con mínima de intervención humana, automatizando así la construcción de modelos analíticos.

Existen algunas técnicas que pueden utilizarse para la transformación de información no estructurada en datos que puedan ser analizados y procesados por un ordenador. Por otro lado, las técnicas de modelización de información estructurada pueden clasificarse en supervisada y no supervisada. [22]

Aprendizaje supervisado: El aprendizaje supervisado es una técnica de machine learning que requiere un conjunto de datos previamente clasificados correctamente que sirve como conjunto de entrenamiento para la máquina. Es un problema de regresión cuando la variable a predecir es continua, mientras que es nominal cuando existe un problema de clasificación.

Aprendizaje no supervisado: El aprendizaje no supervisado es una técnica de machine learning que no requiere un conjunto de datos previamente clasificados para predecir una salida ya sea de tipo discreta o continua, únicamente se basa en las características del conjunto de entrada. [23]

En la figura 2.3 se puede ver la clasificación de algoritmos de machine learning, donde se sub divide en supervisados y no supervisados.

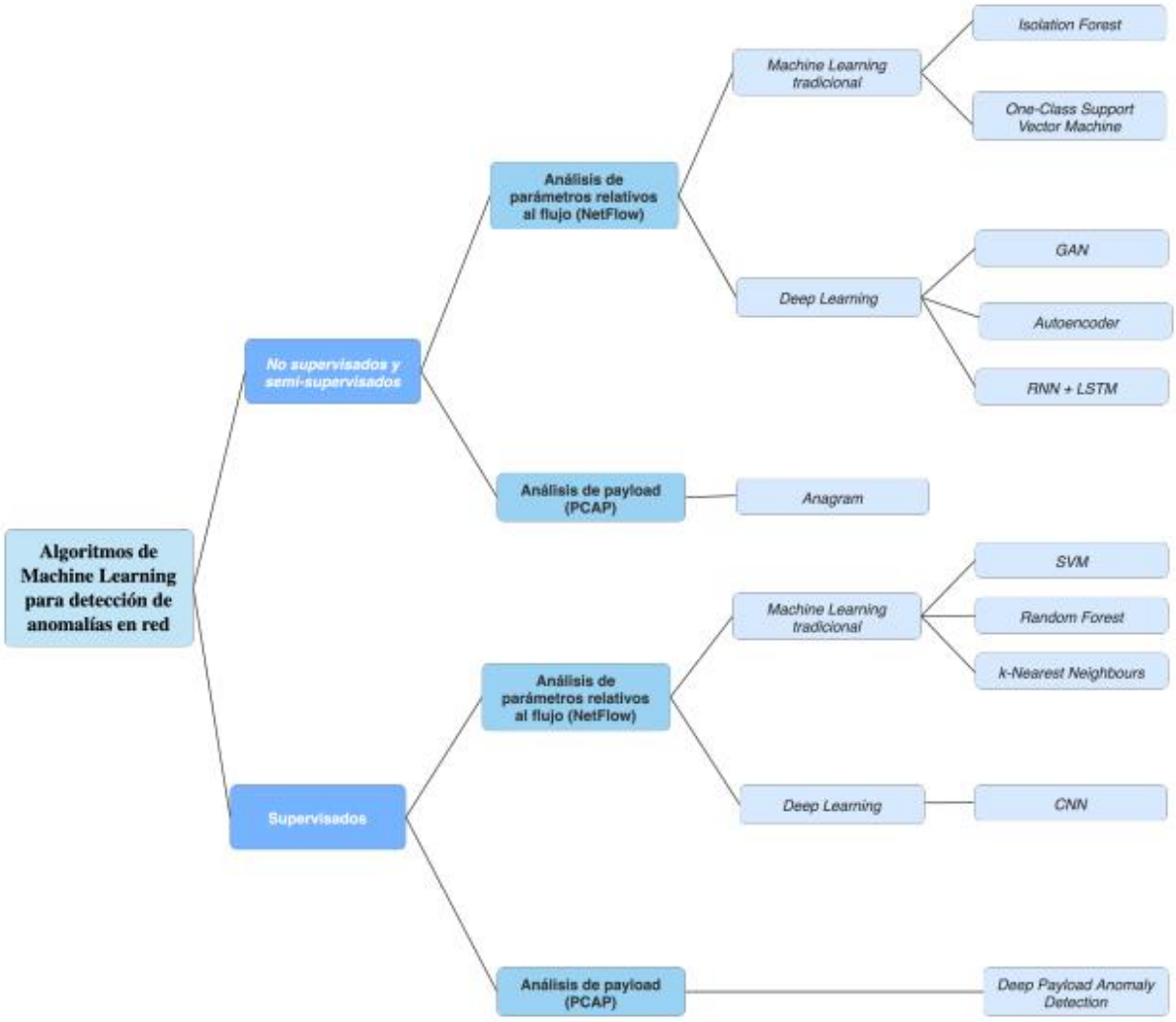

Figura 2.3: Taxonomía de los algoritmos de ML para detección de anomalías en red.
Fuente [24]

Fases de Machine Learning:

Las fases de machine learning son entendimiento de los datos, preparación de datos, selección de la técnica apropiada, entrenamiento y clasificación.

Entendimiento de los datos. - El entendimiento de los datos es el análisis de la calidad de la información.

Preparación de los datos. - La preparación de los datos es la limpieza de los mismos; es decir, reducción o eliminación de las variables redundantes.

Selección de la técnica apropiada. – La selección de la técnica apropiada consiste en la transformación de datos para su modelización y se subdivide en dos partes.

1. Homogenización del rango de las variables

2. Identificación de variables relevantes, detectando las variables que no aportan información. [25]

Entrenamiento de los datos. - El entrenamiento de los datos consiste en cargar un dataset con datos previamente clasificados al algoritmo, para que aprenda de los mismos y cree un modelo predictivo en baso al algoritmo de machine learning que se elija.

Clasificación de los datos. – Con el modelo predictivo el algoritmo es capaz de clasificar nuevos datos entrantes.

Redes neuronales

Las redes neurales son modelos matemáticos multivariantes no lineales su objetivo es minimizar una determinada función de error, clasificando las observaciones. Estas redes neuronales están conectadas y dichas conexiones emulan las dendritas y los axones en los sistemas nerviosos biológicos por lo que pasa la información. Se emplean en problemas supervisados como no supervisados. [26]

En la figura 2.4 se puede observar los componentes principales de una neurona que es la estructura que utiliza el algoritmo supervisado de redes neuronales.

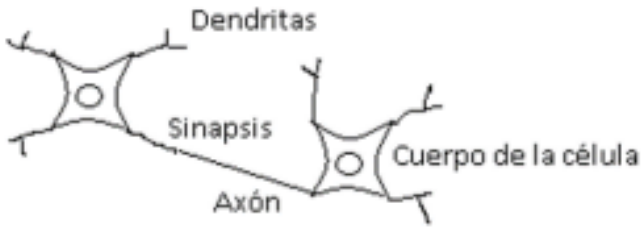

Figura 2.4: Componentes principales de una neurona

Fuente [27]

Como una extensión se tienen las redes profundas que se trata del uso de redes neuronales con múltiples capas y esto se conoce como Deep Learning, pueden tener millones de parámetros en función de la complejidad del problema en abordar. No obstante, con su dificultad de estimación caben múltiples aproximaciones en el uso de este tipo de métodos. [28]

Las máquinas de vector soporte, pretenden clasificar a las observaciones en varios grupos, pero estas no son separables vía un hiperplano en el espacio dimensional definido por los datos, para esto el conjunto de datos se embebe en un espacio de dimensión superior a través de una función que permita poder separar los datos en el nuevo espacio a partir de un hiperplano en dicho espacio de forma simultánea. [29]

Los clasificadores bayesianos son modelos que asumen que la presencia o ausencia de ciertas características permita asignar cierta probabilidad a la ausencia o presencia otra característica. Esta técnica clasifica observaciones en un conjunto de clases tomando formas específicas según la distribución. [30]

Los árboles de clasificación y de regresión son técnicas de análisis que permite predecir la asignación de muestras a grupos predefinidos en función de una serie de variables. Son modelos sencillos y fáciles de interpretar, pero su poder predictivo puede ser más limitado que el de otros modelos porque se realiza una partición octogonal del espacio lo que limita su capacidad predictiva. [31]

El clustering se utiliza para identificar grupos de observaciones similares en un conjunto de datos. El método más utilizado es el k means que consiste en definir un punto central de referencia de cada clúster y asignar a cada individuo al clúster del centroide más próximo. El algoritmo parte de la fijación de k centroides aleatoriamente y se asigna cada punto al clúster con el centroide más próximo procediendo a actualizar el valor de los centroides, este proceso se termina cuando se alcanza un determinado criterio de convergencia. [32]

En la figura 2.5 se puede ver la clasificación de un clúster, que puede ser jerárquico o particional.

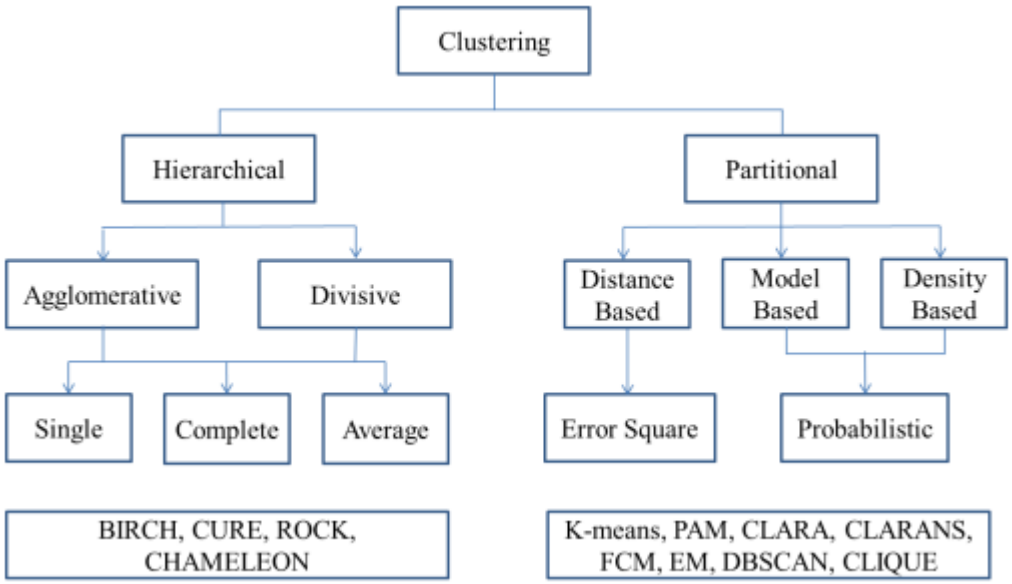

Figura 2.5: Taxonomía del clustering

Fuente [33]

2.2.3. Marco para el desarrollo de software prototipo

Python:

Python es un lenguaje de programación utilizado para desarrollar aplicaciones de todo tipo, de código abierto por lo tanto gratuito lo que permite el desarrollo de software sin límites. Facilita trabajar con inteligencia artificial, big data, machine learning y data science. [34]

En la figura 2.6 se observa las etapas a seguir en machine learning usando el estándar de Python para la preparación, modelado y virtualización de los datos a clasificar.

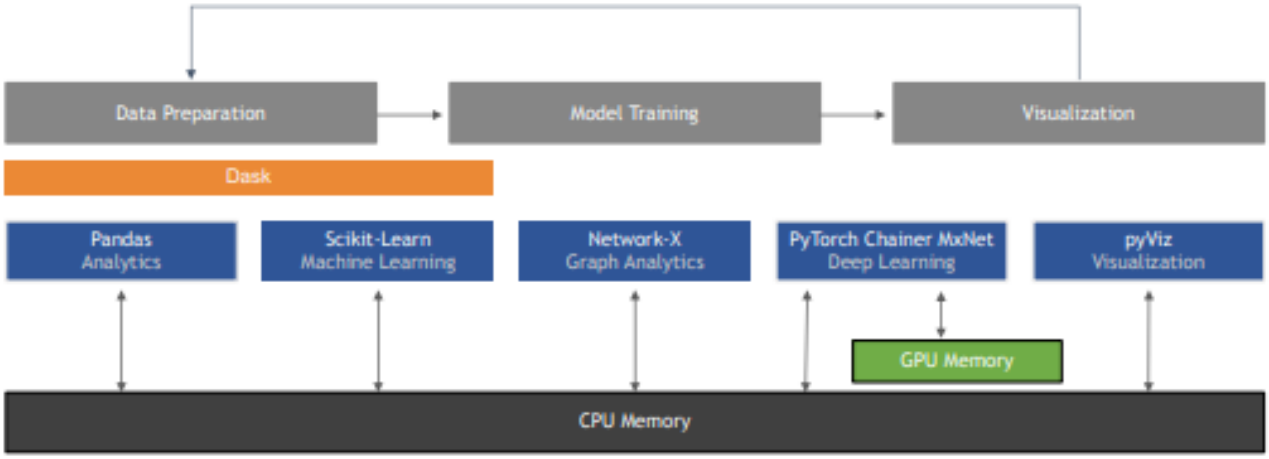

Figura 2.6: El estándar de Python para Machine Learning

Fuente [35]

Sistema de detección de intrusiones (IDS)

El sistema de detección de intrusos por sus siglas en inglés IDS es un programa que detecta accesos no autorizados a una red u ordenador, generando algún tipo de alerta o log para que así pueda ser resuelto por el administrador de sistemas correspondiente. IDS no actúa frente a un ataque, simplemente alerta del mismo. [36]

Los tipos de IDS que se tienen son HIDS y NIDS

HIDS (HostIDS).- Monitorea el tráfico entrante y saliente de un host específico.

NIDS (NetworkIDS).- Captura todo el tráfico de la red y detecta tráfico inusual.

Snort

Snort es un sistema de detección de intrusos basado en red open source que cuenta con un lenguaje de creación en donde se puede definir los patrones que se utilizará a la hora de monitorizar la red. Puede funcionar como sniffer (se puede ver en consola y en tiempo real qué ocurre en la red, y todo el tráfico), registro de paquetes o como un IDS normal. Cuando un paquete coincide con algún patrón establecido en las reglas de configuración se loguea y así sabe cuándo y dónde se produjo el ataque. [37]

El diagrama de flujo en la figura 2.7 indica el proceso que sigue el algoritmo de Snort, donde se captura el flujo de red, detecta el tipo de amenaza, descarta el paquete, se genera un log del evento y posteriormente se da la alarma en caso de detectar coincidencias con su base de datos.

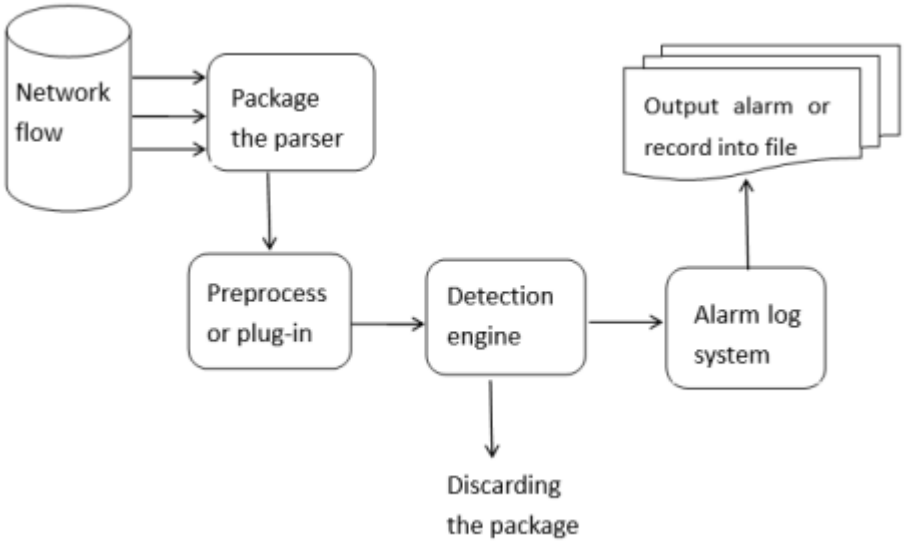

Figura 2.7: El flujo de procesamiento del paquete Snort

Fuente [38]

CAPÍTULO III

METODÓLOGÍA

3.1. Ubicación

Se realizó en la ciudad de Ambato, en la empresa Ícono Sistemas, que se encuentra en la parroquia la Matriz, debido a que esta empresa se dedica a la implementación de sistemas de red, por lo que es un punto de convergencia donde se presentan constantes ataques a nivel de red.

3.2. Equipos y materiales

A continuación, se detallan los equipos y materiales utilizados en el desarrollo de la investigación, según la tabla 3.1.

Tabla 3.1: Equipos y Materiales

PRESUPUESTO					
Ítem	**Descripción**	**Unidad**	**Cantidad**	**Precio Unitario**	**Precio Total**
1	Servidor	1	c/u	$550.00	$550
2	Switch	2	c/u	$28.00	$56
3	Transporte urbano	100	c/u	$0.30	$30
4	Imprevistos				$100
	TOTAL				$966

Fuente: Elaborado por el Investigador

3.3. Tipo de investigación

3.3.1. Investigación Bibliográfica

Esta investigación es de carácter bibliográfico ya que la información para su realización se obtuvo de fuentes de información presente en artículos científicos y revistas de investigación, así como tesis relacionadas y libros.

3.3.2. Investigación Experimental

Se tuvo que experimentar para llegar a datos correctos.

3.4. Prueba de Hipótesis

El trabajo de investigación al ser de carácter aplicado, busca dar solución a un problema en específico, lo que sirve de prueba a la hipótesis.

Hipótesis

El sistema de detección de intrusos detecta en tiempo real ataques de fuerza bruta a una IP pública de la empresa Ícono Sistemas.

Variables

La variable independiente serán las amenazas de red, mientras que los algoritmos de Machine Learning serán la variable dependiente, sujeto a variar dependiendo de la cantidad de amenazas entrantes y sus características.

3.5. Población y Muestra

Dado que se realiza una investigación con característica experimental, no se hace especialmente necesario seleccionar una población y muestra.

CAPÍTULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

4.1. Diseño del prototipo

4.1.1. Esquema del Diseño

El sistema cuenta con un servidor Ubuntu versión 20.04 alojado virtualmente en un equipo físico de la empresa Ícono Sistemas, conectado directamente a internet con IP pública para recibir ataques en la red para entrenar al algoritmo de Machine Learning.

Dentro del servidor Ubuntu se ejecutan 8 scripts para adquisición, filtrado y procesamiento datos de manera autónoma y preprogramando su ejecución cada cierto lapso de tiempo.

El Honeypot Cowrie instalado dentro del servidor Ubuntu será el encargado principal de capturar el tráfico malicioso de manera segura en un archivo de logs, que servirá para el entrenamiento del algoritmo y posterior análisis de datos recibidos.

La presentación corre a cargo de una página web alojada en un hosting externo que recibe los datos clasificados como “amenaza detectada” o “sin amenazas”, que los presenta de manera clara y precisa recibiendo las actualizaciones del servidor Ubuntu en tiempo real. En la figura 4.1 se especifica el diagrama de red del sistema de detección de ataques de fuerza bruta y el hosting Byethost para presentación de datos.

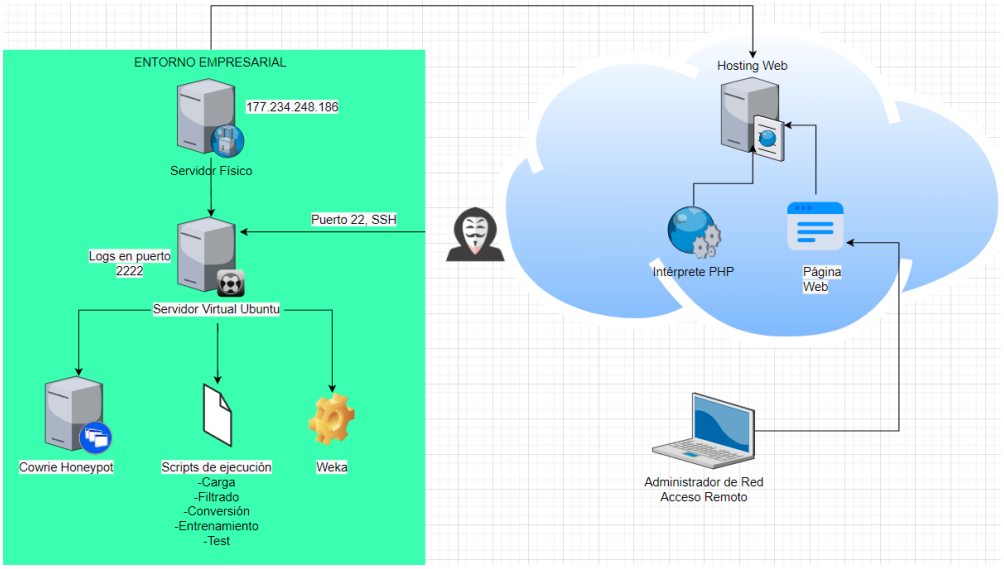

Figura 4.1: Diagrama del Sistema de Detección de Intrusos

Fuente: Desarrollado por el investigador en base a [38]

Los ataques que ingresen por SSH serán redirigidos al puerto 2222, directamente al servidor Ubuntu donde el Honeypot hace la captura de inicios de sesión y lo guarda en un archivo de logs. Mediante los scripts de ejecución el archivo de logs es copiado al usuario root para filtrar las palabras clave “New connection” y “login attempt” que se contabilizan según la ip y puerto de origen desde donde proviene el posible ataque. El archivo resultante se convierte a un formato legible por el software Weka para entrenar los datos y luego en segunda instancia clasificarlos de manera automática usando el algoritmo bosques aleatorios. Al final se envía al Hosting Byethost el archivo resultante de la clasificación por Weka para prestar los datos en una página web programada para indicar si ha habido ataques en los últimos instantes al servidor Ubuntu.

El primer paso es instalar el servidor virtual Ubuntu en el servidor físico de la empresa Icono Sistemas, donde se levantan los servicios de OpenSSH y el honeypot Cowrie para captura de datos de intento de inicio de sesión.

En segunda instancia se diseñan y codifican los scripts de ejecución para capturar datos y realizar el entrenamiento de machine learning con el algoritmo de bosques aleatorios, lo que genera un archivo modelo para la etapa de clasificación.

Como tercera parte se levanta los servicios de hosting gratuito byethost y carga la página web con el archivo php que recibe los datos del servidor Ubuntu. Finalmente se automatiza el proceso activando el servicio de ejecución programada crontab en el servidor Ubuntu, lo que crea un bucle repetido infinito del proceso de detección de amenazas.

4.1.2. Activación del servicio OpenSSH y apertura de puertos

El servidor OPENSSH es parte esencial del sistema de detección de intrusos para que se reciban los ataque por fuerza bruta en el servidor Ubuntu, que está configurado por defecto para escuchar por el puerto 22, redireccionado por motivos de seguridad al puerto 2222, donde entra en funcionamiento el Hoyneypot Cowrie. En la figura 4.2 se observa que el demonio sshd está activo y escuchando.

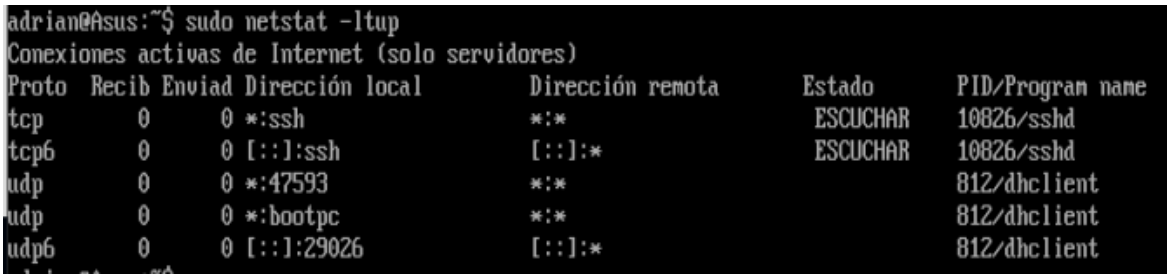

Figura 4.2: Figura: Comprobación de puertos abiertos para SSH en el Servidor Virtual

Fuente: Desarrollado por el investigador

La opción -l es para para visualizar los puertos a la escucha, -t para adjuntar todos los puertos que son de TCP, -u adjunta los puertos y servicios en UDP y finalmente la opción -p los procesos activos de cada puerto.

4.1.3. Activación del Honeypot

Los Honeypot como T-Pot traen un conjunto de aplicaciones previamente instaladas sobre Ubuntu o compilaciones similares, para ejecutarse de manera conjunta y determinar diferentes tipos de ataques en red, para este caso en particular se centra en detectar ataques de fuerza bruta, por lo que Cowrie fue el paquete a elegir dentro de las librerías de T-Pot.

Es preciso que corra en un entorno virtualizado para luego proceder a la escucha de puertos y registrar el tráfico que ingresa en un archivo .log, que se llenará con los intentos fallidos y exitosos de inicio de sesión en el servidor. Se puede ver en la figura 4.3 a continuación que cowrie-env es el entorno virtualizado que se ejecuta para habilitar el inicio de Cowrie en el servidor.

Figura 4.3: Inicialización del entorno virtual para Cowrie

Fuente: Desarrollado por el investigador

Con todas las librerías instaladas y el entorno virtual activado se da inicio al Honeypot mediante start, con lo que se pone a la escucha al puerto 22 y 23. Importante aclarar que cowrie no se debe iniciar de ninguna manera en usuario root/, ya que se pondría en entre dicho la seguridad de servidor al estar recibiendo constantemente tráfico malicioso en la red de internet pública, completamente abierto, la que la función principal del honeypot es la de trabajar como señuelo para que entren todos los ataques que circulan por la red, y registrar en archivo el comportamiento del mismo y sus respectivos comandos, para luego diseñar en este caso un método de detección de dichos ataques, mediante Machine Learning.

Para que cowrie se ejecute correctamente, se crea un usuario llamado cowrie, sin contraseña para permitir los ataques de inicio de sesión de manera relativamente sencilla y comenzar a registrar el comportamiento de algoritmos maliciosos en su interior. Por tal motivo el usuario cowrie tiene permisos de ejecución limitados. En la figura 4.4 se visualiza el honeypot cowrie iniciando su ejecución de manera satisfactoria.

Figura 4.4: Inicio de Cowrie en el entorno virtual de Python

Fuente: Desarrollado por el investigador

Se verifica que cowrie esté escuchando, con el comando netstat, donde aparecen listados los servicios de SSH en TCP para el puerto 2222. En la figura 4.5 se puede ver que el puerto 2222 está redireccionado y a la escucha para tráfico ssh entrante.

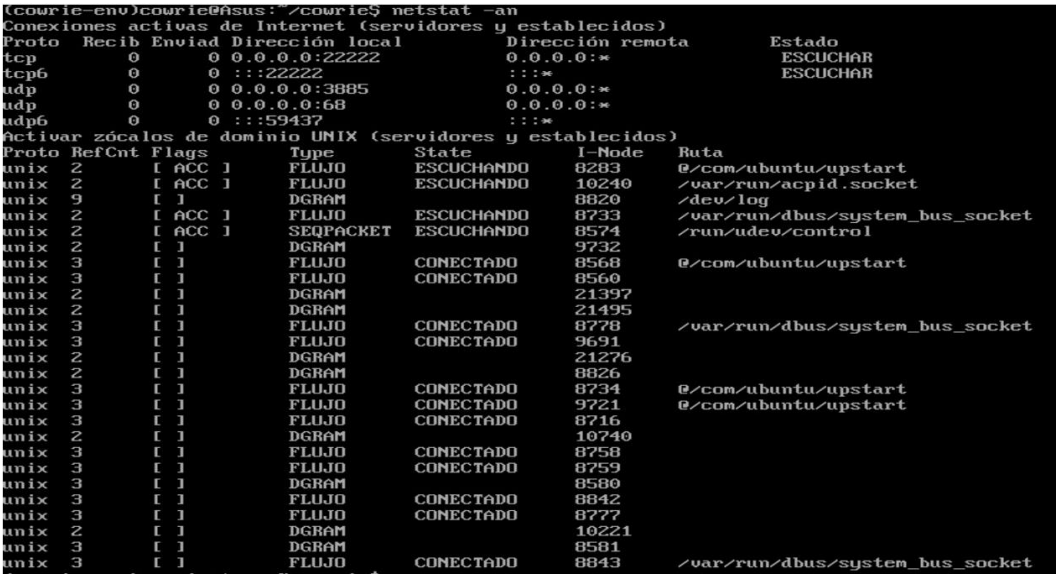

Figura 4.5: Comprobación de estado activo de Cowrie

Fuente: Desarrollado por el investigador

Se puede verificar en cualquier momento que el honeypot cowrie está ejecutándose mediante el comando bin/cowrie status con permisos de administrador, como se muestra en la figura 4.6. Esta comprobación es especialmente importante ya que, si el honeypot cowrie no se encuentra activo, el sistema de detección de intrusos no presentará ninguna amenaza.

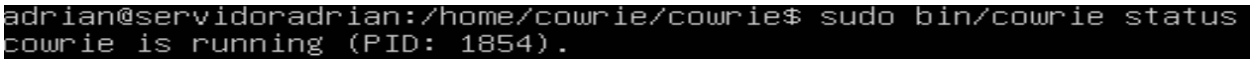

Figura 4.6: Verificación de estado de Cowrie

Fuente: Desarrollado por el investigador

Un vistazo en tiempo real con el comando tail -f, permite evidenciar que el honeypot se encuentra a la espera y listo para registrar ataques de fuerza bruta que intenten ingresar por el puerto 2222 para SSH. Se puede ver en la figura 4.7 que el sistema está listo para captar conexiones entrantes por SSH y Telnet y las registra en el archivo cowrie.log.

Figura 4.7: Comprobación de estado activo de Cowrie

Fuente: Desarrollado por el investigador

4.1.4. Recolección de datos SSH con Cowrie

El servidor Ubuntu cuenta con la IP 192.168.1.101 configurada de manera estática, mientras que la máquina que ejecuta la petición SSH cliente, mediante PuttyGen en Windows 10, tiene la IP 192.168.1.11. En la figura 4.8 se verifica que se registran los usuarios con los que se hacen las pruebas como “root”, así mismo la IP de origen, conexiones exitosas, conexiones rechazadas, e intentos de conexión.

Se utiliza PuttyGen para generar tráfico positivo en el honeypot, ya que el tráfico negativo se va a generar de manera automática en la red pública de internet, por los bots que hacen escaneos de IPs y puertos de manera constante, en busca de potenciales dispositivos carentes de contraseñas robustas y la seguridad necesaria como IDS (Sistema de detección de intrusos), Firewalls de red, entre otros. Como se puede ver en la figura 4.8 hay varios intentos de inicio de sesión en el servidor Ubuntu con el usuario root y contraseña 1234.

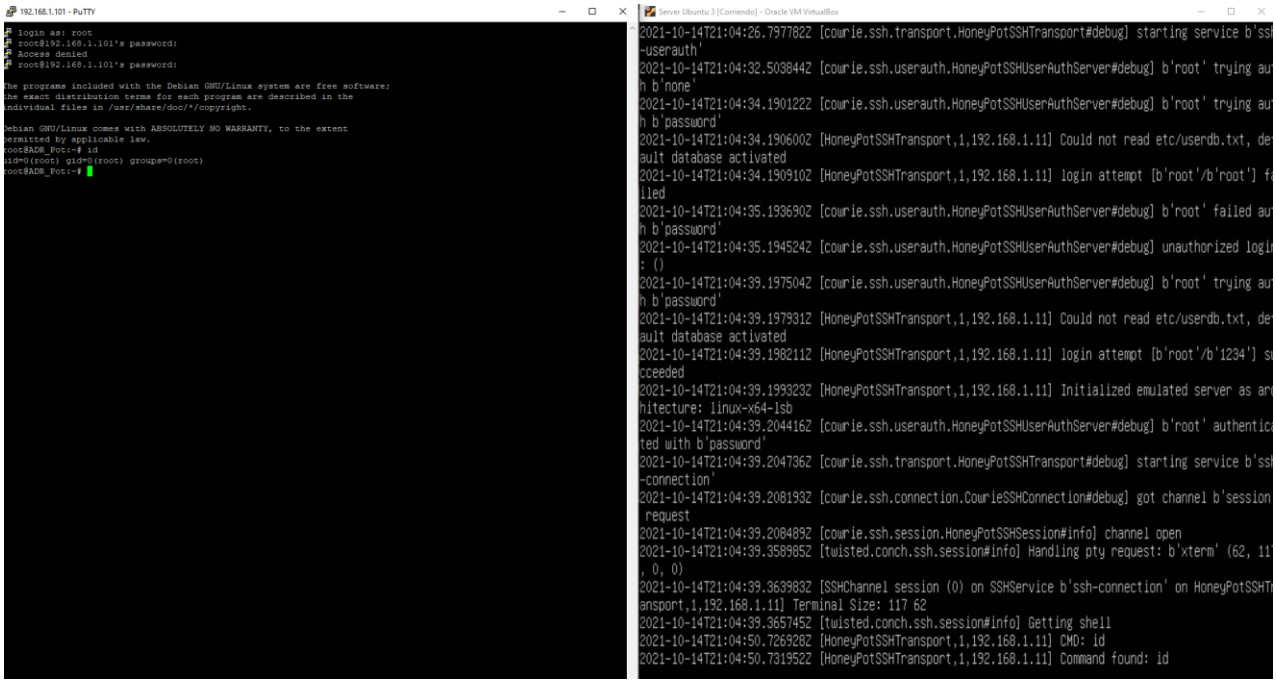

Figura 4.8: Inicio de sesión de manera externa al servidor y registro autónomo en destino

Fuente: Desarrollado por el investigador

4.1.5. Diseño y funcionamiento de Scripts de ejecución

Adquisición de datos.

Se genera en el usuario cowrie un archivo de logs que queda alojado en la dirección /home/cowrie/cowrie/var/log/cowrie.log que se migra a la carpeta de usuario con privilegios con la siguiente instrucción sudo cp /home/cowrie/cowrie/var/log/cowrie.log /home/adrian/alllogs.log, donde cp sirve para copiar archivos de un directorio de origen a otro de destino. Luego se imprime con el comando echo “Recuperación exitosa” y se guarda en un archivo de registro de eventos denominado registro.log y la fecha del evento mediante la siguiente línea de comando:

echo "Recuperación exitosa: $(date)" >>/home/adrian/prototype/debuglogs/registro.log

Selección de características.

En este script se migra nuevamente el archivo de logs a la ubicación /home/adrian/prototype/logs para mantener una buena organización durante el tratamiento de los datos.

El comando grep permite extraer coincidencias de un archivo, para este caso buscamos las coincidencias exactas con la instrucción uniq -u, y solo se extraen de las últimas 100 instrucciones para no sobrecargar memoria. Se buscan las líneas de registro que contengan las palabras “New conection\” y “login attempt” separadas por una pleca “|”. El resultado de selección se guarda en /home/adrian/prototype/logs/extractedlogs con el siguiente comando:

sudo grep 'New connection\|login attempt' /home/adrian/prototype/logs/alllogs.log | uniq -u | tail -n 100 > /home/adrian/prototype/logs/extractedlogs/cowrieresult.log

Este filtrado exitoso también se guarda con fecha en el registro de eventos.

Conteo de incidencias en formato CSV

Este script se codifica en PERL donde los datos son extraídos del log en el script anterior formando las siguientes cuatro columnas: Puerto de origen, Estado de inicio de sesión, Intentos de inicio según el puerto, e intentos de inicio según la IP de origen.

El número de puerto se coloca directamente, en el archivo, al igual que el estado de conexión. Para el caso de los Intentos_al_Puerto e Intentos_a_IP se utiliza un contador que va incrementando con los ingresos desde una misma IP e ingresos desde un mismo puerto, atributos que servirán para determinar cuáles son ataques de fuerza bruta y cuáles no. Todos los campos son separados por comas y guardados en un archivo CSV en la ruta /home/adrian/prototype/honeycsv/. El código completo se puede encontrar en la parte de anexos.

Conversión a ARFF y entrenamiento

Se ejecuta la librería converters.CSVLoader de Weka para importar el archivo CSV y pasarlo a formato comprensible por Weka que es un archivo en formato de relación de atributos (ARFF) muy similar al archivo CSV, con la diferencia que tiene una cabecera que le indican al algoritmo de Weka el tipo de información que se va a manejar:

Puerto - numérico

Estado - {exitoso, fallido}

Intento al puerto - numérico

Intentos a la IP – numérico

Malicioso {0, 1}

El comando utilizado es:

sudo java -cp /usr/share/java/weka.jar weka.core.converters.CSVLoader
home/adrian/prototype/honeycsv/cowrie.csv
> home/adrian /prototype/honeycsv/trainfiles/cowrie.arff

Donde -cp sirve para indicar la ubicación de Weka, seguido de la librería que importa el archivo a ARFF, seguido de la ubicación del archivo CSV y la ubicación destino del archivo importado.

Seguidamente con el comando sudo java -cp /usr/share/java/weka.jar weka.classifiers.trees.Randomforest -t /home/adrian/prototype/honeycsv/trainfiles/cowrie.arff -d $path/prototype/honeycsv/models/cowrie.model se genera el bosque aleatorio que entrena en base al archivo ARFF, obteniendo a la salida la matriz de confusión y el modelo de predicción que se utiliza en lo posterior. La opción -t indica la ruta del archivo de entrenamiento y -d indica la dirección destino donde se guardará el modelo creado con bosques aleatorios.

Clasificación de datos entrantes

Se repiten los scripts de adquisición, selección y conteo de datos, cambiando únicamente el número de datos adquiridos de 100 en la etapa de entrenamiento a 60 datos para mejorar la velocidad de predicción del modelo.

Para utilizar el modelo en clasificación la opción -l indica la ruta del modelo a utilizar, -T la ruta de los datos a clasificar y “>” especifica la ruta donde guardar los datos clasificados por el algoritmo. El archivo resultante es cowrie.txt.

java -cp /usr/share/java/weka.jar weka.classifiers.trees.Randomforest
-l /home/adrian/prototype/honeycsv/models/cowrie.model
-T /home/adrian/prototype/honeycsv/testfiles/cowrietesttest.arff
> /home/adrian/prototype/honeycsv/results/cowrie.txt

Envío de resultados a la aplicación Web

Especificando las variables host, user y password, se genera el envío de los resultados de clasificación al servidor mediante el comando send, a través del protocolo ftp.

La instrucción cd /htdocs indica la ruta de destino en el hosting Web.

4.2. Algoritmo de Machine Learning

El algoritmo de machine learning de bosques aleatorios se genera en dos etapas, la primera es la fase es de entrenamiento donde se recopila la mayor cantidad de información, creando un archivo modelo con la matriz de confusión, que servirá para realizar la futura clasificación de nuevos datos entrantes, que pueden categorizarse en dos tipos: como tráfico amenaza y tráfico auténtico.

La segunda fase es la fase de Clasificación, donde se aplica el algoritmo de machine Learning ya entrenado para identificar el tráfico entrante, si es malicioso o no, los resultados de esta ejecución se guardan en un archivo.txt, que posteriormente se subirá a un hosting externo para mostrar si hubo un ataque mediante una página web básica.

Extracción y Preparación de Datos

La fase de extracción y preparación de datos se realiza mediante scripts, donde se copian los registros generados por el Honeypot a una carpeta del usuario principal, se extraen los atributos relevantes de la totalidad de los datos para luego ponerlo en formato .CVS (Valores separados por coma)

Con el archivo en formato .CVS, se hace uso de un tercer script codificado en PERL, para darle el formato que acepta el algoritmo de Machine Learning, generando al final un archivo.arff (Formato de archivo de relación de atributos).

En este punto se debe editar manualmente el archivo.arff, verificando los intentos de conexión que se consideran como ataques de fuerza bruta, y los intentos de conexión que pueden ser un intento de conexión normal, colocando una columna adicional de ceros y unos, que serán la base del entrenamiento.

4.2.1. Fase de Entrenamiento

En la fase de entrenamiento de Machine Learning el software de código abierto Weka se utiliza con su librería bosques aleatorios para generar un modelo de clasificación.

Se envía el archivo. ARFF a Weka para entrenar usando el algoritmo de bosques aleatorios, ya que es una versión superior de árboles de decisión donde se genera un número elevado de folds o iteraciones para menorar así el error en clasificación.

El algoritmo de árboles aleatorios genera la matriz de confusión que es una herramienta para calcular la eficiencia en base al conjunto de entrenamiento y al conjunto de test que se divide

en relación 70/30, donde el 70% de los datos son de entrenamiento y el 30% de test para evidenciar verdaderos positivos VP, verdaderos negativos VN, falsos positivos FP y falsos negativos FN, utilizando la fórmula

$$\boldsymbol{Tasa\ de\ Error} = \frac{\boldsymbol{FP + FN}}{\boldsymbol{Total}}$$

A continuación, en la tabla 4.1 se indica la matriz de confusión, en base a ataques de fuerza bruta que se puede tener en el servidor Ubuntu.

Tabla 4.1: Matriz de confusión

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos	Falsos Negativos
	Negativos	Falsos Negativos	Verdaderos Negativos

Fuente: Desarrollado por el investigador en base a [35]

4.2.2. Fase de Clasificación

En la fase se utilizan de manera similar los scripts generados anteriormente, para recolectar nuevos datos del archivo de registros del honeypot Cowrie, extraer únicamente las columnas de “Intentos de Sesión por IP” por su gran cantidad de información que aportan a la toma de decisión, y la columna de “estado” que indica si se logró el inicio de sesión.

Con los últimos 100 datos extraídos en un archivo separado por comas CVS, se convierte al formato que acepta Weka, para luego ejecutar el algoritmo de árboles aleatorios, usando el modelo ya entrenado y generar un archivo .TXT (de texto) con la clasificación de los últimos 100 datos como 0’s o 1’s, donde 0 significa que no existe amenaza, y 1 significa que se trata de un ataque de fuerza bruta.

4.2.3. Monitoreo de Ataques

El monitorio de ataques se logra por medio de una página web sencilla que está ubicada por motivos de seguridad y fácil acceso en un servidor web externo BYET Host que permite crear cuentas gratuitas para el efecto con un máximo de almacenamiento de 1000 mega bytes.

La interfaz de la página web recibe de manera periódica cada 30 segundos una actualización por medio de un script alojado en el servidor Ubuntu, para presentar los siguientes datos:

-Estado de subida de datos desde el Servidor Ubuntu (Carga Exitosa, Carga Fallida)

-Estado de detección de ataques de fuerza bruta (Sin Amenazas, Posible amenaza detectada)

-Estado del Servidor Ubuntu (Online-Offline)

Se realiza el registro en el hosting web Byet para tener acceso a los servicios de alojamiento web y las credenciales de subida de datos mediante FTP. En la figura 4.9 se puede ver los datos de conexión al panel de control, acceso FTP y la URL de la página web.

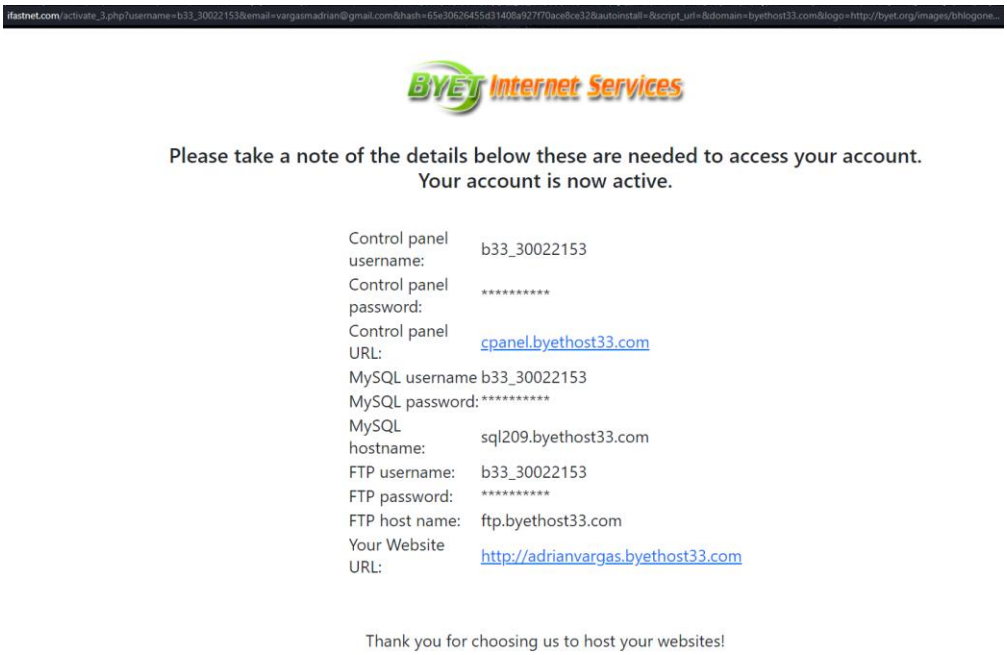

Figura 4.9: Datos del Hosting para la Página Web

Fuente: Desarrollado por el investigador

Desde el panel de control vistapanel se puede visualizar el buen funcionamiento del hosting, así como el espacio utilizado y el ancho de banda que es ilimitado como se muestra en la figura 4.10.

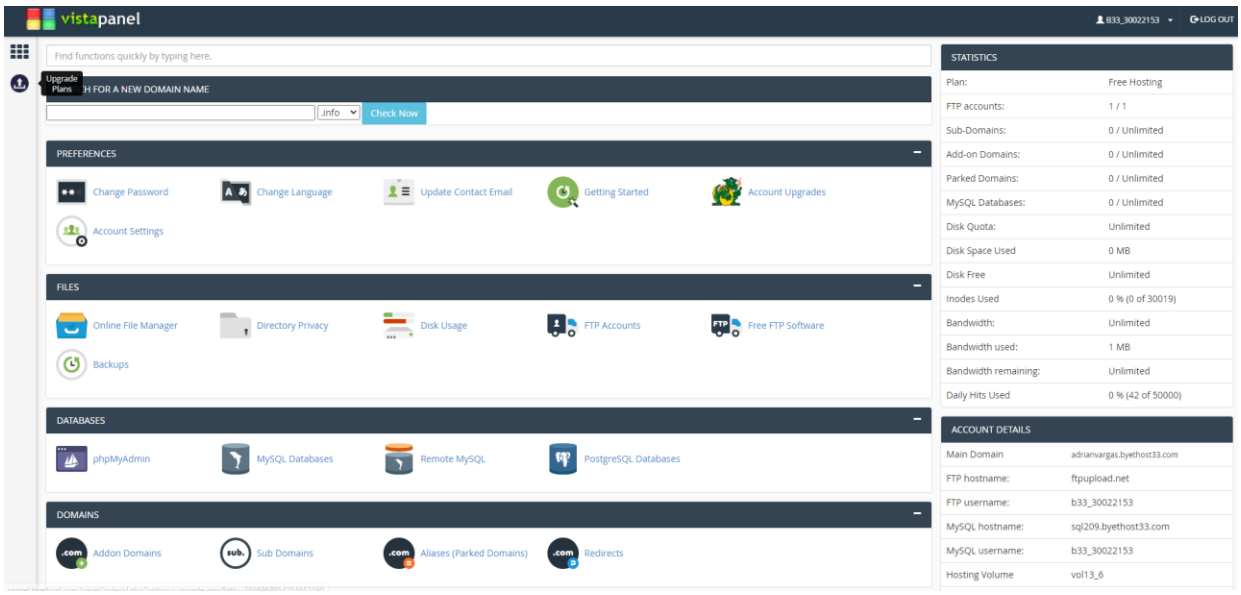

Figura 4.10: Panel de Control del Hosting

Fuente: Desarrollado por el investigador

Por medio de un script adicional en el servidor se hace el envío del archivo cowrie.txt, que tiene los datos de reporte de clasificación generado en la fase anterior por el algoritmo de bosques aleatorios, para que el encargado de la red verifique si ha habido nuevos ataques de fuerza bruta en los últimos 30 segundos a la empresa.

4.2.4. Automatización del proceso

El sistema de detección de amenazas o ataques de fuerza bruta se estará ejecutando de manera casi en tiempo real; es decir, cada 30 segundos, por medio del comando crontab que ejecutará de manera automática los scripts y en orden secuencial.

Es importante aclarar que la fase de entrenamiento solo se la realiza una vez, para generar el modelo de Machine Learning entrenado con los datos capturados durante una semana de

ejecución del honeypot. Luego de este paso inicial se va a estar ejecutando de manera repetida la toma de los últimos 100 registros para la fase de Clasificación y Monitoreo, por lo que solo se consideran estos scripts en el archivo de configuración de crontab.

4.3. Pruebas del sistema en Entorno Empresarial

4.3.1. Prueba del Sistema en Entrenamiento del Algoritmo de Machine Learning

En la empresa Ícono Sistemas se realizó las pruebas tanto para la fase de aprendizaje como para la fase de test, debido a que teniendo salida a una IP pública se va a recibir la mayoría de los ataques que rondan internet, mejorando el registro de datos y aumentando la eficiencia del algoritmo de Machine Learning en detección de nuevas amenazas.

La IP pública que posee el servidor con honeypot es la 177.234.248.186, dejando a la escucha el honeypot durando un lapso de 2 semanas, para no saturar la memoria RAM de la máquina Host del servidor virtual desde el 1 de octubre al 15 de octubre del 2021, capturando un total de 18385 datos.

Tras dos semanas de mantener en línea el honeypot, se extrae el archivo de logs del sistema para comenzar con la ejecución de los scripts antes mencionados en el levantamiento del prototipo.

En la figura 4.11 se puede observar los últimos logs de inicio de sesión del 15 de octubre, donde los campos de “login attempt” y “New connection” se van a extraer para entrenar y testear al modelo de predicción de machine learning.

```
2021-10-15T15:33:36.750955Z [HoneyPotSSHTransport,1108,141.98.10.60] login attempt [b'user'/b'1234'] failed
2021-10-15T15:33:37.753984Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'user' failed auth b'password'
2021-10-15T15:33:37.755132Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2021-10-15T15:33:37.966045Z [HoneyPotSSHTransport,1108,141.98.10.60] Got remote error, code 11 reason: b'Normal Shutdown, Thank you for playing'
2021-10-15T15:33:37.966583Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2021-10-15T15:33:37.966698Z [HoneyPotSSHTransport,1108,141.98.10.60] Connection lost after 2 seconds
2021-10-15T15:33:44.915112Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 141.98.10.60:52348 (177.234.248.186:2222) [session: 458626eea082]
2021-10-15T15:33:44.916459Z [HoneyPotSSHTransport,1109,141.98.10.60] Remote SSH version: SSH-2.0-libssh2_1.4.3
2021-10-15T15:33:45.128048Z [HoneyPotSSHTransport,1109,141.98.10.60] SSH client hassh fingerprint: 92674389fa1e47a27ddd8d9b63ecd42b
2021-10-15T15:33:45.130119Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'diffie-hellman-group14-sha1' key alg=b'ssh-rsa'
2021-10-15T15:33:45.130243Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha1' b'none'
2021-10-15T15:33:45.130361Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha1' b'none'
2021-10-15T15:33:45.562953Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2021-10-15T15:33:45.773831Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2021-10-15T15:33:45.985526Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'user' trying auth b'password'
2021-10-15T15:33:45.985879Z [HoneyPotSSHTransport,1109,141.98.10.60] Could not read etc/userdb.txt, default database activated
2021-10-15T15:33:45.986078Z [HoneyPotSSHTransport,1109,141.98.10.60] login attempt [b'user'/b'123456'] failed
2021-10-15T15:33:46.989369Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'user' failed auth b'password'
2021-10-15T15:33:46.989837Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2021-10-15T15:33:47.201254Z [HoneyPotSSHTransport,1109,141.98.10.60] Got remote error, code 11 reason: b'Normal Shutdown, Thank you for playing'
2021-10-15T15:33:47.201766Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2021-10-15T15:33:47.201882Z [HoneyPotSSHTransport,1109,141.98.10.60] Connection lost after 2 seconds
2021-10-15T15:48:18.468763Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 103.161.16.161:44762 (177.234.248.186:2222) [session: 62dc3f61589e]
2021-10-15T15:48:18.768858Z [HoneyPotSSHTransport,1110,103.161.16.161] Remote SSH version: SSH-2.0-libssh-0.1
2021-10-15T15:48:19.066960Z [HoneyPotSSHTransport,1110,103.161.16.161] Got remote error, code 11 reason: b'Bye Bye'
2021-10-15T15:48:19.067469Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2021-10-15T15:48:19.067585Z [HoneyPotSSHTransport,1110,103.161.16.161] Connection lost after 0 seconds
2021-10-15T15:52:43.743515Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 205.185.118.155:42380 (177.234.248.186:2222) [session: b6c244a79d7d]
2021-10-15T15:52:43.881209Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2021-10-15T15:52:43.881436Z [HoneyPotSSHTransport,1111,205.185.118.155] Connection lost after 0 seconds
2021-10-15T15:55:43.057832Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 221.181.185.151:13685 (177.234.248.186:2222) [session: a5847252b53d]
2021-10-15T15:55:43.064579Z [HoneyPotSSHTransport,1112,221.181.185.151] Remote SSH version: SSH-2.0-PUTTY
2021-10-15T15:55:43.378821Z [HoneyPotSSHTransport,1112,221.181.185.151] SSH client hassh fingerprint: 1616c6d18e845e7a01168a44591f7a35
2021-10-15T15:55:43.388467Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'ecdh-sha2-nistp256' key alg=b'ecdsa-sha2-nistp256'
2021-10-15T15:55:43.388819Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-512' b'none'
2021-10-15T15:55:43.389116Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-512' b'none'
2021-10-15T15:55:44.006731Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2021-10-15T15:55:44.318483Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2021-10-15T15:55:44.631055Z [HoneyPotSSHTransport,1112,221.181.185.151] Got remote error, code 11 reason: b''
2021-10-15T15:55:44.631688Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2021-10-15T15:55:44.631843Z [HoneyPotSSHTransport,1112,221.181.185.151] Connection lost after 1 seconds
2021-10-15T16:01:28.269950Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 222.186.42.13:63508 (177.234.248.186:2222) [session: e7959a06a5bc]
2021-10-15T16:01:28.277707Z [HoneyPotSSHTransport,1113,222.186.42.13] Remote SSH version: SSH-2.0-PUTTY
2021-10-15T16:01:28.559990Z [HoneyPotSSHTransport,1113,222.186.42.13] SSH client hassh fingerprint: 1616c6d18e845e7a01168a44591f7a35
2021-10-15T16:01:28.572069Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'ecdh-sha2-nistp256' key alg=b'ecdsa-sha2-nistp256'
2021-10-15T16:01:28.572373Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-512' b'none'
2021-10-15T16:01:28.572664Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-512' b'none'
2021-10-15T16:01:29.122308Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2021-10-15T16:01:29.401890Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2021-10-15T16:01:30.445914Z [HoneyPotSSHTransport,1113,222.186.42.13] Got remote error, code 11 reason: b''
2021-10-15T16:01:30.446937Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2021-10-15T16:01:30.447213Z [HoneyPotSSHTransport,1113,222.186.42.13] Connection lost after 2 seconds
```

Figura 4.11: Logs de intentos de inicio de sesión al servidor Ubuntu durante 2 semanas de escucha

Fuente: Desarrollado por el investigador

En la figura 4.11 consta como han existido diferentes intentos de inicio de sesión y sesiones exitosas al honeypot desde IP’s públicas que atacan por defecto el puerto 22 de SSH, el mismo que está redireccionado al puerto 2222 donde está a la escucha el sistema.

El primer script logretreiver_1.sh se encarga de copiar el archivo de logs ubicado en el usuario cowrie, y lo guarda en la carpeta de privilegios restringidos del usuario root “adrian”. En la

figura 4.12 se puede ver al script 1 que ha recibido y cargado exitosamente los logs en el usuario root.

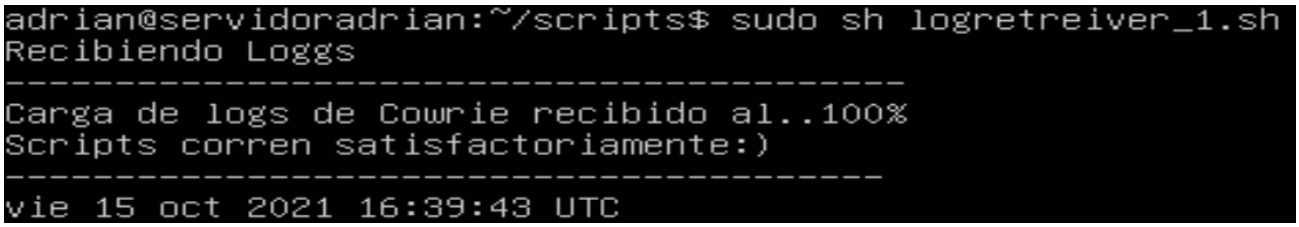

Figura 4.12: Ejecución exitosa del scrip logretreiver_1.sh

Fuente: Desarrollado por el investigador

Con el archivo guardado exitosamente en la carpeta de logs, lo siguiente es extraer la información relevante como “New connection” y “login attempt” que representan los intentos de sesión al servidor, para el efecto el comando grep permite extraer las coincidencias. Como se puede visualizar en la figura 4.13 el filtrado resulta exitoso al ejecutar el script 2.

Figura 4.13: Ejecución exitosa del script extractor_2.sh

Fuente: Desarrollado por el investigador

Una vez ejecutado correctamente el filtrado de datos, se quedan solamente los datos de las conexiones nuevas y los intentos de inicio de sesión en el archivo de logs como se muestra en la figura 4.14.

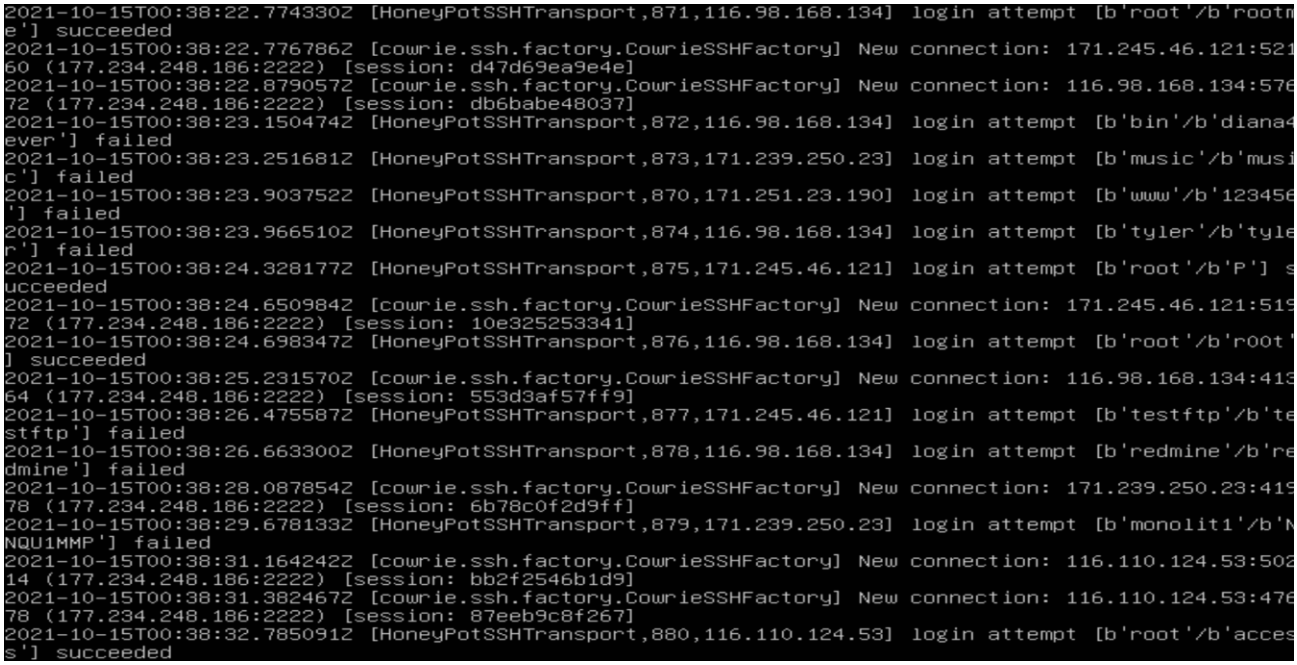

Figura 4.14: Archivo de log filtrado por New connection y login attempt

Fuente: Desarrollado por el investigador

A continuación, se ejecuta el script codificado en Perl, que realiza el conteo y conversión de los datos filtrados para presentarlos por columnas, indicando el puerto de origen, estado de la conexión, intentos de login por ese puerto e intento de login por la IP. En la figura 4.15 se evidencia la ejecución completa del script de conteo de incidencias.

```
adrian@servidoradrian:~/scripts$ sudo perl prototype_3.prg
Feature Selection and formatting complete!!
```

Figura 4.15: Ejecución exitosa del script prototype_3.prg

Fuente: Desarrollado por el investigador

El archivo resultante de conteo de datos se crea en formato CSV, separado por comas como se puede ver en la figura 4.16, es importante identificar que la última columna que indica si el dato es malicioso o no se debe ingresar manualmente.

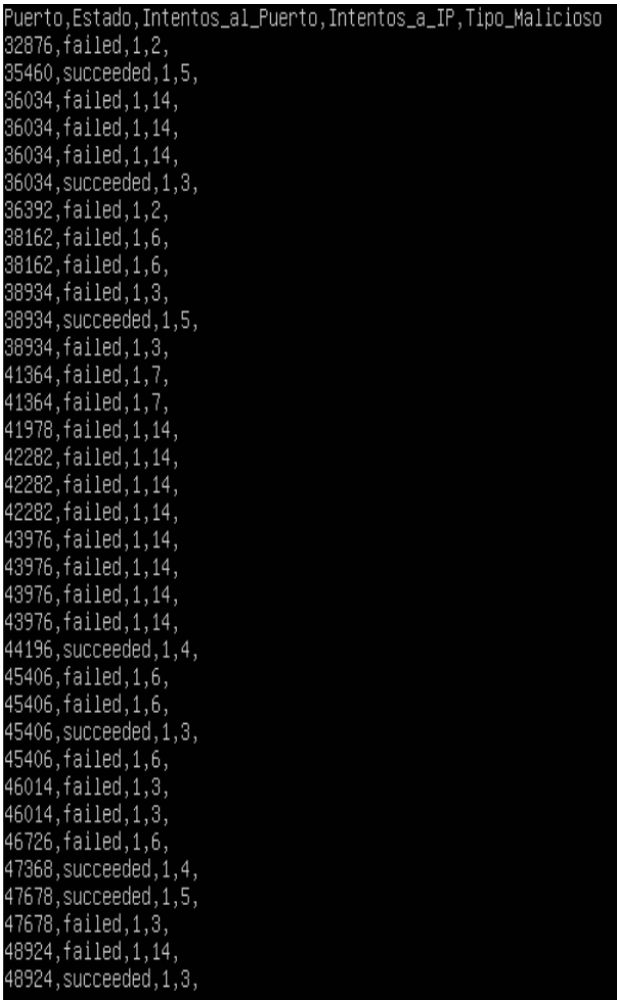

Figura 4.16: Archivo CSV con datos sin clasificar para entrenamiento en Machine Learning

Fuente: Desarrollado por el investigador

En el script converter se realiza la conversión de los datos a un formato entendible por Weka que debe ser editado al final, identificando cuales se deben considerar ataques y cuales no son consideras ataques de fuerza bruta. La figura 4.17 muestra la correcta ejecución del comando que formatea el archivo CSV en formato ARFF.

```
adrian@servidoradrian:~/prototype/honeycsv/trainfiles$ sudo java -cp /usr/share/java/weka.jar weka.c
ore.converters.CSVLoader /home/adrian/prototype/honeycsv/cowrieok.csv > /home/adrian/prototype/honey
csv/trainfiles/cowrieok.arff
---Registering Weka Editors---
```

Figura 4.17: Conversión exitosa de los datos de entrenamiento CSV a formato ARFF

Fuente: Desarrollado por el investigador

Se determina si es o no un ataque real, tomando en cuenta la cantidad de intentos de login de un puerto. Para un usuario normal se puede esperar que falle un máximo de 6 veces su

contraseña para luego tener un ingreso de sesión exitoso, sin embargo, considerando que los atacantes por lo general cambian de puerto constantemente para no ser descubiertos, también se utiliza el apartado de intentos por IP. Por lo que detectar el ataque se basará primordialmente en el número de este apartado. El archivo CSV queda completo con 5 columnas luego de agregar manualmente la columna 0,1 según el criterio antes expuesto, donde 0 representa que no es malicioso y 1 que es un intento de inicio de sesión malicioso como se visualiza en la figura 4.18.

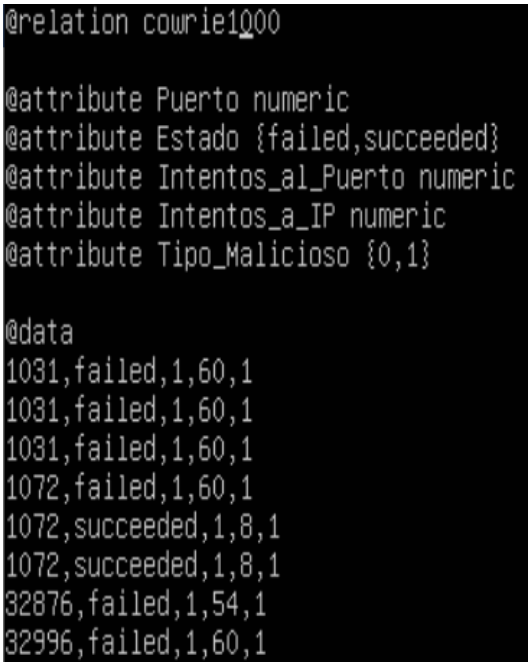

Figura 4.18: Datos formateados a ARFF clasificados con columna de "Tipo Malicioso"
Fuente: Desarrollado por el investigador

La matriz de confusión que se genera utiliza internamente validación cruzada, que realiza varias iteraciones para generar el modelo y compararlo, comenzando con 1% de los 500 datos ingresados para entrenamiento y 99% para prueba, para terminar en 99% en entrenamiento y 1% para prueba. Al final el algoritmo entrega el error generado total y la efectividad de bosques aleatorios. En la figura 4.19 se puede ver que el error es del 0,2% con solo un dato mal clasificado y 499 datos clasificados correctamente lo que equivale al 99.8% de efectividad.

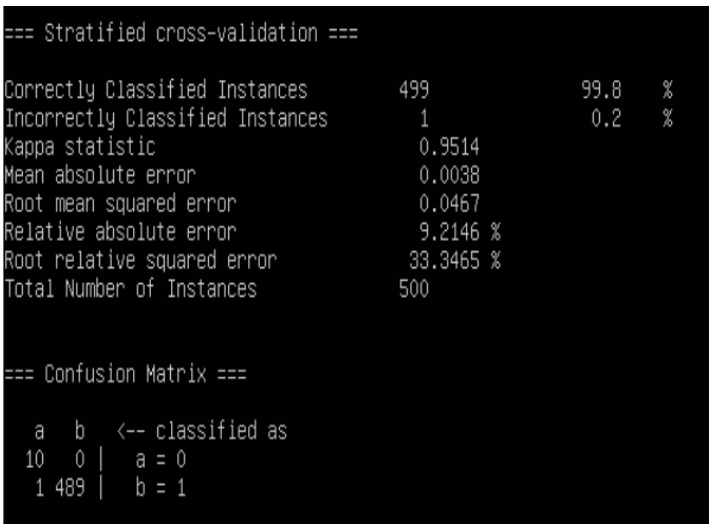

Figura 4.19: Matriz de confusión con bosques aleatorios

Fuente: Desarrollado por el investigador

Utilizando árboles de decisión se puede ver en la figura 4.20 que el error sube ligeramente al 0.4% y la efectividad baja al 99.5%, lo que sugiere trabajar con el modelo entrenado con bosques aleatorios por su efectividad del 99.8%.

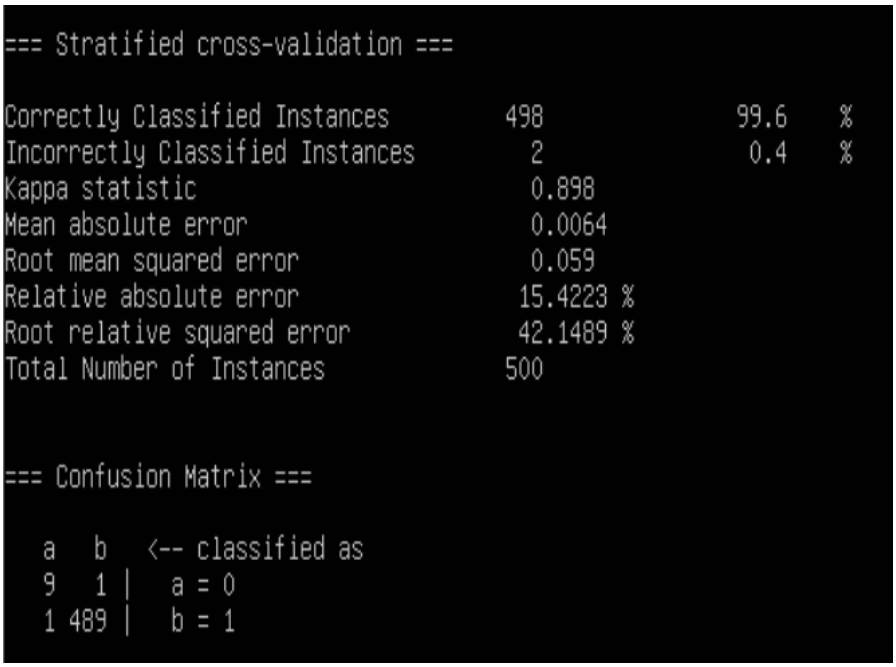

Figura 4.20: Matriz de confusión con árboles de decisión

Fuente: Desarrollado por el investigador

En la figura 4.21 se puede observar la eficacia y error para los modelos entrenados con árboles de decisión y bosques aleatorios respectivamente, el cual es ligeramente superior para el árbol de decisión.

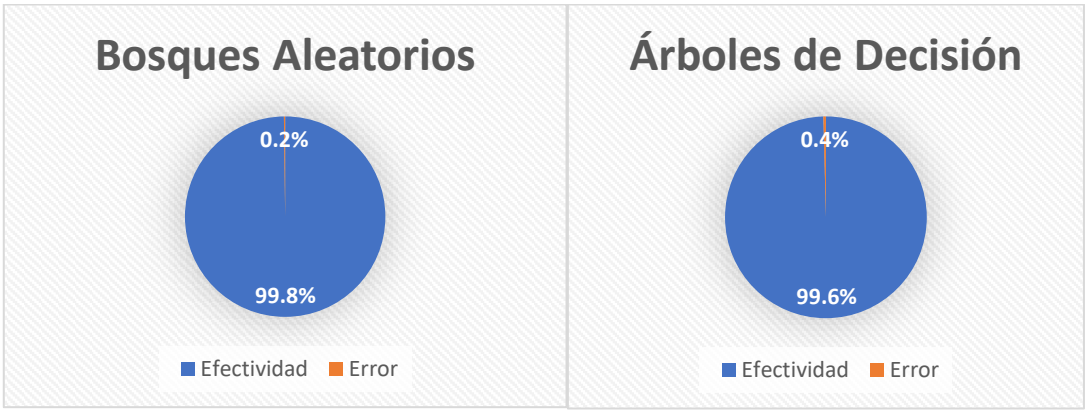

Figura 4.21: Diagrama de pastel de efectividad-error de Bosques aleatorios y Árboles de decisión

Fuente: Desarrollado por el investigador

4.3.2. Prueba del Sistema en Test (Clasificación de Amenazas)

La prueba del Sistema de detección de ataques de fuerza bruta en su fase de test, utiliza el modelo entrenado de machine learning con bosques aleatorios para clasificar de manera automática intentos de inicio de sesión al servidor Ubuntu.

Los scripts anteriores se reutilizan para la adquisición y filtrado de datos, a excepción del script en PERL, que ahora filtra únicamente los últimos 100 datos para mejorar la eficiencia del sistema de detección de ataques de fuerza bruta.

Los scripts de ejecución se repiten en bucle, por lo que el archivo de configuración de crontab se edita de tal manera que se ejecuten a cada minuto. En la figura 4.22 se puede ver que la rutina se programa para iniciar a cada minuto, de cada hora, de cada día y todos los meses mediante cinco asteriscos respectivamente.

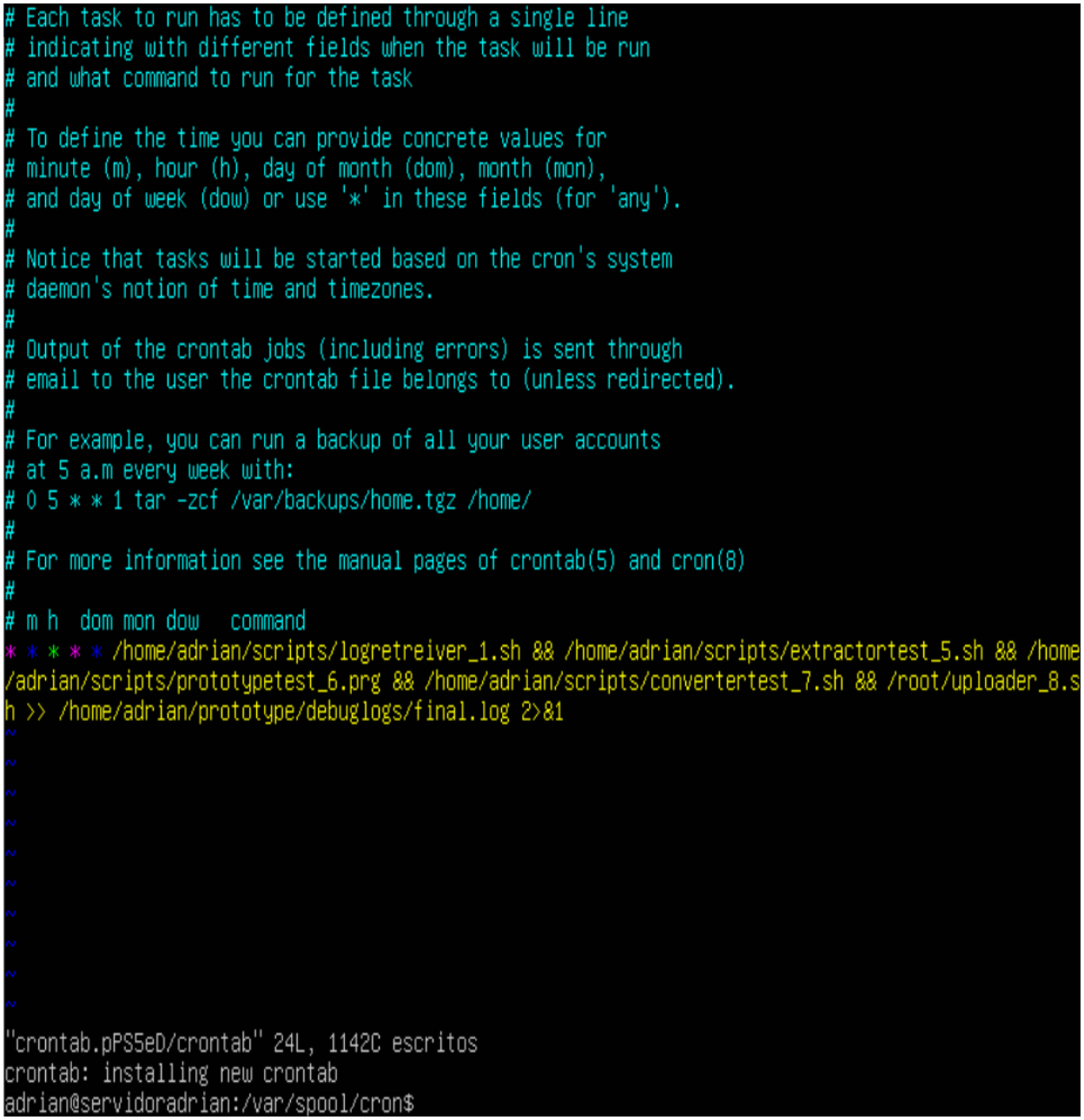

Figura 4.22: Automatización del Proceso de ejecución de Scripts

Fuente: Desarrollado por el investigador

4.3.3. Alerta de Amenazas

La alerta de amenazas se presenta por medio de una página web sencilla se presenta el aviso de amenazas, donde consta en pantalla la recepción de datos del servidor Ubuntu (Carga exitosa o fallida), la detección de amenazas (Posible amenaza detectada o sin amenazas), el estado del servidor (online u offline) y cada cuanto tiempo se está actualizando. En la figura 4.23 se puede verificar que el sistema de detección de ataques de fuerza bruta no ha detectado ninguna

amenaza, también presenta que la carga de los datos desde el servidor Ubuntu ha sido exitosa y que se encuentra en línea.

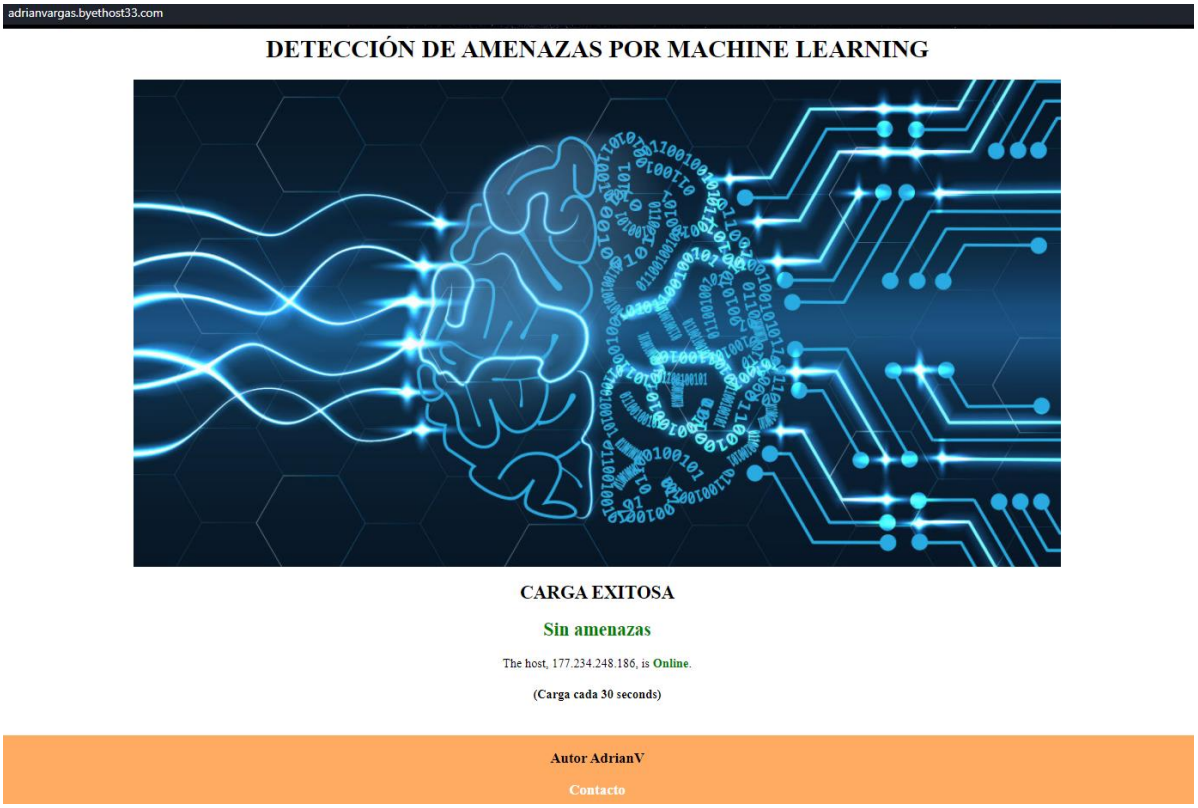

Figura 4.23: Página web de monitoreo, sin amenazas detectadas

Fuente: Desarrollado por el investigador

En la figura 4.24 el sistema ha detectado una amenaza recibida de ataque de fuerza bruta, indicando también que el servidor Ubuntu se encuentra en línea y la carga de los resultados clasificados por machine learning ha resultado exitosa.

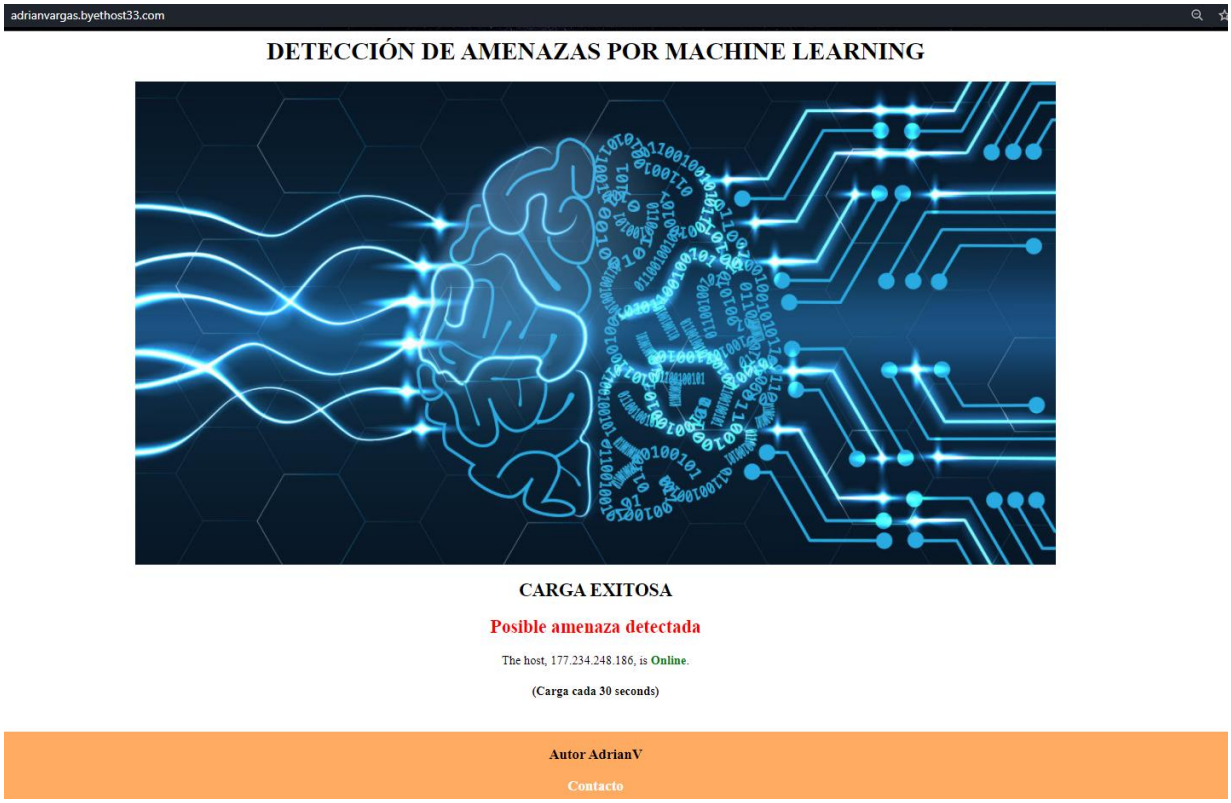

Figura 4.24: Página web de monitoreo, con posible amenaza detectada

Fuente: Desarrollado por el investigador

4.3.4 Pruebas de detección de amenazas

Las pruebas de detección de amenazas se realizan por medio de PuttyGen para generar tráfico benigno y tráfico malicioso, mediante inicios de sesión correctos e inicios de sesión incorrectos respectivamente.

En la figura 4.25 se puede visualizar del lado izquierdo al servidor Ubuntu, recibiendo mas de 6 intentos de inicio de sesión fallidos, en la parte derecha inferior la herramienta PuttyGen generando el tráfico malicioso y en la esquina superior derecha la página web se actualiza de manera automática mostrando en rojo a la amenaza de ataque de fuerza bruta detectada.

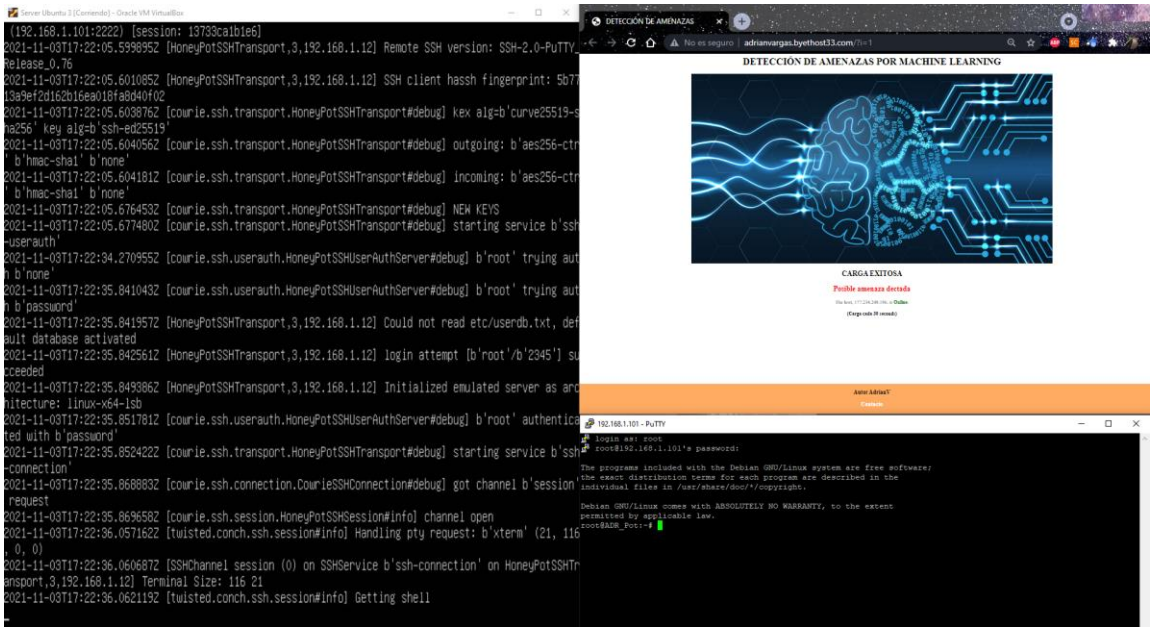

Figura 4.25: Prueba de detección de amenaza de red positiva

Fuente: Desarrollado por el investigador

En la figura 4.26 se puede visualizar del lado izquierdo al servidor Ubuntu, recibiendo más de 6 intentos de inicio de sesión exitosos, en la parte derecha inferior la herramienta PuttyGen generando el tráfico benigno y en la esquina superior derecha la página web se actualiza de manera automática mostrando en verde sin amenazas de ataque de fuerza bruta detectadas.

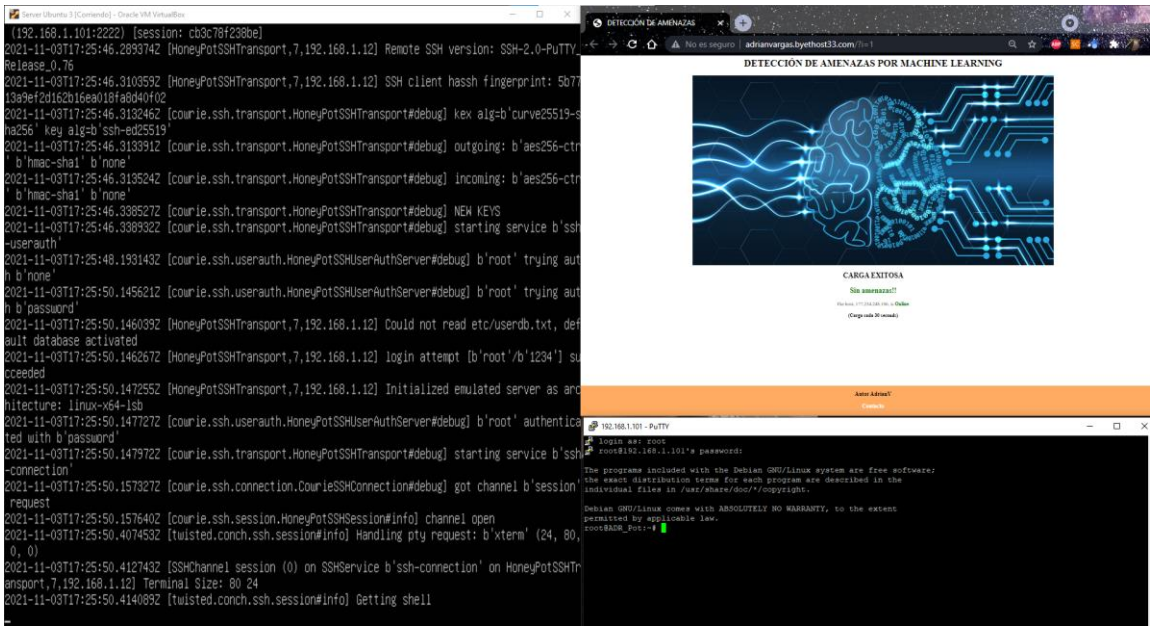

Figura 4.26: Prueba de detección de amenaza de red negativa

Fuente: Desarrollado por el investigador

4.4. Análisis de resultados

4.4.1. Análisis por Verificación Cruzada en Bosques Aleatorios

El análisis por verificación cruzada en bosques aleatorios indica un porcentaje menor de error con respecto al uso de árboles de decisión que fue del 0.2% y 0.4% respectivamente.

Con el entrenamiento y test del modelo desarrollado haciendo uso de Bosques Aleatorios hay una mejora con respecto a árboles de decisión en efectividad ya que se tiene un 99.8% en porcentaje de aciertos con el uso de bosques aleatorios. Lo que indica que utilizando Bosques Aleatorios se logró predecir 499 correctamente del total de 500 ingresados.

4.4.2. Análisis presupuestario

Para este proyecto se hizo una proyección del presupuesto a invertir para el desarrollo e implementación del sistema de detección de ataques de fuerza bruta, de aproximadamente $956 dólares, donde se considera un costo de $550 como gastos para el servidor que alberga al sistema y $50 dólares para cableado de red. Ambos costos se vieron favorecidos por el uso de servidores propios de la empresa donde se realizaron las pruebas, mismos que debido a sus características de aplicación tienen una gran capacidad de memoria RAM, atributo indispensable para el levantamiento del honeypot Cowrie en el equipo, reduciendo en un 20% los costos en este punto.

Para los equipos de red, se considera solamente la compra de dos switches de capa dos básico en $56, con lo que mejora la factibilidad de implementación en otros puntos de la empresa para verificar posibles ataques.

Tabla 4.2: Tabla del presupuesto económico requerido para la implementación del sistema de detección de amenazas de red

PRESUPUESTO					
Ítem	**Descripción**	**Unidad**	**Cantidad**	**Precio Unitario**	**Precio Total**
1	Servidor	1	c/u	$550.00	$550
2	Switch	2	c/u	$28.00	$56
3	Desarrollo	1	c/u	$200	$200
4	Cableado de red	10	c/u	$5	$50
5	Imprevistos				$100
	TOTAL				$956

Fuente: Desarrollado por el investigador

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Tras realizar la investigación se concluye que existen diferentes tipos de ataques a nivel de red, entre los principales Cross Site Scripting, ataques de inyección, falta de seguridad en la configuración y ataques de fuerza bruta que son los que más rondan la red pública de internet, que en dos semanas de escucha con un honeypot fácilmente supera los 8000 intentos de inicio de sesión.
- El modelo de Machine Learning, Bosques Aleatorios menora el error con respecto a árboles de decisión de 0.4% a 0.2%, por lo que se hace más relevante al momento de detectar bots de ataques de fuerza bruta a costa de un mayor número de iteraciones con respecto al algoritmo base de árboles de decisión. Esta consideración no se toma en cuenta debido al alcance del sistema que hace una clasificación booleana, lo que menora en impacto de procesamiento y uso de recursos.
- La solución a la detección de amenazas que se plantea detecta ataques de fuerza bruta en un entorno empresarial en un lapso de 30 segundos lo que le permite al administrador de red cerrar los puertos que más reciben ataques o mejorar las prácticas de seguridad.

5.2. Recomendaciones

- El sistema de clasificación detecta como verdadero o falso si hubo un ataque, por lo que se puede mejorar adecuando el algoritmo para detectar no solamente el tipo de ataque, sino también el riesgo en base a la IP y puerto de origen del atacante, lo que implicaría el uso de un equipo hosting con mayor capacidad de procesamiento y RAM.
- El sistema de Bosques Aleatorios se basa en una configuración estándar del número de iteraciones, dado que no se especifica un número exacto en la configuración. Una estimación profunda del número de Folds o iteraciones podría mejorar el error medio relativo del algoritmo para los datos que se analizan en el ámbito de redes.
- Utilizar el sistema de detección de intrusos por ataque de fuerza bruta, aplicando Deep Learning, sería la oportunidad para utilizar algoritmos no supervisados y redes neuronales, en función de menorar el error de clasificación.

Referencias

[1] Betz, C. (2019). Informe sobre amenazas 2019. CenturyLink. Disponible en: https://info.centurylinkforbusiness.com/rs/131-SYO-861/images/CenturyLink ReportSep2019%20_sp.pdf?aliId=eyJpIjoiMFpxaGp5dFliZlZvVStXZyIsInQiOiJ0bklTZlNYU mo3OFBlcHdFWmdYdVFnPT0ifQ%25253D%25253D.

[2] Russell, I., & Markov, Z. (2017, March). An introduction to the Weka data mining system. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 742-742).

[3] Ibáñez, K. (2017). Estudio Comparativo De Técnicas De Entrenamiento Y Clasificación En sistemas De Detección De Intrusos (Ids), Basados En Anomalías De Red. 12,13,14,22,23.

[4] Lau-Fernández, I. L.-F.-P.-G.-J.-M. (2009). Sistema de Detección de Intrusos de Red basado en. ResearchGate.

[5] Albayati, M., & Issac, B. (2015). Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system. International Journal of Computational Intelligence Systems, 8(5), 841-853.

[6] Ghafir, I., Hammoudeh, M., Prenosil, V., Han, L., Hegarty, R., Rabie, K., & Aparicio-Navarro, F. J. (2018). Detection of advanced persistent threat using machine-learning correlation analysis. Future Generation Computer Systems, 89, 349-359.

[7] Hechmi, J. M., Khlaifi, H., Bouatay, A., Zrelli, A., & Ezzedine, T. (2018, September). Intrusion Detection Using Data Fusion and Machine Learning. In 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp. 1-6). IEEE.

[8] Maleh, Y. (2020). Machine Learning Techniques for IoT Intrusions Detection in Aerospace Cyber-Physical Systems. In Machine Learning and Data Mining in Aerospace Technology (pp. 205-232). Springer, Cham.

[9] Bindra, N., & Sood, M. (2019). Detecting DDoS Attacks Using Machine Learning Techniques and Contemporary Intrusion Detection Dataset. Automatic Control and Computer Sciences, 53(5), 419-428.

[10] Agarwal, M., Pasumarthi, D., Biswas, S., & Nandi, S. (2016). Machine learning approach for detection of flooding DoS attacks in 802.11 networks and attacker localization. International Journal of Machine Learning and Cybernetics, 7(6), 1035-1051.

[11] Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications, 9(1), 16.

[12] Wang, G., Xu, M., Wu, Y., Zheng, N., Xu, J., & Qiao, T. (2018, August). Using machine learning for determining network robustness of multi-agent systems under attacks. In Pacific Rim International Conference on Artificial Intelligence (pp. 491-498). Springer, Cham.

[13] Cabrera, C. &. (2006). Firewalls Y Seguridad en Internet mediante IPCop. 5-9.

[14] [Alonzo, 2012] Alonzo, E. R. G. C. (2012). Hacking de Aplicaciones web: SQL Injection, chapter 1. 0xWord.

[15] Klenzi, R. O., & López, M. (2017, August). Detección de ataques DoS con herramientas de minería de datos. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

[16] Narváez, D., Romero, C., & Núñez, M. (2016). Evaluación de ataques de Denegación de servicio DoS y DDoS, y mecanismos de protección. GEEKS DECC-REPORTS, 2(1).

[17] Bautista Rosell, J. (2020). Ataques DDoS con IoT, Análisis y Prevención de Riesgos (Bachelor's thesis).

[18] Bolivar, T. N., Delgado, M. S. A., Flores, Y. A. T., & Almonte, F. U. R. (2021). Análisis y optimización del proceso de validación de ataques de secuencia de comandos en sitios cruzados (XSS) empleando

[19] Osorio, E. F. R., Zea, M. P. C., & Casanova, W. A. C. (2020). Evaluación de ataques DDoS y fuerza bruta utilizando entorno virtual Kali Linux como plataforma experimental. Dilemas contemporáneos: Educación, Política y Valores.

[20] Traberg, G., Molinari, L. H., Venosa, P., Macia, N., & Lanfranco, E. F. (2015). Automatizando el descubrimiento de portales de autenticación y evaluación de la seguridad mediante ataques de fuerza bruta en el marco de una auditoría de seguridad. In XXI Congreso Argentino de Ciencias de la Computación (Junín, 2015).

[21] Bolivar, T. N., Delgado, M. S. A., Flores, Y. A. T., & Almonte, F. U. R. (2021). Análisis y optimización del proceso de validación de ataques de secuencia de comandos en sitios cruzados (XSS) empleando Burp Suite para evadir medidas de seguridad. Revista Ibérica de Sistemas e Tecnologias de Informação, (E39), 414-432.

[22] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Machine learning basics. Deep learning, 1(7), 98-164.

[23] Piñón Blanco, C. (2021). Detección de ciberataques en entornos industriales mediante la aplicación de técnicas de Machine Learning (Doctoral dissertation).

[24] Despujol Zabala, I., & Martínez Navarro, J. Á. (2021). Machine Learning para la mejora de la experiencia con MOOC: el caso de la Universitat Politècnica de València.

[25] Russo, C., Ramón, H., Alonso, N., Cicerchia, B., Esnaola, L., & Tessore, J. P. (2016). Tratamiento masivo de datos utilizando técnicas de Machine Learning.

[26] Perez, C. (2005). Aplicación de redes Neuronales para la detección de intrusos en redes y sistemas de información. 2.

[27] Acevedo, E., Serna, A., & Serna, E. (2017). Principios y características de las redes neuronales artificiales. Desarrollo e innovación en ingeniería, 173.

[28] Acevedo, E., Serna, A., & Serna, E. (2017). Principios y características de las redes neuronales artificiales. Desarrollo e innovación en ingeniería, 173.

[29] [Pozi et al., 2016] Pozi, M. S. M., Sulaiman, M. N., Mustapha, N., and Perumal, T. (2016). Improving Anomalous Rare Attack Detection Rate for Intrusion Detection System Using Support Vector Machine and Genetic Programming. Neural Processing Letters, 44(2):279–290.

[30] Oviedo, B., Zambrano-Vega, C., & Gómez, J. G. (2019). Clasificador Bayesiano Simple aplicado al aprendizaje. Revista Ibérica de Sistemas e Tecnologías de Informação, (E18), 74-85.

[31] Pérez-Rave, J., & González Echavarría, F. (2018). Árboles de clasificación vs regresión logística en el desarrollo de competencias genéricas en ingeniería. Computación y Sistemas, 22(4), 1519-1541.

[32] [Pandeeswari and Kumar, 2016] Pandeeswari, N. and Kumar, G. (2016). Anomaly Detection System in Cloud Environment Using Fuzzy Clustering Based ANN. Mobile Networks and Applications, 21(3):494–505.

[33] Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., ... & Lin, C. T. (2017). A review of clustering techniques and developments. Neurocomputing, 267, 664-681.

[34] Zhao, Y., Nasrullah, Z., & Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. arXiv preprint arXiv:1901.01588.

[35] Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. Information, 11(4), 193.

[36] Ortego, D. (2017). Las 8 mejores herramientas open source de detección de intrusión. Obtenido de https://openwebinars.net/blog/las-8-mejores-herramientas-open-source-de-deteccion-de-intrusion/?cat=ethical-hacking

[37] Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. Future Generation Computer Systems, 80, 157-170.

[38] Zammit, D. (2016). A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data. University of Malta, 1-55.

[39] Zhang, D., & Wang, S. (2019, July). Optimization of traditional Snort intrusion detection system. In IOP Conference Series: Materials Science and Engineering (Vol. 569, No. 4, p. 042041). IOP Publishing.

ANEXOS

Anexo 1. Script de recepción de datos .SH

```
#!/bin/bash
#Codificación en UTF-8
# -*- ENCONDING: UTF-8 -*-
echo "Recibiendo Loggs"
echo "_____"
#Copia los datos del usuario cowrie al usuario principal
sudo cp /home/cowrie/cowrie/var/log/cowrie/cowrie.log /home/adrian/alllogs1.log
echo "Carga de logs de Cowrie recibido al...100%"
echo "Scripts corren satisfactoriamente :)"
exit
```

Anexo 2. Script de extracción de datos .SH

```
#!/bin/bash
# -*- ENCONDING: UTF-8 -*-
#copia los logs a la carpeta de logs
sudo cp /home/adrian/alllogs1.log /home/adrian/prototype/logs/
#extrae las últimas 500 coincidencias con las palabras “New connection” y “login attempt”
sudo grep 'New connection\|login attempt' /home/adrian/prototype/logs/alllogs1.log | uniq -u | tail -n 500 > /home/adrian/prototype/logs/extractedlogs/cowrieresult1.log
echo "Filtrado de Datos Exitoso"
exit
```

Anexo 3. Script de conversión a formato CSV .Perl

```
#!/usr/bin/perl -w
$path2 = "/home/adrian/prototype/logs/extractedlogs/cowrieresult1.log";
$cowrie = ">/home/adrian/prototype/honeycsv/cowrie1.csv";
$end = "";

sub cowrieExtractor(){
open(FILE2, $cowrie) or die "Can't open '$cowrie': $!";
open(LOG2, $path2) or die "Can't open '$path2': $!";

my (%rept, %ip_tot);
my ($ip, $port);

while (my $line = <LOG2>)
{
if ($line =~ /New connection/) {
($ip, $port) = $line =~ /New connection:\s+([^:]+):(\d+)/;
next;
}
elsif (!$ip or !$port) { next } # First lines come before New connection

my ($usr, $status) = $line =~ m/login attempt\s+\[([^\]]+)\]\s+(\w+)/;
if ($usr and $status) {
$rept{$port}{$ip}{$usr}{$status}++;
$ip_tot{$ip}{$status}++;
}
else { warn "Line with an unexpected format:\n$line" }
}
#Imprime el encabezado de los datos
print FILE2 "Puerto,Estado,Intentos_al_Puerto,Intentos_a_IP,Tipo_Malicioso\n";
```

```
foreach my $port (sort keys %rept) {
foreach my $ip (sort keys %{$rept{$port}}) {
foreach my $usr (sort keys %{$rept{$port}{$ip}}) {
foreach my $stat ( sort keys %{$rept{$port}{$ip}{$usr}} ) {
print FILE2 "$port,$stat,$rept{$port}{$ip}{$usr}{$stat}";
print FILE2 ",$ip_tot{$ip}{$stat},\n";
}
}
}
}

#prints IP and Number of Occurrences based on that IP for testing purposes

#print "\n";
#print "IP,Status,Occurences\n";
#foreach my $ip (sort keys %ip_tot) {
# foreach my $stat ( sort keys %{$ip_tot{$ip}} ) {
# print "$ip,$stat,$ip_tot{$ip}{$stat}\n";
# }
#}


# if all variables are not equal to null, then print to file
#if($ip && $port && $usr && $pass && $status ne ""){
#print FILE2 join ",",($ip, $port, $usr, $pass, $status);
#print FILE2 "\n";
}

#cuenta el número de incidencias por puerto e ip de origen
sub counter(){
$result = 0;
```

```
#open(FILE2, $cowrie) or die "Can't open '$cowrie': $!";
while(my $otherlines = <LOG2>){

if($otherlines =~ /login attempt/){
($user, $password) = (split /[\s:\[\]\/]+/, $otherlines)[-3,-2];
if($_[1] =~ /$user/ && $_[2] =~ /$password/){
$result++;
}#if ip matches i think i have to do this with split

#print "TEST\n";
}
#print "Combo $_[0] and $_[1]\n";

}
#print "$result";
return $result;
}

close(LOG);
close(FILE1);
close(FILE2);

close(LOG2);
close(FILE2);

cowrieExtractor();
print "Conteo y formateo correcto\n";
```

Anexo 4. Script de conversión a formato ARFF y Entrenamiento en WEKA .SH

```
#!/bin/bash
# -*- ENCONDING: UTF-8 -*-

#Convierte el archivo CSV a formato ARFF
#sudo java -cp /home/adrian/prototype/weka-3-9-0/weka.jar
weka.core.converters.CSVLoader /home/adrian/prototype/honeycsv/cowrie1.csv >
/home/adrian/prototype/honeycsv/trainfiles/cowrie1.arff

#crea el modelo de clasificación con los datos previamente identificados (0 para negativo, 1
para positivo)
sudo java -cp /home/adrian/prototype/weka-3-6-13/weka.jar weka.classifiers.trees.J48 -t
/home/adrian/prototype/honeycsv/trainfiles/cowrie1.arff -d
/home/adrian/prototype/honeycsv/models/cowrie1.model
exit
```

Anexo 5. Script de extracción de datos en test .SH

```
#!/bin/bash
# -*- ENCONDING: UTF-8 -*-
#copia los datos a la carpeta de logs
sudo cp /home/adrian/alllogs1.log /home/adrian/prototype/logs/
#extrae las últimas 500 coincidencias con las palabras “New connection” y “login attempt”
sudo grep 'New connection\|login attempt' /home/adrian/prototype/logs/alllogs1.log | uniq -u | tail -n 100 > /home/adrian/prototype/logs/extractedlogs/cowrieresulttest1.log
#limpieza de los datos previos a clasificar
sudo cp /home/adrian/prototype/honeycsv/testfiles/cowrietest.arff /home/adrian/prototype/honeycsv/testfiles/cowrietesttest1.arff
exit
```

Anexo 6. Script de conversión a formato CSV en test.Perl

```
#!/usr/bin/perl -w

$path2 = "/home/adrian/prototype/logs/extractedlogs/cowrieresulttest1.log";
$testcowrie = "+>/home/adrian/prototype/honeycsv/testcowrie1.log";
$cowrie = ">>/home/adrian/prototype/honeycsv/testfiles/cowrietesttest1.arff";

$end = "0";

sub cowrieExtractor(){
open(FILE2, $testcowrie) or die "Can't open '$testcowrie': $!";
open(LOG2, $path2) or die "Can't open '$path2': $!";
my (%rept, %ip_tot);
my ($ip, $port);
while (my $line = <LOG2>)
{
if ($line =~ /New connection/) {
($ip, $port) = $line =~ /New connection:\s+([^:]+):(\d+)/;
next;
}
elsif (!$ip or !$port) { next } # First lines come before New connection

my ($usr, $status) = $line =~ m/login attempt\s+\[([^\]]+)\]\s+(\w+)/;
if ($usr and $status) {
$rept{$port}{$ip}{$usr}{$status}++;
$ip_tot{$ip}{$status}++;
}
else { warn "Line with an unexpected format?:\n$line" }
}
#imprime el encabezado para el archivo CSV y realiza el conteo en una sola fila
```

```
print FILE2 "Puerto,estado,Intentos_al_puerto,Intentos_a_la_ip,Malicioso\n";
foreach my $port (sort keys %rept) {
foreach my $ip (sort keys %{$rept{$port}}) {
foreach my $usr (sort keys %{$rept{$port}{$ip}}) {
foreach my $stat ( sort keys %{$rept{$port}{$ip}{$usr}} ) {
print FILE2 "$port,$stat,$rept{$port}{$ip}{$usr}{$stat},";
print FILE2 "$ip_tot{$ip}{$stat},$end\n";
}
}
}
}
#close (FILE2);

#open (FILE2, $testcowrie);
 seek FILE2, 0, 0;
 chomp(my @lines = <FILE2>);
 my $last_one = pop @lines;

 open (FILE10, $cowrie) or die "Can't open '$cowrie': $!";

 print FILE10 "$last_one\n";
 close (FILE10);
 }
 close(LOG2);
 close(FILE2);
 cowrieExtractor();
```

Anexo 7. Script de conversión a formato ARFF y Clasificación en WEKA .SH

```
#!/bin/bash
# -*- ENCONDING: UTF-8 -*-
#copia el formato de encabezado para weka
sudo cp /home/adrian/prototype/honeycsv/testfiles/cowrietestformat.arff
/home/adrian/prototype/honeycsv/testfiles/cowrietestformat1.arff
#agrega el conteo de datos al encabezado
sudo cat /home/adrian/prototype/honeycsv/testfiles/cowrietesttest1.arff >>
/home/adrian/prototype/honeycsv/testfiles/cowrietestformat1.arff
#Clasifica los datos con el modelo generado y almacena la matriz de confusión en un archivo
de texto
java -cp /home/adrian/prototype/weka-3-6-13/weka.jar weka.classifiers.trees.RandomForest -
l /home/adrian/prototype/honeycsv/models/cowrie1.model -T
/home/adrian/prototype/honeycsv/testfiles/cowrietestformat1.arff >
/home/adrian/prototype/honeycsv/results/cowrie1.txt
copia los resultados al usurio root
sudo cp /home/adrian/prototype/honeycsv/results/cowrie1.txt /root/
echo "Operación Completa"
exit
```

Anexo 8. Script de subida de datos al Hosting Web .SH

```
#!/bin/bash
#envía los datos clasificados por FTP al hosting weg Byethost
curl -u b33_30022153:aeros3333 -T cowrie.txt ftp://ftp.byethost33.com/htdocs/
exit
```

Anexo 9. Script de página Web .Php

```
<html>
<head>
<title>DETECCIÓN DE AMENAZAS</title>
<link rel="icon" type="img/ico" href="images/favicon.jpg">
<meta http-equiv="refresh" content="30">
<style>
body {
background-image:url("https://img.freepik.com/vector-gratis/fondo-tecnologico-geometrico-color-azul_1055-1907.jpg?size=338&ext=jpg");
}
a:link {
color: white;
text-decoration: none;
}

/* visited link */
a:visited {
color: white;
text-decoration: none;
}

/* mouse over link */
a:hover {
color: black;
text-decoration: none;
}

/* selected link */
a:active {
```

```
color: yellow;
text-decoration: none;
}

footer
{

}

footer *
{
display: block;
}
#footer {
background:#ffab62;
width:100%;
height:100px;
position:absolute;
bottom:0;
left:0;
}
h3 {
text-align:center;
}
</style>
</head>
<body>
<center>
<h1>DETECCIÓN DE AMENAZAS POR MACHINE LEARNING</h1>
</center>
<p></p>
```

```
<p></p>
<p></p>
<p style="text-align:center"><img
src="https://www.ionos.es/digitalguide/fileadmin/DigitalGuide/Teaser/deep-learning-vs-machine-learning-t.jpg"
alt="It works!!"></p>
<p></p>
<p></p>

<div align="center">
<?php

$filename = "cowrie.txt";
$line = file($filename);



if(file_exists($filename)){
echo "<h2>Read through file and</h2>";
}
else{
echo "<h2>CARGA FALLIDA</h2>";
}

if(trim($line[15]) == "Correctly Classified Instances 0 0 %") {
echo "<h2><font color='red'>Posible amenaza dectada</font></h2>";
} else {
echo "<h2><font color='green'>Sin amenazas</font></h2>";
}

function pingAddress($ip) {
$pingresult = exec("/bin/ping -n 3 $ip", $outcome, $status);
```

```
if (0 == $status) {
$status = "<font color='green'><b>Online</b></font>.";
} else {
$status = "<font color='red'><b>Offline</b></font>.";
}
echo "The host, $ip, is ".$status;
}
pingAddress("177.234.248.186")
?>
<h4><font color='black'>(Carga cada 30 seconds)</h4>
</div>
<div id="footer">
<footer>
<h3>Autor AdrianV</h3>
<h3><a
href="https://mail.google.com/mail/u/0/?fs=1&to=vargasmadrian@gmail.com&su=Hola+Adrian!&tf=cm" target="_blank">Contacto</a></h3>
</footer>
</div>
</body>
</html>
```