UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

Sistema de Seguridad Basado en S.T.R.I.D.E (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) Utilizando Inteligencia Artificial

TESIS

Para obtener el grado de Doctor en Ingeniería con

Orientación en Tecnologías de la Información

PRESENTA

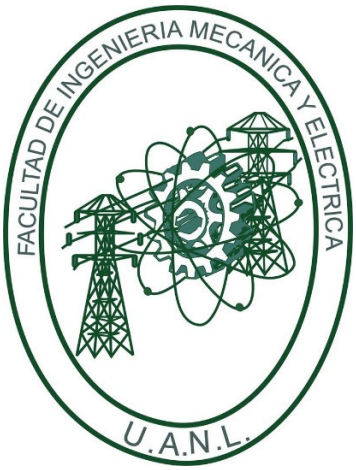
Sergio Antonio Ordoñez González

Ciudad Universitaria, San Nicolás de los Garza, N.L, octubre 2021

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



Sistema de Seguridad Basado en S.T.R.I.D.E (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) Utilizando Inteligencia Artificial

TESIS

Para obtener el grado de Doctor en Ingeniería con

Orientación en Tecnologías de la Información

PRESENTA

Sergio Antonio Ordoñez González

Ciudad Universitaria, San Nicolás de los Garza, N.L., octubre 2021

PRÓLOGO

En la actualidad la sociedad cada vez más está inmersa en procesos relacionados con la tecnología, Esta ha sido de gran ayuda para su crecimiento y desarrollo, estableciendo una colaboración y conexión en las actividades que hasta hace algunos años requerían esfuerzos que en ocasiones eran insuficientes para lograr la oportunidad y atención a las necesidades de una comunidad; de esta manera la tecnología y su propios dispositivos en hardware y software se ha convertido en un elemento clave y de atención para gobernantes, empresarios y científicos quienes han apostado a su desarrollo, sin embargo el reto ha sido mayor al encontrar embates que trasgreden la seguridad y confiabilidad de la misma.

Ante estos retos de seguridad en la información y con la clara necesidad de interactuar con una sociedad global que requiere de conexión, de todo con todos los seres humanos, la Inteligencia Artificial ha sido una área determinante para que a través de la generación de conocimiento digital se pueda generar aprendizaje y con ello operar modelos, mecanismos y sistemas que conllevan a la predicción de sucesos que sean fortalecidos o erradicados según sea el fin establecido, siendo de utilidad para realizar tareas de gran impacto y trascendencia en el manejo óptimo de las tecnologías, salvaguardando su correcta operación.

En este sentido la integridad de la información y sus procesos son una prioridad, ya que toda organización requiere una operación estable y segura para la toma de decisiones de manera oportuna, por lo que es de suma importancia contar con

mecanismos que al hacer uso de las tecnologías de información y comunicaciones, se asegure la protección de las acciones digitales que se realizan y que pudieran ser susceptibles a la vulnerabilidad causada por personas o instancias que pretenden dañar o revelar información dentro de la ilegalidad.

Por ello la importancia de la presente investigación donde muestra con bases científicas la determinación de un modelo de seguridad basado en Inteligencia Artificial para la predicción del comportamiento de transgresiones a la seguridad en red y con ello a la información y sus procesos, estableciendo patrones de comportamiento que lleven a la predicción para la prevención en la seguridad de los procesos digitales.

Finalmente, con esta investigación es posible conocer la problemática actual y retos que se tiene en el tema de seguridad en la tecnología, permitiendo comprobar que la Inteligencia Artificial es de suma importancia para la solución de la problemática dejando con este estudio una contribución científica que da valor y responde a los retos que hoy más que nunca enfrentamos al estar inmersos en la era digital, ligados a una gran evolución del conocimiento y uso de las tecnologías de información.

Dr. Jaime A. Castillo Elizondo

DEDICATORIAS

Este trabajo está dedicado primeramente a Dios, por darme la fuerza, las ganas y la motivación para poder concluir este periodo. A mi Esposa y mis hijos por tener paciencia en los tiempos que dediqué a esta investigación y no pude compartir con ellos, a mis Padres y hermanos por ser ese soporte y poner en mí el deseo de siempre seguir cumpliendo las metas.

A todos mis amigos por su apoyo, a mis maestros por sus enseñanzas y a mis alumnos por permitirme compartir con ellos el fruto de esta investigación.

Toda la gloria es para Dios.

AGRADECIMIENTOS

Quiero agradecer profundamente al Dr. Jaime A. Castillo Elizondo por el soporte, el gran ejemplo y la confianza depositada en mí, al Dr. Fernando Banda Muñoz por el apoyo para cumplir con estos retos, a mi asesor de Tesis el Dr. César Guerra por su tiempo invertido en mí, a Antonio Obregón, Joaquín Lara, Omar González y; Manuel Mendoza, alumnos que demostraron interés en la presente investigación y me acompañaron en el viaje del desarrollo de este proyecto.

Le agradezco de manera especial a mis maestros investigadores, a mis compañeros maestros, a el Ing. Hugo Lira por sus aportaciones a mi tesis, por la orientación y por la información que pudimos compartir para el desarrollo de esta Tesis.

Gracias a todos los que formaron parte de este extraordinario viaje.

Dios les bendiga.

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

En los últimos años y en especial en la última década, la tecnología, así como los desarrollos e innovaciones en el campo de la informática, han mejorado la forma en la cual, las personas y empresas se comunican entre sí, llegando a ser de gran importancia y utilidad, no solo en el campo social y personal, sino también en el laboral, llegando a ser no solo útiles y prácticas; sino completamente necesarias para que las empresas puedan competir en el mundo actual globalizado.

Son exactamente estas tecnologías el objetivo de ataques cibernéticos por personas conocidos como hackers, los cuales aprovechan sus conocimientos y capacidades, así como el uso inapropiado de la tecnología para buscar las vulnerabilidades de las organizaciones e infraestructuras, para posteriormente ingresar ilegalmente a información que es de carácter privado para cierto grupo de personas, con el fin de obtener un beneficio propio o de terceros.

Actualmente se ha sufrido en empresas el terrorismo electrónico de hackers, tales como los ataques a las redes sociales que se han perpetrado en los últimos años buscando denegar los servicios, o bien los ataques a empresas multimillonarias en busca de información clasificada para posteriormente asignarle un valor monetario a dicha información al mejor postor.

De tal manera, es imperativo ponerle un alto a este tipo de ataques, buscando una alternativa que nos permita conocer ciertos ataques y aprender a clasificar los mismos para estar lo mejor preparados posibles.

Una manera de llevar a cabo una correcta clasificación del tipo de información de paquetes informáticos, para observar si se trata de un ataque o no, es con el uso de las redes neuronales.

En la presente tesis, se busca el desarrollo de una tecnología alterna a las ya existentes, basada en una red neuronal como protección en una red computacional y un sistema clasificador de los diversos ataques existentes en diversos datasets (representación de datos residente en memoria que proporciona un modelo de programación relacional coherente independientemente del origen de datos que contiene). utilizados en el mundo de la seguridad.

Aunque pueden usarse muchos tipos de redes neuronales para clasificación, en este trabajo se usará una red multicapa llamada feedforward o perceptrón multicapa, que es el tipo de clasificador basado en redes neuronales más ampliamente estudiado y usado. La reciente actividad investigadora en la clasificación neuronal ha establecido que las redes neuronales son una alternativa prometedora a varios métodos de clasificación alternativos.

CAPÍTULO 1

1.1 Antecedentes

Durante los últimos años, las amenazas de Seguridad de la información han ido aumentando y diversificando su manera de vulnerar sistemas de información, según los Reportes de Riesgos Globales del Foro Mundial de Economía, se estima que el crecimiento continúe en ascenso debido a la proliferación de dispositivos que incluyen una conexión a internet. (World Economic Forum , 2020)

El robo y fraude de datos, así como los ciberataques se colocan entre el “top ten” de probabilidades de riesgo en este 2020, incluso por encima de indicadores como los son la crisis de agua y solo por debajo de elementos relacionados al impacto ambiental. (Forbes Mexico , 2019)

Las cifras del Reporte de Riesgos Globales en su Décimo quinta edición nos presentan un panorama donde más del 50% de la población a nivel mundial está ahora conectado a internet y establece que diariamente se suma un millón de personas a esta tendencia. Además, establece que dos terceras partes de la población mundial tienen a su alcance un dispositivo móvil, con lo cual la cantidad de usuarios con posibilidad de ser atacados se magnifica. (WEF, 2020)

Por otro lado, el desarrollo de la 4ta Revolución Industrial y su naturaleza digital hace de manera intrínsecamente vulnerable el uso de tecnología y la variante de métodos de robo de información, ransomware y sistemas inteligentes para producir ataques a gran escala, motivan a esta investigación para buscar establecer

parámetros de clasificación de ataques mediante sistemas inteligentes y utilizando herramientas creadas por grandes compañías como Microsoft, con la finalidad de aprender y adaptarse a las necesidades del ecosistema de ataques tan cambiante, buscando con esto satisfacer la necesidad de tener un sistema versátil de identificación de ataques por medio de patrones de comportamiento de los ataques previamente analizados.

Dentro de los elementos de la cuarta Revolución Industrial, ciberseguridad es uno de los principales elementos que en conjunto con el uso de modelos de Inteligencia Artificial facilitan las tareas para los sistemas empresariales e industriales más complejos, estos sectores han hecho crecer la relación de ambos ejes. El uso de la Inteligencia Artificial (IA) desde un enfoque básico puede ser sistemas de intrusión/detección/prevención (IDPS por sus siglas en inglés) consiste en una colección de archivos de los cuales sus datos son almacenados para analizarse posteriormente, la idea principal es que el IDPS (Pavlova, 2017), compara los datos de comportamiento buscando similitudes en patrones de ataques o actividades inusuales para mandar alertas al administrador de los sistemas. Los algoritmos genéticos, por otro lado, utilizan el concepto de la teoría de la evolución que a través de la selección natural los genes más fuertes u óptimos suelen quedar para nuevas generaciones, produciendo que el algoritmo sea más eficiente y complejo, existen ejercicios donde se usan 57 genes (W. Li, 2004), donde cada uno representa una característica, por ejemplo, la dirección IP o su destino, el puerto, protocolo de comunicación etc. Dependiendo de la forma en que los paquetes son enviados en sistemas de redes (M. Luck, 2003) (A. Goyal, 2007), el algoritmo adopta en base a

iteraciones o generaciones la manera en que sea una comunicación segura y equilibrada evitando conexiones intrusivas basados en una función de evaluación.

Actualmente el uso de la IA en la ciberseguridad también incluye el desarrollo de agentes inteligentes basados en los aspectos indicados en la norma ISO/IEC 27001 y la COBIT 5 para la seguridad de la información y el marco de mitigación de riesgo. (E. Jhordany, 2020).

1.2 Planteamiento del problema

Cada día es más común el uso de nuevas tecnologías por el gran conocimiento tecnológico e informático que se ha generado en los últimos años. Estos avances mejoran aspectos en el ámbito personal, social y laboral; por lo que es notable que cada día las empresas y organizaciones dependen en mayor medida de la información, tecnologías y sus comunicaciones. La información es uno de los activos más importantes de las organizaciones, y especialmente para algunas compañías que operan en determinados sectores de actividad en donde este es su principal y único recurso (de Pablos Heredero, 2004).

Es por eso, que algunos ciberdelincuentes buscan la manera de obtener algún tipo de beneficio propio, llevando a cabo lo que se conoce como ataques informáticos, para obtener dicha información. Un ataque informático consiste en aprovechar alguna debilidad o falla (vulnerabilidad) en el software, en el hardware, e incluso, en las personas que forman parte de un ambiente informático; todo esto a fin de obtener un beneficio, por lo general de índole económico, causando un efecto

negativo en la seguridad del sistema, que luego repercute directamente en los activos de la organización (Mieres, 2009).

Estos delincuentes son llamados hackers. Si bien, un hacker puede ser definido como cualquier persona a la que le apasiona el conocimiento, el descubrimiento, el aprendizaje y el funcionamiento de las cosas (Jara & Pacheco, 2012), hay otras definiciones que son más populares, en las cuales se define a un hacker como un delincuente informático que por razones diversas se dedican a servir a organizaciones delincuenciales, la subversión o intereses económicos propios (Gacharná G., 2009).

En este contexto se puede apreciar que, las vulnerabilidades en los sistemas informáticos han variado con el paso del tiempo; por ejemplo, el Informe de Seguridad Anual 2018 de Cisco Systems, una de las empresas más importantes en el área de telecomunicaciones a nivel mundial, revela que las amenazas diseñadas para aprovechar la confianza de los usuarios en sistemas, aplicaciones y redes personales han alcanzado niveles asombrosos. De acuerdo con el reporte, la falta de casi un millón de profesionales expertos en seguridad a nivel mundial está impactando las habilidades de las organizaciones de monitorear y asegurar las redes, mientras las vulnerabilidades y amenazas en general alcanzaron sus niveles más altos desde el año 2000. (CISCO, 2018)

Las conclusiones del informe ofrecen una imagen clara de los desafíos con respecto de los retos en seguridad que evolucionan rápidamente y que enfrentan hoy las empresas, los departamentos de TI y los individuos. Los métodos de los atacantes incluyen robo socialmente planeado de contraseñas y acreditaciones, infiltraciones

escondidas a simple vista, y explotación de la confianza requerida para transacciones económicas, servicios de gobierno e interacciones sociales.

Dentro de los hechos más relevantes del informe se encuentran las siguientes aportaciones:

- ***Los adversarios están llevando el malware a niveles de sofisticación e impacto sin precedentes*.** La evolución del malware, así como la aparición de los cryptoworms ransomware basados en la red elimina la necesidad del elemento humano en el lanzamiento de campañas de ransomware. Y para algunos adversarios, el premio no es un rescate, sino la eliminación de sistemas y datos, el malware de auto propagación es peligroso y tiene el potencial de acabar con Internet.

- ***Los adversarios son cada vez más expertos en la evasión y en usar como armas los servicios de la nube y otras tecnologías utilizadas con fines legítimos.*** La encriptación está destinada a mejorar la seguridad, pero también proporciona a los actores maliciosos una poderosa herramienta para ocultar la actividad de comando y control (C2), lo que les brinda más tiempo para operar e infligir daños.

- ***Los adversarios están explotando grietas en la seguridad, muchas de las cuales surgen de la expansión del Internet de las Cosas (IoT) y del uso de los servicios de la nube*.** Los defensores están desplegando dispositivos de

IoT a un paso rápido, pero a menudo prestan poca atención a la seguridad de estos sistemas. Los dispositivos IoT sin parche y no monitoreados brindan a los atacantes la oportunidad de infiltrarse en las redes. Una investigación sugiere que las organizaciones con dispositivos IoT susceptibles a ataques también parecen desmotivadas para acelerar las correcciones.

El CISO (Chied Information Security Officer), o Directores de Seguridad de la Información que fueron entrevistados para el Estudio de Referencia de las Capacidades de Seguridad de Cisco 2018, informaron que están ansiosos por agregar herramientas que usan Inteligencia Artificial y aprendizaje de máquina, y creen que su infraestructura de seguridad está creciendo en sofisticación e inteligencia. Sin embargo, también se sienten frustrados por la cantidad de falsos positivos que generan dichos sistemas, ya que los falsos positivos aumentan la carga de trabajo del equipo de seguridad. (CISCO, 2018)

Por lo anterior, las evoluciones de las estrategias para combatir los ataques informáticos han sido optimizadas a través del tiempo. Las primeras estrategias se dieron en los años 70 basadas en la seguridad física de los equipos, lo cual era muy poco confiable debido a que no se tenía información de que tipo de riesgos existían.

Posteriormente, en la década de los 80´s aparecen los primeros virus informáticos y con ellos se utilizaron herramientas basadas en firmas digitales llamados antivirus los cuales lograron darles un sentimiento de seguridad a los usuarios. En la década de los 90´s aparecen los primeros ataques por medio de internet, el crecimiento en el uso de medios extraíbles hace el proceso de seguridad vulnerable.

Para el año 2000 inician los ataques principalmente enfocados en las herramientas que se dedican a proteger la información, el uso de redes sociales crece de manera exponencial y se llevan a cabo los primeros fraudes online.

A partir del 2010 las diversas formas en las que los usuarios tienen acceso a Internet y sus recursos, exponen de manera considerable la posibilidad y las modalidades de ataques, con la accesibilidad de las personas a dispositivos móviles, los cuales no cumplen con los requisitos de seguridad necesarios para las modalidades actuales de robo de información, por lo que se opta por el uso de legislaciones adecuadas para proteger información e infraestructuras, el cifrado de información, así como el uso de normativas de privacidad va en aumento.

En la última década el uso de la Inteligencia Artificial ha sido utilizado como un recurso para prevenir los ataques hacia los sistemas computacionales, proporcionando algunas ventajas con respecto a los métodos tradicionales como menciona Tyugu en su artículo “Artificial Intelligence in Cyber Defense” en 2011, la cantidad de información que se usa en la actualidad y la velocidad de procesamiento de la información, hace más complicado que el software tradicional basado en tomar decisiones de menor complejidad pueda cumplir la función de lograr el objetivo de mantener íntegros los sistemas, por esto la Inteligencia Artificial toma un papel fundamental la cual se ajusta a las necesidades actuales. (Tyugu, 2011)

Por lo anterior, buscando resolver necesidades contemporáneas en el uso de sistemas inteligentes, se utilizará una red neuronal como protección en una red computacional en una red multicapa Feedforward o perceptrón multicapa que es el

tipo de clasificador basado en redes neuronales más ampliamente estudiado y usado. La reciente actividad investigadora en la clasificación neuronal ha establecido que las redes neuronales son una alternativa prometedora a varios métodos de clasificación alternativos.

1.3 Estado del Arte

La evolución de los sistemas de seguridad ha tomado un cambio radical dentro del entorno de empresarial, la información expuesta dentro de las redes de comunicación tomo un papel esencial para las organizaciones y sus estrategias para conquistar el mercado global. Así mismo, el uso diario de información en los diversos dispositivos de uso doméstico ha incrementado de manera exponencial los riesgos a los que nos enfrentamos como usuarios de internet

Los pronósticos de Cisco Systems líder en telecomunicaciones a nivel Mundial se previeron que en el 2020 existieron cerca de 50 mil millones de dispositivos conectados a la red, con lo cual las amenazas se magnificaron. (CISCO , 2020)
En la actualidad, existen diversas soluciones para la protección de sistemas de información las cuales se dividen en diversos tipos dependiendo de la tecnología utilizada para dar protección a los sistemas.

Las exigencias en los nuevos sistemas de seguridad se han visto en la necesidad de no tomar la decisión humana como la última fuente de seguridad, los sistemas basados en firmas no cuentan con la suficiente eficiencia y rapidez para proteger los sistemas para lo cual se ha optado por utilizar sistemas inteligentes que sean

capaces de tomar decisiones en tiempo real y las cuales se enumeran a continuación:

Inteligencia Artificial: Charniak definió en 1985 la Inteligencia Artificial como el estudio de las capacidades mentales a través del uso de modelos computacionales. (Charniak, Eugene, Mcdermott, Drew, 1986)

Wenke Lee propone los IDS (Intrusion Detection System) tradicionales basados en firmas, los cuales cuentan con la limitación de solo detectar ataques conocidos, así mismo, el Machine Learning y la Inteligencia Artificial toman fuerza como opciones que proveen la habilidad de aprender a las computadoras sin ser programados para un objetivo en específico. (Wenke Lee, S. J. Stolfo and K. W. Mok, 1999)

La detección de Intrusos en la red se divide y se clasifica según las herramientas utilizadas y los métodos de detección de intrusos empleados: destacan el uso de Machine Learning, Data Sets, Lógica Difusa, K- Nearest Neighboors, Naive Bayes, Arboles de Decisión, Maquinas de Vectores, Random Forest, entre otros.

Machine Learning, por ejemplo, trata con la construcción y estudio de algoritmos que pueden generalizar sets limitados de datos. Cada algoritmo opera construyendo modelos basados en entradas y usando estos modelos para hacer predicciones o decisiones, mejorando el solo seguir instrucciones explícitamente programadas.

Los algoritmos híbridos combinan algunas de las soluciones antes mencionadas con la finalidad de buscar cubrir los huecos de seguridad que un solo algoritmo no

podría cubrir en su totalidad, el combinarlos permite establecer utilizando la combinación de funciones un menor índice de falla.

Kazenko menciona múltiples modelos de aprendizaje se ha demostrado teórica y empíricamente que se proporciona significativamente mejor rendimiento que sus contrapartes de modelo base único. Los métodos híbridos de razonamiento han encontrado su aplicación en varios problemas verbales que van desde el reconocimiento de personas hasta el diagnóstico médico y clasificación de texto para pronósticos financieros.

Lo cual los convierten en un método prometedor para la detección de intrusos dada la variedad de ataques con los que se puede enfrentar un sistema de seguridad. (O. Cordon, P. Kazienko and B. Trawinski, 2011)

Otros métodos utilizados según la literatura consultada es el uso de Data Sets, el más utilizado como forma de estudio son conocidos como DARPA 1998 y DARPA 1999, desarrollados por el Laboratorio MIT Lincoln, estos datasets fueron utilizados durante la copa KDD competición anual de minería de datos y descubrimiento de conocimiento organizado por el Special Interest Group on Knowledge Discovery and Data Mining de ACM. Este Data Set consiste en cerca de 5 millones de registros de conexiones entrenadas, etiquetadas como una intrusión o una no intrusión y separa datasets de prueba para verificar si se analiza o no un ataque. (Fertalj, 2015)

Los métodos de Evaluación de los IDS para cada posible prueba de valores existen dos tipos de error: falso positivo (FP) y Falso Negativo (FN). FP ocurre cuando un evento es predicho como intruso, pero es un hecho normal, mientras FN ocurre

cuando un evento de verdadero intruso ocurre sin ser reconocido como uno. De otra manera, True Positive (TP) mide la proporción de los positivos actuales los cuales son correctamente identificados como tal, mientras el True Negative mide la proporción de negativos que fueron correctamente identificados como tal. La presentación de cada clasificador puede ser cuantificada usando la tasa de detección (DR) y overall Accuracy (OA). (Fertalj, 2015)

Por su parte, las Redes Neuronales o Redes Neuronales Artificiales comúnmente se refieren a un perceptrón multicapa, el cual funciona como un algoritmo de aprendizaje supervisado basado en la retroalimentación red neuronal con una o más capas entre entrada y capa de salida. Este tipo de red se entrena con el error de propagación hacia atrás algoritmo de aprendizaje. La ventaja y el poder de estos algoritmos se basan en su capacidad de representar tanto relaciones lineales como no lineales y en su capacidad de aprender estas relaciones directamente de los datos que se modelan.

Ghosh y col. Utilizaron perceptrones multicapas para la detección de anomalías, donde el modelo propuesto es una sola neural red de capa oculta. El rendimiento de este modelo probado en el conjunto de datos DARPA 1998 fue una Tasa de Detección del 77% con 2.2% Falsos Positivos. (A. K. Ghosh, and A. Schwartzbard, 1999)

Los Algoritmos Genéticos son algoritmos de búsqueda que funciona de manera similar al proceso de selección natural. Comienza con un conjunto de muestras de posibles soluciones que luego evoluciona hacia un conjunto de mejores soluciones. El algoritmo modifica repetidamente una población de soluciones individuales.

Durante generaciones sucesivas, la población "evoluciona" hacia una solución óptima.

Wei Li describe una técnica de aplicación del algoritmo genético (GA) a los sistemas de detección de intrusiones en red (IDS). Presenta una descripción del sistema de detección de intrusiones, algoritmo genético y técnicas de detección relacionadas en su artículo “Using Genetic Algorithm for Network Intrusion Detection”. A diferencia de otras implementaciones del mismo problema, esta implementación considera la información tanto temporal como espacial de las conexiones de red en codificar la información de conexión de red en reglas en IDS. Esto es útil para la identificación de complejos comportamientos anómalos. Este trabajo se centra en los protocolos de red TCP / IP. (Li, 2004)

La lógica difusa (FL) es una forma de lógica de muchos valores. Donde se trata de un razonamiento aproximado que fijo y exacto. Los defensores de FL afirman esta generalidad permite una mayor flexibilidad, libertad, precisión y compacidad al representar situaciones del mundo real. Todas las propiedades habituales del álgebra booleana se pueden extender a FL, y el grado de probabilidad de confianza en un booleano. Los defensores de FL afirman esta generalidad permite una mayor flexibilidad, libertad, precisión y compacidad al representar situaciones del mundo real. Todas las propiedades habituales del álgebra booleana se pueden extender a FL, y el grado de probabilidad de confianza en un booleano. “A Hybrid Machine Learning And Fuzzy Logic Approach to Cit Diagnostic Development” es un ejemplo en el que John Williams proporciona una forma de principios para codificar el conocimiento experto o heurística en algoritmos que imitan el enfoque de un

humano para resolver problemas desafiantes al pesar diferentes fuentes de información y emitir juicios basados en preponderancia de la evidencia. (J. K. Williams, J. Craig, A. Cotter, and J. K. Wolff, 2007)

Chimphlee logró tener porcentajes de Evaluación superiores al 93% de tasa de detección utilizando Lógica Difusa y el KDD99.Los Data Sets más utilizados para la detección de intrusos son los proporcionados por los laboratorios MIT Lincoln, patrocinados por el DARPA y la fuerza Aérea de Estados Unidos.

Estos Data Sets consisten en cerca de 5 millones de registros de conexiones entrenadas etiquetados como intrusión o no intrusión y separar datasets de prueba que consisten en ver y no ver ataques. (W. Chimphlee, A. H. Abdullah, M. N. M. Sap, S. Srinoy, and S. Chimphlee, 2006)

Wenke Lee en su investigación titulada “Mining in a Data-flow Environment: Experience in Network Intrusion Detection” utiliza estos Data Sets para monitorear el flujo de datos ya sea en datos de bajo nivel y propone algunas soluciones para el uso de datos de alto nivel o estructurados, buscando establecer modelos para el flujo de detecciones de intrusión en tiempo real. (W. Lee, S. J. Stolfo, and K. W. Mok, 1999.)

Así mismo existen diferentes métodos como los mencionados por Zhang Ma que utiliza K Nearest Neighbours obteniendo una Tasa de Detección de Intrusiones del 90.28, Liao utilizo este mismo método de KNN obteniendo tasas de Detección de Intrusiones de 91.7%. Panda y col. propuso IDS basados en Naive Bayes y logró un

DR de 94.90% en el conjunto de datos KDD’99. Amor hizo una propuesta similar y logró 91.52% DR.

Zhang alcanzó un DR promedio de 92.58% en conjunto de datos KDD'99 original y 99.86% en conjunto de datos equilibrado, donde las clases minoritarias se han sobre muestreado y la mayoría de las clases disminuyeron para aumentar la DR de intrusiones minoritarias. (Z. Ma and A. Kaban, 2013)

Después del análisis de literatura la presente investigación considera el uso de la muestra KDDCup99 en conjunto con el uso de Redes Neuronales Artificiales de Feedforward, las cuales se describirán a detalle más adelante.

1.4 Propuesta y alcance

La construcción de un modelo de seguridad basado en Inteligencia Artificial, está fundamentado en la versatilidad que proveen las redes neuronales artificiales, la manipulación de información, el uso de diversos métodos de entrenamiento, las Redes Neuronales Artificiales de Backpropagation utilizados en el presente artículo, nos permiten establecer el camino para identificar el comportamiento de los ataques de red establecidos por data sets previamente elaborados por diversos organismos encargados de análisis de riesgos.

El alcance esperado, consiste en crear un mecanismo de detección de Vulnerabilidades, buscando catalogar mediante el protocolo de clasificación STRIDE de Microsoft, los diversos ataques para identificar el tipo de ataque que se

está llevando a cabo con la posibilidad de predecir y tomar medidas de prevención en nuestros sistemas de seguridad.

1.5 Organización de la tesis

1.5.1 Justificación

El beneficio de trabajar con nuevas técnicas de Inteligencia Artificial y utilizando recursos de entrenamiento preestablecidos, nos da la ventaja de poder robustecer nuestro sistema de seguridad, el uso de tecnologías recientes y comprobadas en cuanto a su efectividad abre nuevos caminos para el combate a las amenazas que varían de manera tan voraz, que el estar creando contenido validado por el creador, sería una tarea caótica para los encargados del sistema. Es por eso por lo que en el presente trabajo se propone un modelo prototipo de solución que pueda adaptarse a las necesidades actuales de seguridad y además propone un sistema abierto para poder seguir incrementando sus capacidades de reacción ante las diversas amenazas a las que nos enfrentamos hoy en día.

1.6 Preguntas de investigación

Una forma de poder visualizar el alcance del aporte de la presente tesis es planteándonos las siguientes preguntas de investigación:

1.- ¿Es posible construir un análisis de vulnerabilidades a través de un histórico de Ataques?

2.- ¿El analizar y optimizar históricos de Ataques nos permite encontrar patrones de comportamiento de estos?

3.- ¿Se puede elegir alguna metodología de Clasificación de estos datos?

4.- ¿Es posible Clasificar y entrenar Redes Neuronales para aprender los ataques acordes a su comportamiento y preclasificación?

5.- ¿Se podrá optimizar una Red Neuronal Artificial hasta obtener el método óptimo y de mayor porcentaje de acierto?

1.6.1 Objetivo General

Proponer un modelo que cumpla con la encomienda de identificar ataques de manera automatizada utilizando herramientas de clasificación y datos preestablecidos y con esto reducir el índice de error al momento de identificar ataques, mediante el uso de Inteligencia Artificial.

1.6.2 Objetivos Específicos

Con el afán de poder dar respuesta a las preguntas de investigación se plasmaron los siguientes objetivos:

1.- Analizar Dataset de Ataques

2.-Optimizar Dataset de Ataques con la finalidad de reconocer patrones

3.- Elegir Metodología de Clasificación

4.- Clasificar y entrenar Red Neuronal de Backpropagation para aprender la mayor cantidad de ataques acorde a su comportamiento y clasificación.

5.- Reentrenar la RNA hasta obtener el método óptimo y de mayor porcentaje de acierto.

1.6.3 Hipótesis

Al plantear las preguntas de investigación junto a los objetivos específicos nos permite visualizar un alcance más objetivo de nuestra investigación y formular las siguientes hipótesis:

Hipótesis

Un sistema de seguridad entrenado con una Red Neuronal Artificial de Feedforward o perceptrón puede aumentar la eficiencia de un sistema de seguridad entrenado mediante clasificación STRIDE, reduciendo así el índice de Error con un entrenamiento adecuado.

Hipótesis Nula

Un Sistema de Seguridad entrenado con una Red Neuronal Artificial de Feedforward o perceptrón no puede aumentar la eficiencia de un sistema de seguridad entrenado mediante clasificación STRIDE, para Reducir el índice de Error con un entrenamiento adecuado.

La estructura de este artículo está compuesta de 5 capítulos que están ordenados de la siguiente manera:

Dentro del Capítulo 1 se establecen las bases y lineamientos que sustentan esta investigación, además de establecer la fundamentación teórica y el objetivo de realizar la misma.

El Capítulo 2 cuenta con el marco teórico dividido en 3 secciones principales: Uso de Redes Neuronales, Sistemas de Seguridad de la Información y Sistemas de

Seguridad utilizando Inteligencia Artificial. En el Capítulo 3 se desarrolla el prototipo de experimentación, así mismo, se expone la metodología utilizada para este proceso de investigación, de igual manera el proceso de experimentación puesto en práctica, utilizando el Capítulo 4 para exponer los resultados obtenidos, así como, las complicaciones identificadas durante el proceso de experimentación. En el Capítulo 5 se proponen las conclusiones de este estudio y se prevén trabajos futuros, utilizando esta misma línea de investigación con la finalidad de mejorar los resultados obtenidos.

Comentarios finales

El presente capítulo nos muestra el panorama general de la situación actual de la ciberseguridad a nivel mundial, podemos comparar diversos trabajos realizados con antelación, buscando optimizar los porcentajes de detección de vulnerabilidades y poder fortalecer el sistema de seguridad planteado. Este capítulo nos presenta un gran reto debido a la necesidad de adaptación de los sistemas de seguridad con respecto al número tan variado de ataques, así como, a la innovación de los mismos.

CAPÍTULO 2. Marco Teórico

2.1 Inteligencia Artificial

Historia

La Inteligencia Artificial como campo de investigación científica es casi tan antigua como las computadoras electrónicas. Una posibilidad de construir dispositivos, software, sistemas más inteligentes que humanos seres ha sido desde los primeros días de la IA "en el horizonte". El problema es que el horizonte temporal se aleja cuando pasa el tiempo.

Identificar el comienzo de la investigación de IA es complicado. George Boole (1815-1864) tenía muchas ideas sobre el análisis matemático de los procesos de pensamiento, y el campo aún conserva varias de sus ideas. Sin embargo, como no tenía computadora, se descarta a Boole como el fundador de AI.

Así como los historiadores alrededor del mundo tienen opiniones diferentes sobre quién construyó la primera computadora programable, también divergen sobre los orígenes de la IA. Los historiadores británicos señalan el artículo de Alan Turing de 1950 que definió lo que ahora se conoce como la prueba de Turing para determinar si una computadora muestra inteligencia. Los historiadores estadounidenses prefieren señalar la conferencia de Dartmouth de 1956, que se facturó explícitamente como un estudio de IA y es se cree que fue la fuente del término "Inteligencia Artificial".

El Espectro del comportamiento Inteligente

Una dificultad con la definición simple de IA es que deja la noción de inteligencia bastante vaga. La Figura 2.1 Ayuda a aclarar el asunto al mostrar un espectro de comportamientos inteligentes basados en el nivel de comprensión involucrado. Los comportamientos de nivel más bajo incluyen reacciones instintivas, como retirar una mano de un objeto caliente o esquivar un proyectil. Los comportamientos de alto nivel exigen experiencia especializada, como los requisitos legales de las adquisiciones de empresas o la interpretación de espectrogramas de masas.

Figura 2.1 Un espectro de comportamiento inteligente, en el rango de comportamiento basado en reacción hasta Expertis especializado, adaptado de “Inteligencia Artificial” Un Enfoque Moderno por Russell, S. 2004

Una manera para verificar y comprobar una definición operacional y satisfactoria de inteligencia es la prueba propuesta por Alan Turing denominada “prueba de Turing”, la cual está diseñada de tal manera que evita el uso de la larga lista común de atributos necesarios para obtener inteligencia artificialmente, él sugirió una prueba basada en la incapacidad de diferenciar entre entidades inteligentes indiscutibles y seres humanos. La computadora supera la prueba si un evaluador humano no es

capaz de distinguir si las respuestas, a una serie de preguntas planteadas, son de una persona o no.

En la actualidad para que una computadora realice dicha prueba debe de contar con los siguientes atributos:

- **Procesamiento de lenguaje natural** que le permita comunicarse satisfactoriamente en inglés.
- **Representación del conocimiento** para almacenar lo que se conoce o siente.
- **Razonamiento automático** para utilizar la información almacenada para responderá preguntas y extraer nuevas conclusiones.
- **Aprendizaje automático** para adaptarse a nuevas circunstancias y para detectar y extrapolar patrones.

La Prueba de Turing evitó deliberadamente la interacción física directa entre el evaluador y el computador, dado que para medir la inteligencia es innecesario simular físicamente a una persona. Sin embargo, la llamada Prueba Global de Turing incluye una señal de vídeo que permite al evaluador valorar la capacidad de percepción del evaluado, y también le da la oportunidad al evaluador de pasar objetos físicos «a través de una ventanita». Para superar la Prueba Global de Turing el computador debe estar dotado de:

• Visión computacional para percibir objetos.

• Robótica para manipular y mover objetos.

Estas seis disciplinas abarcan la mayor parte de la IA, y Turing merece ser reconocido por diseñar una prueba que se conserva vigente después de 50 años.

A pesar de esto los investigadores han dedicado poco a esfuerzo a evaluar este método dado que están más interesados en que más importante el estudio de los principios en los que se basa la inteligencia que duplicar un ejemplar.

Para poder profundizar en el concepto de Inteligencia Artificial podemos establecer algunos de los fundamentos que permiten su desarrollo, entre ellos se encuentran:

- Filosofía (año 428 a.C. hasta el presente)
- Matemáticas (desde el año 800 al presente)
- Economía (1776 hasta el presente)
- Neurociencia (año 1861 hasta el presente)
- Psicología (año 1879 hasta el presente)
- Ingeniería computacional (año 1940 hasta el presente)
- Teoría de control y cibernética (año 1948 hasta el presente)
- Lingüística (año 1957 hasta el presente)

Para poder definir la IA se pueden establecer algunos sucesos históricos que marcan el desarrollo de la misma, dichos sucesos permiten establecer una cronología del desarrollo de esta ciencia. (Russell, 2004)

El primer trabajo de Inteligencia Artificial reconocido corresponde al periodo transcurrido entre 1944 y 1955, por medio de los autores McCulloch y Pitts partiendo de tres fuentes: conocimientos sobre la fisiología básica y funcionamiento de las neuronas en el cerebro, el análisis formal de la lógica proposicional de Russell y Whitehead y la teoría de la computación de Turing.

Hay un número de trabajos iniciales que se pueden caracterizar como de IA, pero fue Alan Turing quien articuló primero una visión de la IA en su artículo Computing Machinery and Intelligence, en 1950. Ahí, introdujo la prueba de Turing, el aprendizaje automático, los algoritmos genéricos y el aprendizaje por refuerzo.

El nacimiento formal del término Inteligencia Artificial data del año de 1956 cuando John McCarthy presento su taller en el Dartmouth College, el taller de Dartmouth no produjo ningún avance notable, pero puso en contacto a las figuras importantes de este campo. Durante los siguientes 20 años, el campo estuvo dominado por estos personajes, así como por sus estudiantes y colegas del MIT, CMU, Stanford e IBM. Quizá lo último que surgió del taller fue el consenso en adoptar el nuevo nombre propuesto por McCarthy para este campo: Inteligencia Artificial.

Durante los años siguientes se dieron varios procesos de evolución en el sector de IA hasta inicios de los 80 se convierte en una industria. El primer sistema experto comercial que tuvo éxito, R1, inició su actividad en Digital Equipment Corporation (McDermott, 1982). El programa se utilizaba en la elaboración de pedidos de nuevos sistemas informáticos. En 1986 representaba para la compañía un ahorro estimado

de 40 millones de dólares al año. En 1988, el grupo de Inteligencia Artificial de DEC había distribuido ya 40 sistemas expertos, y había más en camino. Du Pont utilizaba ya 100 y estaban en etapa de desarrollo 500 más, lo que le generaba ahorro de diez millones de dólares anuales aproximadamente. Casi todas las compañías importantes de Estados Unidos contaban con su propio grupo de IA, en el que se utilizaban o investigaban sistemas expertos.

A partir de 1986 las Redes Neuronales hacen el impulso más fuerte se produjo a mediados de la década de los 80, cuando por lo menos cuatro grupos distintos reinventaron el algoritmo de aprendizaje de retroalimentación, mencionado por vez primera en 1969 por Bryson y Ho. El algoritmo se aplicó a diversos problemas de aprendizaje en los campos de la informática y la psicología, y la gran difusión que conocieron los resultados obtenidos, publicados en la colección Parallel Distributed Processing (JL McClelland, 1986) suscitó gran entusiasmo. (Russell, 2004)

Aquellos modelos de Inteligencia Artificial llamados conexionistas fueron vistos por algunos como competidores tanto de los modelos simbólicos propuestos por Newell y Simon como de la aproximación lógica de McCarthy entre otros (Smolensky, 1988). Puede parecer obvio que los humanos manipulan símbolos hasta cierto nivel, de hecho, el libro The Symbolic Species (1997) de Terrence Deacon sugiere que esta es la característica que define a los humanos, pero los conexionistas más ardientes se preguntan si la manipulación de los símbolos desempeña algún papel justificable en determinados modelos de cognición. (Deacon, 1997)

Fue en el año de 1987 cuando la IA se ha descubierto muchos cambios en la metodología de trabajo de la Inteligencia Artificial. Actualmente es más usual el desarrollo sobre teorías ya existentes que proponer teorías totalmente novedosas.

En términos metodológicos, se puede decir, con rotundidad, que la IA ya forma parte del ámbito de los métodos científicos. Para que se acepten, las hipótesis se deben someter a rigurosos experimentos empíricos, y los resultados deben analizarse estadísticamente para identificar su relevancia (Cohen, 1995). El uso de Internet y el compartir repositorios de datos de prueba y código, ha hecho posible que ahora se puedan contrastar experimentos. Un buen modelo de la tendencia actual es el campo del reconocimiento del habla.

La utilización de metodologías mejoradas y marcos teóricos ha autorizado que este campo alcance un grado de conocimiento que ha permitido que ahora las redes neuronales se puedan comparar con otras técnicas similares de campos como la estadística, el reconocimiento de patrones y el aprendizaje automático, de forma que las técnicas más prometedoras pueden aplicarse a cualquier problema.

Como resultado de estos desarrollos, la tecnología denominada minería de datos ha generado una nueva y vigorosa industria. Revoluciones similares y suaves se han dado en robótica, visión por computador, y aprendizaje automático. La comprensión mejor de los problemas y de su complejidad, junto a un incremento en la sofisticación de las matemáticas ha facilitado el desarrollo de una agenda de investigación y de métodos más robustos. En cualquier caso, la formalización y

especialización ha llevado también a la fragmentación: áreas como la visión y la robótica están cada vez más aislados de la «rama central» de la IA. La concepción unificadora de IA como diseño de agentes racionales puede facilitar la unificación de estos campos diferentes.

Con esto Los avances recientes logrados en el entendimiento de las bases teóricas de la inteligencia han ido aparejados con las mejoras realizadas en la optimización de los sistemas reales. Los subcampos de la IA se han integrado más y la IA ha encontrado elementos comunes con otras disciplinas.

Los trabajos de Warren McCulloch y Walter Pitts (1943) han sido reconocidos como los autores del primer trabajo de IA. Partieron de tres fuentes: conocimientos sobre la fisiología básica y funcionamiento de las neuronas en el cerebro, el análisis formal de la lógica proposicional de Russell y Whitehead y la teoría de la computación de Turing.

Propusieron un modelo constituido por neuronas artificiales, en el que cada una de ellas se caracterizaba por estar «activada» o «desactivada»; la «activación» se daba como respuesta a la estimulación producida por una cantidad suficiente de neuronas vecinas. El estado de una neurona se veía como “equivalente, de hecho, a una proposición con unos estímulos adecuados”. Mostraron, por ejemplo, que cualquier función de cómputo podría calcularse mediante alguna red de neuronas interconectadas, y que todos los conectores lógicos (*and*, *or*, *not*, etc.) se podrían implementar utilizando estructuras de red sencillas. McCulloch y Pitts también

sugirieron que redes adecuadamente definidas podrían aprender. Donald Hebb (1949) propuso y demostró una sencilla regla de actualización para modificar las Intensidades de las conexiones entre neuronas. Su regla, ahora llamada de aprendizaje Hebbiano o de Hebb, sigue vigente en la actualidad.

Hay un número de trabajos iniciales que se pueden caracterizar como de IA, pero fue Alan Turing quien articuló primero una visión de la IA en su artículo *Computing Machinery and Intelligence*, en 1950. Ahí, introdujo la prueba de Turing, el aprendizaje automático, los algoritmos genéricos y el aprendizaje por refuerzo. (Russell, 2004)

Nacimiento de la Inteligencia Artificial (1956)

Princeton acogió a otras de las figuras señeras de la IA, John McCarthy. Posteriormente a su graduación, McCarthy se trasladó al Dartmouth College, que se erigiría en el lugar del nacimiento oficial de este campo. McCarthy convenció a Minsky, Claude Shannon y Nathaniel Rochester para que le ayudaran a aumentar el interés de los investigadores americanos en la teoría de autómatas, las redes neuronales y el estudio de la inteligencia.

Organizaron un taller con una duración de dos meses en Dartmouth en el verano de1956. Hubo diez asistentes en total, entre los que se incluían Trenchard More de Princeton, Arthur Samuel de IBM, y Ray Solomonoff y Oliver Selfridge del MIT.

Dos investigadores del Carnegie Tech, Allen Newell y Herbert Simon, acapararon la atención. Si bien los demás también tenían algunas ideas y, en algunos casos, programas para aplicaciones determinadas como el juego de damas, Newell y Simon contaban ya con un programa de razonamiento, el Teórico Lógico (TL), del que Simon afirmaba: “Hemos inventado un programa de computación capaz de pensar de manera no numérica, con lo que ha quedado resuelto el venerable problema de la dualidad mente-cuerpo”. Se dice que Russell se manifestó complacido cuando Simon le mostró que la demostración de un teorema que el programa había generado era más corta que la que aparecía en *Principia*. Los editores de la revista *Journal of Symbolic Logic* resultaron menos impresionados y rechazaron un artículo cuyos autores eran Newell, Simon y el Teórico Lógico (TL).

John McCarthy se trasladó de Dartmouth al MIT, donde realizó tres contribuciones cruciales en un año histórico: 1958. En el Laboratorio de IA del MIT Memo Número 1, McCarthy definió el lenguaje de alto nivel Lisp, que se convertiría en el lenguaje de programación dominante en la IA. Lisp es el segundo lenguaje de programación más antiguo que se utiliza en la actualidad, ya que apareció un año después de FORTRAN. Con Lisp, McCarthy tenía la herramienta que necesitaba, pero el acceso a los escasos y costosos recursos de cómputo aún era un problema serio. Para solucionarlo, él, junto a otros miembros del MIT, inventó el tiempo compartido.

También, en 1958, McCarthy publicó un artículo titulado Programs with Common Sense, en el que describía el Generador de Consejos, un programa hipotético que podría considerarse como el primer sistema de IA completo. Al igual que el Teórico

Lógico y el Demostrador de Teoremas de Geometría, McCarthy diseñó su programa para buscar la solución a problemas utilizando el conocimiento. (Russell, 2004)

Pero, a diferencia de los otros, manejaba el conocimiento general del mundo. Por ejemplo, mostró cómo algunos axiomas sencillos permitían a un programa generar un plan para conducirnos hasta el aeropuerto y tomar un avión. El programa se diseñó para que aceptase nuevos axiomas durante el curso normal de operación, permitiéndole así ser competente en áreas nuevas, sin necesidad de reprogramación. El Generador de Consejos incorporaba así los principios centrales de la representación del conocimiento y el razonamiento: es útil contar con una representación formal y explícita del mundo y de la forma en que la acción de un agente afecta al mundo, así como, ser capaces de manipular estas representaciones con procesos deductivos. Es sorprendente constatar cómo mucho de lo propuesto en el artículo escrito en 1958 permanece vigente incluso en la actualidad. 1958 fue el año en el que Marvin Minsky se trasladó al MIT. Sin embargo, su colaboración inicial no duró demasiado. McCarthy se centró en la representación y el razonamiento con lógica formal, mientras que Minsky estaba más interesado en lograr que los programas funcionaran y eventualmente desarrolló un punto de vista anti-lógico.

En 1963 McCarthy creó el Laboratorio de IA en Stanford. Su plan para construir la versión más reciente del Generador de Consejos con ayuda de la lógica sufrió un considerable impulso gracias al descubrimiento de J. A. Robinson del método de resolución. El trabajo realizado en Stanford hacía énfasis en los métodos de

propósito general para el razonamiento lógico. Algunas aplicaciones de la lógica incluían los sistemas de planificación y respuesta a preguntas de Cordell Green (1969), así como el proyecto de robótica de Shakey en el nuevo Instituto de Investigación de Stanford (Stanford Research Institute, SRI).

Años más tarde, el programa DENDRAL (Buchanan *et al.*, 1969) utilizó los procedimientos denominados métodos débiles, los cuales consisten en entrelazar elementos de razonamiento básicos para encontrar así soluciones completas. Fue diseñado en Stanford, donde Ed Feigenbaum (discípulo de Herbert Simon), Bruce Buchanan (filósofo convertido en informático) y Joshua Lederberg (genetista ganador del Premio Nobel) colaboraron en la solución del problema de inferir una estructura molecular a partir de la información proporcionada por un espectrómetro de masas.

La versión más simple del programa generaba todas las posibles estructuras que correspondieran a la fórmula, luego predecía el espectro de masas que se observaría en cada caso, y comparaba éstos con el espectro real. La trascendencia de DENDRAL se debió a ser el primer sistema de conocimiento intenso que tuvo éxito: su base de conocimiento estaba formada por grandes cantidades de reglas de propósito particular.

Feigenbaum junto con otros investigadores de Stanford dieron comienzo al Proyecto de Programación Heurística, PPH, dedicado a determinar el grado con el que la nueva metodología de los sistemas expertos podía aplicarse a otras áreas de la actividad humana.

El primer sistema experto comercial que tuvo éxito, R1, inició su actividad en Digital Equipment Corporation (McDermott, 1982). El programa se utilizaba en la elaboración de pedidos de nuevos sistemas informáticos. En 1986 representaba para la compañía un ahorro estimado de 40 millones de dólares al año.
La IA se fundó en parte en el marco de una rebelión en contra de las limitaciones de los campos existentes como la teoría de control o la estadística, y ahora abarca estos campos. (Russell, 2004)

Tal y como indica David McAllester (1998), En los primeros años de la IA parecía perfectamente posible que las nuevas formas de la computación simbólica, por ejemplo, los marcos y las redes semánticas, hicieran que la mayor parte de la teoría clásica pasara a ser obsoleta. Esto llevó a la IA a una especie de aislamiento, que la separó del resto de las ciencias de la computación. En la actualidad se está abandonando este aislamiento. Existe la creencia de que el aprendizaje automático no se debe separar de la teoría de la información, que el razonamiento incierto no se debe separar de los modelos estocásticos, de que la búsqueda no se debe aislar de la optimización clásica y el control, y de que el razonamiento automático no se debe separar de los métodos formales y del análisis estático.

En términos metodológicos, se puede decir, con rotundidad, que la IA ya forma parte del ámbito de los métodos científicos. Para que se acepten, las hipótesis se deben someter a rigurosos experimentos empíricos, y los resultados deben analizarse estadísticamente para identificar su relevancia (Cohen, 1995). El uso de Internet y

el compartir repositorios de datos de prueba y código, ha hecho posible que ahora se puedan contrastar experimentos.

La Inteligencia Artificial ha sido tema de estudio en los últimos años, diversos autores han expuesto su definición, a continuación, se describe algunas de estas definiciones, divididas por el tipo de comportamiento que el autor analizó, algunas de ellas enfocadas en los procesos mentales y el razonamiento, así como algunas de ellas a la conducta. Otro criterio seleccionado es el grado en que la Inteligencia Artificial mide el éxito en términos de fidelidad en base a la forma de actuar como el humano, mientras otras hacen referencia al concepto ideal de inteligencia llamado racionalidad. (Racional: si el sistema hace lo correcto, en función de su conocimiento) (Figura 2.2):

Sistemas que piensan como humanos	**Sistemas que piensan racionalmente**
«El nuevo y excitante esfuerzo de hacer que los computadores piensen… máquinas con mentes, en el más amplio sentido literal». (Haugeland, 1985) «[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje…» (Bellman, 1978)	«El estudio de las facultades mentales median- te el uso de modelos computacionales». (Charniak y McDermott, 1985) «El estudio de los cálculos que hacen posible percibir, razonar y actuar». (Winston, 1992)

Sistemas que actúan como humanos	Sistemas que actúan racionalmente
«El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realiza- das por personas requieren de inteligencia». (Kurzweil, 1990) «El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor». (Rich y Knight, 1991)	«La Inteligencia Computacional es el estudio del diseño de agentes inteligentes». (Poole *et al.*, 1998) «IA… está relacionada con conductas inteligentes en artefactos». (Nilsson, 1998)

Figura 2.2 Algunas definiciones de Inteligencia Artificial, organizadas en cuatro categorías adaptado de "Inteligencia Artificial" Un Enfoque Moderno por Russell, S. 2004

Otros autores como Adrian Hopgood de la Universidad de Nottingham Trent, mencionan que la mayoría de las definiciones son demasiado complejas para un estudio general del campo, por lo que aquí hay una simple que será suficiente: la Inteligencia Artificial es la ciencia de imitar las facultades mentales humanas en una computadora.

2.1.2 Paradigmas de la IA

Kasabov define la Inteligencia Artificial comprende métodos, herramientas, y sistemas para resolver problemas que normalmente requieren la inteligencia humana. El termino Inteligencia siempre está definido por la capacidad de aprender efectivamente, de reaccionar adaptativamente, de tomar decisiones propias, de comunicar en lenguaje o imágenes en una manera sofisticada y el entender. Los principales objetivos de la Inteligencia Artificial son desarrollar métodos y sistemas para resolver problemas, usualmente resueltos por el intelecto de la actividad humana, por ejemplo, el reconocimiento de imágenes, lenguaje y procesamiento de

voz, planeación, predicción, mejorando así los sistemas de cómputo de información; y desarrollar modelos que simulan organismos vivos y el cerebro humano en particular, mejorando así nuestro entendimiento de cómo trabaja el cerebro humano.

La principal dirección de desarrollo de la Inteligencia Artificial es desarrollar métodos y sistemas para resolver problemas de IA sin la necesidad de seguir el camino que usan los humanos para hacerlo, pero proveyendo resultados similares, por ejemplo, los sistemas expertos; y desarrollando métodos y sistemas para resolver problemas de IA modelando la manera humana de pensar o la manera en que trabaja el cerebro humano físicamente, por ejemplo, las redes neuronales artificiales.

En general, la IA es modelar la inteligencia humana, hay dos paradigmas adoptados para lograr esto: El simbólico y el subsimbólico. El primero está basado en la manipulación y el segundo en el neurocomputo.

El paradigma simbólico está basado en la teoría de los sistemas simbólicos físicos de Newel y Simon en 1972. Un sistema simbólico consiste en dos conjuntos:

Un set de elementos (o símbolos) los cuales pueden ser usados para construir elementos más complicados o estructuras; y un conjunto de procesos y reglas, las cuales cuando son aplicadas a símbolos y estructuras, producen nuevas estructuras.

Los símbolos tienen significado semántico. Ellos representan conceptos u objetos. La lógica proposicional, la lógica de predicados y la producción de sistemas facilitan el tratar con sistemas simbólicos.

Algunos de sus correspondientes implementaciones de Inteligencia Artificial son los sistemas basados en reglas, los lenguajes de programación lógica y de producción. Los Sistemas simbólicos de IA han sido aplicados al procesamiento de lenguaje natural, sistemas expertos, aprendizaje automático, modelos de procesos cognitivos y otros.

Desafortunadamente, ellos no han tenido éxito en todos los casos cuando son inexactos, faltantes o inciertos con la información usada, cuando solo se dispone de datos brutos y se debe realizar la adquisición de conocimientos, o cuando es necesario elaborar soluciones paralelas. Estas tareas no resultan difíciles para los humanos.

El paradigma subsimbólico de Smolenski desarrollado en los 90s afirma que el comportamiento inteligente es realizado en un nivel subsimbólico que es más grande que el nivel neuronal en el cerebro, pero es diferente al simbólico.

El procesamiento del conocimiento es el cambio de estados de redes construidas de pequeños elementos llamados neuronas, replicando la analogía de las neuronas reales. Una neurona, o una colección de neuronas pueden representar micro representaciones de un concepto o un objeto. Se ha demostrado que es posible diseñar un sistema inteligente que alcance un comportamiento de proporción global aun cuando todos los componentes del sistema son simples y operan solo con información local pura.

El paradigma subsimbólico hace posible no solo el uso de todos estos resultados significativos en el área de las redes neuronales artificiales, si no ha logrado en alrededor de 20 años áreas como el reconocimiento de patrones e imagen, procesamiento de voz y además hizo posible el uso de modelos de procesamiento de aprendizaje con modelos conexionistas.

Conforme los modelos subsimbólico se mueven más cerca, aunque lentamente, al cerebro humano, se cree que es el camino correcto para entender y modelar la inteligencia humana para la ingeniería del conocimiento.

Hay varias maneras en las cuales los modelos simbólicos y subsimbólicos de procesamiento de conocimiento pueden interactuar:

- Ellos pueden ser desarrollados y usados de manera separada y alternativa.
- Los Sistemas híbridos que incorporan ambos sistemas tanto los simbólicos como los subsimbólicos pueden ser desarrollados,
- Los sistemas subsimbólicos pueden ser usados para modelar sistemas simbólicos puros.

Por lo tanto, hay un tercer paradigma una mezcla de sistemas simbólicos y subsimbólicos. Veremos que esos sistemas difusos pueden representar conocimiento simbólico, pero ellos también usan representaciones similares a las usadas en los sistemas subsimbólicos. Al momento podemos observar que la unión

de métodos simbólicos y subsimbólicos en muchos de los casos puede ser la mejor solución en problemas complejos de Inteligencia Artificial. (Russell, 2004)

2.2 Tipos de Inteligencia Artificial

Redes neuronales Artificiales

Como menciona Shihab en su artículo titulado: “A Backpropagation Neural Network for Computer Network Security”, el problema de proteger la información ha existido desde que se ha administrado esta misma. Sin embargo, los avances tecnológicos y los sistemas de administración de información se han vuelto más y más poderosos, el problema de transmitir información segura también se ha vuelto más crítico. El uso masivo de redes de comunicación para diversos propósitos en los últimos años ha encontrado nuevas serias amenazas de seguridad y ha incrementado el potencial daño que estas violaciones pueden causar. A medida que las organizaciones incrementan su dependencia en ambientes de redes de computadoras, ellas se vuelven más vulnerables a las infracciones de seguridad. Los sectores público y privado más que nunca dependen de la información que ellos manejan. Una violación de seguridad de la información puede poner en peligro todo el sistema funcional y causar serios daños. El avance en Redes Neuronales Artificiales provee soluciones efectivas para este problema. (Shihab, 2006)

Deboeck y Kohonen describen las redes neuronales (RN) como una colección de técnicas matemáticas que pueden ser usadas para el procesamiento de señales, previsión y agrupamiento y definido como técnicas de regresión paralela, no linear

y multicapa. Se afirma además que el modelado de redes neuronales es como ajustar una línea, plano o hiperplano a través de un conjunto de puntos de datos. Una línea, plano o hiperplano puede ser ajustado a través de cualquier data set para definir la relación que puede existir entre las entradas y las salidas; o puede ser ajustado para identificar una representación de datos en una escala menor.

La primera definición describe la Red Neuronal Artificial desde la similitud de la actividad humana como se muestra en la siguiente Figura 1.2 y la segunda (definición de Kohonen) en una perspectiva de aplicación.

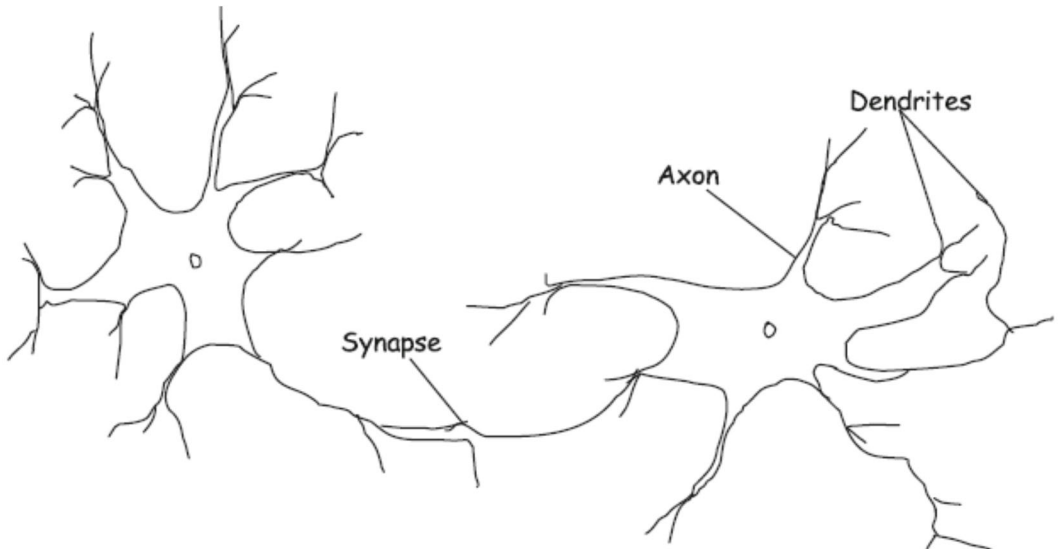


Figura 2.3 Representación de Neurona Biológica adaptado de “Artificial Neural Network Modelling” por Shanmuganathan, Subana, Samarasinghe, Sandhya, Copyright 2016 por Springer.

Es verdaderamente aceptado incluyendo iniciativas recientes de investigación del cerebro que el cerebro humano es mucho más complicado ya que muchas de sus funciones cognitivas son todavía desconocidas. Sin embargo, las siguientes son las principales características consideradas y descritas como funciones comunes en redes reales y artificiales:

1.- Aprendizaje y Adaptación

2.- Generalización

3.- Paralelismo masivo

4.- Robustez

5.- Almacenamiento asociativo de información

6. Procesamiento espacio- temporal de información.

En 1943 McCulloch y Pitts fueron los primeros en introducir un modelo matemático de una neurona. Ellos continuaron su trabajo y exploraron los paradigmas de reconocimiento de patrones a pesar de los problemas relacionados con el ángulo de rotación, la traslación y el factor de escala. Muchos de sus trabajos utilizaron un modelo de neurona simple y a estas redes generalmente se les conoce como perceptrones.

Rosenblatt en 1958 usando el modelo de Pitts y McCulloch realizo una red con la finalidad de modelar los fenómenos de percepción visual. En 1962, Rosenblatt probó un teorema de aprendizaje en los perceptrones. El mostro que un perceptrón puede aprender cualquier cosa que pueda ser representada. Widrow en colaboración con Angell y con Hoff demostró modelos convincentes. Para entonces el mundo estaba explorando el potencial de los perceptrones, pero seguían representando un sistema de capas con tareas simples y con ciertas fallas. Minsky probó más adelante que los perceptrones de capa simple tienen ciertas restricciones en su habilidad de representar y aprender. Además, dudaba en encontrar un algoritmo de aprendizaje para redes neuronales multicapa. Esto hizo que la

investigación diera un giro a desarrollar otro tipo de Inteligencia Artificial Simbólica y Sistemas como por ejemplo los Sistemas Expertos.

De 1977 en adelante surgieron nuevos modelos, como los de Memoria Asociativa, perceptrones multicapa y algoritmos de aprendizaje de Propagación hacia atrás; Teorías de Resonancias Adaptativas y redes auto- organizadas.

Estos nuevos modelos conexionistas despertaron el interés de muchos investigadores en los Sistemas Sub simbólicos y como resultado muchas redes fueron diseñadas y usadas desde entonces como las Redes de memoria introducida asociativa bidireccional, funciones de base radial, Redes Neuronales Probabilísticas RAM, Neuronas Fuzzy y Redes Fuzzy presentadas y muchas más. Basado en los diferentes Modelos de neuronas y aplicaciones de Red se han desarrollado y así mismo usado exitosamente una gran cantidad de aplicaciones en diversas disciplinas. (Shanmuganathan, 2016)

2.3 Seguridad Informática

El aspecto de la seguridad al transmitir información ha sido de vital importancia a través de la historia, ya que presenta registros de las grandes civilizaciones anteriores a la época moderna, tales como instrumentos de encriptación encontrados en la antigua Mesopotamia, los hebreos, griegos y los primeros libros de criptografía en el año 855 D.C nos revelan la importancia que representan para las civilizaciones la integridad de la información.

En la época moderna a partir de 1980, con el artículo de James Anderson, Computer Security Threat Monitoring and Surveillance, comienza una nueva época en la cual la detección de intrusos marca su inicio. (Anderson, 1980)

Este trabajo fue escrito para una organización gubernamental y presentó la moción de que las pistas de auditoría contenían información vital que podría ser valiosa para rastrear el mal uso y la comprensión del comportamiento de los usuarios. Con el lanzamiento de este artículo surgió el concepto de "detectar" mal uso y eventos específicos del usuario. Su conocimiento de los datos de auditoría y su importancia llevaron a mejoras tremendas en los subsistemas de auditoría de prácticamente todos los sistemas operativos. La hipótesis de Anderson también proporcionó la base para el diseño futuro y desarrollo del sistema de detección de intrusos. Su trabajo fue el comienzo de la detección de intrusiones basadas en host e IDS en general.

En 1983, el Dr. Dorothy Denning, comenzó a trabajar en un proyecto gubernamental que lanzó un nuevo esfuerzo en la detección de intrusos. Su objetivo era analizar pistas de auditoría de computadoras centrales del gobierno y crear perfiles de usuarios basados en sus actividades. Un año después, El Dr. Denning ayudó a desarrollar el primer modelo para la detección de intrusos, el Sistema experto en detección de intrusiones (IDES), que proporcionó el fundamento para la tecnología IDS desarrollo que pronto seguiría.

En 1984, SRI (Stanford Research Institute) también desarrolló un medio de seguimiento y análisis de datos de auditoría que contiene información de autenticación de usuarios en ARPANET, el Internet original. Poco después, SRI completó un Contrato de la Marina SPAWAR con la realización del primer sistema de detección de intrusos funcional, IDES.

El desarrollo comercial de tecnologías de detección intrusiones comenzó en los principios de los noventa. Haystack Labs fue el primer vendedor comercial de herramientas IDS, con su línea Stalker de host productos. Desde entonces las compañías le han dado una verdadera importancia a la detección de intrusos dentro de sus soluciones de servicios. (Asmaa Shaker Ashoor, 2011)

2.3.1 Estrategias de seguridad informática

Introducción al funcionamiento de sistemas de detección de intrusos (IDS)

La detección de intrusos, un componente importante de la tecnología de seguridad de la información, ayuda a descubrir, determinar e identificar el uso no autorizado, la duplicación, la alteración y la destrucción de la información y los sistemas de información. La detección de intrusiones se basa en el supuesto de que la información y los sistemas de información bajo ataque exhiben varios patrones o características de comportamiento distinguibles a los de los normales. Aunque la tecnología de detección de intrusos se está volviendo omnipresente en la defensa de red actual; carece de definiciones básicas y comprensión matemática. La detección de intrusiones es subjetiva; cada sistema de detección de intrusiones (IDS) tiene una clasificación diferente y mecanismos de etiquetado de ataque. Es

más común que los IDS alarmen cualquier conjunto de comportamientos de ataque conocidos. En el curso debido de determinar si una actividad particular es normal o maliciosa, los IDS no pueden alarmar un ataque (falso negativo) o la actividad normal de alarma como maliciosa (falso positivo). La forma más popular de detectar intrusiones se ha realizado mediante el uso de datos de auditoría generados por sistemas operativos y redes. Dado que casi todas las actividades se registran en un sistema, es posible que una inspección manual de estos registros permita detectar intrusiones. Es importante analizar los datos de auditoría incluso después de que haya ocurrido un ataque, para determinar la extensión del daño ocurrido, este análisis ayuda a rastrear el ataque y también ayuda a registrar los patrones de ataque para la prevención futura de tales ataques. Se puede usar un sistema de detección de intrusiones para analizar los datos de auditoría para tales percepciones. Esto hace que el sistema de detección de intrusos sea una valiosa herramienta de detección y prevención en tiempo real, así como una herramienta de análisis forense. (Asmaa Shaker Ashoor, 2011)

2.4 Comentarios finales

La inevitable evolución de la tecnología, nos permite ver las diversas investigaciones que hacen que la seguridad de la información sea un tema estudiado de manera constante, la diversificación de tipos de ataque nos hacen recurrir a herramientas que cada vez son menos posibles de identificar con los recursos humanos disponibles, por lo que se tiene que mezclar de una manera especial con el uso de las tecnologías de inteligencia Artificial, lo cual despierta el interés de la presente investigación.

CAPITULO 3 Experimentación

3.1 Modelo Propuesto y Experimentación

El modelo propuesto en la presente investigación está compuesto por una red neuronal que será entrenada por ataques de una colección de datos como motivo de experimentación a nivel mundial, esta colección de datos (data set) será clasificado mediante los criterios del modelo denominado STRIDE de Microsoft, el cual divide las amenazas de acuerdo a su comportamiento de la siguiente manera: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). Lo anterior con la finalidad de establecer los parámetros para que nuestra red neuronal, establezca un patrón de comportamiento y pueda darnos la mayor cantidad de casos detectados de esta data set, el entrenamiento se llevará a cabo utilizando el siguiente modelo:

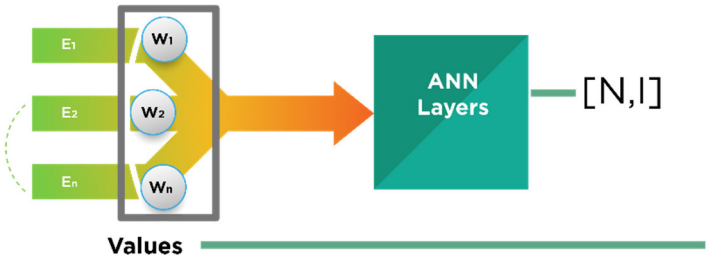


Figura 3.1 Representación de Modelo Porpuesto elaborada por S.A. Ordoñez, 2021

Para esto se establecen los criterios de análisis para detección de Intrusión a diferentes sistemas, obteniendo indicadores en función al tipo de escenario en el cual se presentan comúnmente. La clasificación se realiza de la siguiente manera:

Escenarios Comunes

1. Sistema de Detección de Intrusos (IDS).

2. – Probabilidad de Ataque.

3. – Detección de comportamientos anormales.

4. – Filtrado y Aislamiento de Paquetes

3.1.1 Elementos de Análisis:

Los elementos de Análisis se seleccionaron con especial cuidado, buscando que los registros nos den la información más precisa, estableciendo parámetros de comportamiento más vulnerables y relevantes dentro de las redes y sistemas de cómputo, los cuales se definen en:

1. – Registro de Eventos (logs).

2. – Encabezados TCP.

3. –Tráfico de Datos.

4. – Análisis de Puertos.

Evaluación del IDS

$$OA = \frac{\text{Numero de Ataques clasificados correctamente}}{\text{Numero total de Registros}} = \frac{VP+VN}{TP+TN+FP+FN}$$

OA: Índice de calificación de Eficiencia del IDS

VP: Verdadero positivo, Clasificación correcta de registros de intrusión como intrusión.

VN: Verdadero negativo, Clasificación correcta de registros normales como normales.

FP: Falso positivo, clasificación incorrecta de registros como intrusión y realmente normal.

FN: Falso negativo, clasificación incorrecta de registros como normal, y realmente son una intrusión.

Tabla 3.1

Evaluación del IDS

	N	I
N	TN	FN
I	FP	TP

N: Normal I: Intrusión

Nota: Representación de las Evaluaciones IDS
Tabla Elaborada por S.A. Ordoñez, 2021

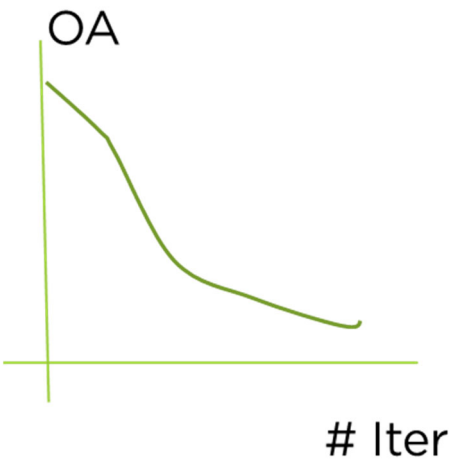


Fig. 3.1. Presentación del Entorno propuesto, El principal objetivo del modelo es minimizar el Overall Average de incidentes, lo más cercano a 0.Elaborada por S.A. Ordoñez, 2021

3.2 Especificaciones de la Plataforma.

La plataforma propuesta se basa en una Red Neuronal Multicapa. La red neuronal que se utilizará por su robustez y buen funcionamiento es una red de propagación hacia atrás, que se ha utilizado en otros esquemas de seguridad informática; Esta red neuronal contendrá las entradas de la plataforma, que consiste en un conjunto de características o incidentes aislados, en el sentido de que no necesariamente tienen que estar presentes para representar un tipo de ataque.

Lo anterior se puede ver en la siguiente Tabla 3.2:

Tabla 3.2

Representación de propuesta basada en una Red Neuronal Multicapa

	Car1	Car2	Car3	Car4	TP	TN
Attack1	1	0	1	1	1	0
Attack2	1	1	0	0	1	0
No Attack1	0	0	0	0	0	1
No Attack2	1	0	0	0	0	1

Tabla 3.2 Modelo Propuesto con base a Red Neuronal Multicapa, Elaborada por S.A. Ordoñez, 2021

Debido a que está destinado a maximizar los valores de las funciones A (precisión) y RD (detección de frecuencia), así como a minimizar la FAR (frecuencia de falsa alarma), se propone la red neuronal que utilizará estas funciones como funciones de evaluación para ajustar los pesos de la red para cumplir con el objetivo de maximizar y minimizar las funciones previamente establecidas.

3.2.1 Análisis de Requerimientos

Al comenzar con el procedimiento de elaboración de la red neuronal se deben tomar en cuenta los siguientes puntos:

- Generar un sistema de seguridad basado en Inteligencia Artificial, específicamente haciendo uso de las redes neuronales.
- Se debe usar un conjunto de datos que cubra con los requisitos necesarios para el entrenamiento, así como identificar los valores más adecuados para resolver este tipo de problemas.
- El método de entrenamiento de la red debe de ser óptimo, confiable y efectivo para detectar posibles ataques a los equipos informáticos, para así asegurarse que los resultados obtenidos son los correctos y así posteriormente al realizar cálculos masivos se tenga una mayor certeza de que se tiene todo correcto.
- Se realiza una serie de pruebas para identificar posibles conflictos en el funcionamiento.
- Se comprobarán los resultados asegurándonos que sean los más cercanos a los resultados de prueba.

3.3 Materiales

Se busca atacar el problema utilizando la Inteligencia Artificial inmersa en una red neuronal. Importante para una red neuronal es tener una fuente de datos para alimentar su “conocimiento”, es decir para entrenarla; tanto esto, las herramientas utilizadas y demás, serán analizadas a continuación.

La base del conocimiento para la red se tomó de la Copa KDD de 1999. De aquí hay que explicar dos cosas: KDD y la Copa per se. El descubrimiento de conocimiento en bases de datos (KDD por sus siglas en inglés) se define como la extracción no trivial de información implícita, previamente desconocida y potencialmente útil de un conjunto de datos (Frawley, Matheus, & Piatetsky-Shapiro, 1992). Existe un organismo llamado Association for Computing Machinery el cual tiene un grupo de interés especializado en KDD. El SIGKDD[1] organiza anualmente una competencia para la minería de datos y descubrimiento de conocimiento siempre con un tema en específico. En su tercera edición se centró en la detección de intrusión en redes computacionales. Los datos se encuentran disponibles al público en la red y divide los ataques en 22 tipos y uno “normal”.

Este conjunto de datos, aunque basto y suficiente, muestra algunas deficiencias. Alrededor del 78% y el 75% de los registros están duplicados en los conjuntos de entrenamiento y de pruebas, respectivamente (Bagheri, Ghorbani, Lu, & Tavallaee, 2009). Investigadores de la Universidad de Nueva Brunswick detallaron su análisis de los datos en “A Detailed Analysis of the KDD CUP 99 Data Set” en el 2009 y pusieron a disposición el set de datos optimizado por ellos. Este conjunto de datos, llamado NSL-KDD[2], es el que se usó como fuente de datos.

Como es obvio, lo que se busca en los estudios experimentales, es obtener los mejores resultados posibles. Para esto es necesario contar con una cantidad adecuada de datos a trabajar. Sin embargo, no siempre es posible tratar con toda

[1] https://www.kdd.org/about
[2] https://www.unb.ca/cic/datasets/nsl.html

la cantidad de datos que pertenecen a una población, esto debido a la cantidad de datos totales. Es por esta razón que es necesario seleccionar una parte de dicha población, y a esta parte se le conoce como muestreo. Este muestreo consiste en un conjunto de reglas, procedimientos y criterios mediante los cuales se selecciona un conjunto de elementos de una población que representan lo que sucede en toda esa población (Mata & Macassi, 1997).

Para propósitos de este trabajo la población total de la muestra es el NSL KDD Dataset. Aun así, la cantidad de datos de este conjunto es superior a lo requerido para la implementación de la red neuronal. Es por esto que se seleccionó una muestra de esta población, que la represente y, desde luego, se pretende que esta muestra sea un fiel reflejo de la población.

La base de datos con la población total de registros, cuenta con un número superior a 125,000 registros totales de paquetes informáticos de diverso tipo. En este caso la muestra que se tomó en cuenta para la implementación de la red será de 1500 registros. De estos, 1050 registros serán utilizados para el entrenamiento de la red neuronal; mientras que, por otro lado, los 450 registros restantes, serán aplicados a la red entrenada, para realización de pruebas y testeo del funcionamiento óptimo de la red neuronal. Es de notar, que la selección de la cantidad de datos fue una decisión empírica, pues no existe un método cien por ciento fiable de que se seleccionarán los registros mínimos necesarios para una implementación correcta y efectiva de la red.

Aun así, el método realizado para la selección de la muestra de la población, no fue al azar. En el método para la selección de los datos de muestra, se buscaba que los

registros fueran obtenidos de tal forma que se mantenga una proporción de los datos, en comparación con los originales. Es así, que se utilizó un método conocido como muestreo aleatorio estratificado. Este tipo de muestreo divide la población en grupos, en función de un carácter determinado y después se muestrea cada grupo aleatoriamente, para obtener una parte adecuada de cada subgrupo (Casal & Mateu, 2003). Este método se aplicó para evitar que por azar algún grupo de la muestra este menos representado que los otros, o que no sea incluido, ya que como ya se mencionó, el NSL KDD cuenta con registros de 23 tipos de paquetes informáticos o ataques informáticos, y es necesario que nuestra muestra contenga esos paquetes, y que, si se hubiese utilizado el azar, tal vez dichos registros no se hubiesen incluido, y la fiabilidad de la red no sería tan óptima.

La selección conllevo a seleccionar 1500 registros, con un aproximado proporcional al NSL KDD original, subdividiendo el NSL KDD original en los 23 tipos de paquetes informáticos del registro. Teniendo estos grupos, se procedió a seleccionar al azar algunos registros de cada subgrupo, para asegurar que los 23 tipos de registros se incluyesen en la muestra. La muestra obtenida se puede ver en los Anexos.

Teniendo las dos muestras, tanto para el entrenamiento, como para las pruebas que realizaremos con la red, procedemos a crear 4 archivos.

Tabla 3.3

Archivos derivados de los datos.

Archivo	Descripción
trainX	Contiene las entradas que agregaremos a la red neuronal para que sea entrenada
trainY	Contiene las salidas que agregaremos a la red neuronal para que sea entrenada
testX	Contiene las entradas que agregaremos a la red neuronal para que sea probada con estos registros
testY	Contiene las salidas que agregaremos a la red neuronal para que sea probada con estos registros

Tabla 3.3 Descripción de archivos derivados de los datos de análisis,
Elaborada por S.A. Ordoñez, 2021

Con los datos obtenidos, se tiene que garantizar que los datos que contiene el archivo son válidos y completos. Esto se consigue al realizar pruebas tales como contar los registros de la muestra, totalizar los campos que contienen dichos registros y verificando los datos, con el fin de asegurar que los archivos contengan el número correcto de registros, que los totales numéricos correspondan a los totales proporcionados en el archivo que contiene la población total de registros (NSL KDD Dataset) y que los campos contengan sólo datos válidos. Así como verificar que no haya registros duplicados en las muestras.

Ahora bien, para el desarrollo de la red se optó por utilizar el lenguaje de programación Python[3] y la distribución Anaconda[4], se eligieron porque muestran muchas bondades para la ciencia de los datos. Ambos son de código abierto y pueden ser descargados desde su sitio oficial.

3.4 Definición de Tareas

Dados los requerimientos anteriormente detallados se busca hacer una planeación, al ser un proyecto con muchos detalles importantes con un plan bien estructurado se evitaría pasar por alto algún de estos. Lo primero que se hizo fue dividir el proyecto en fases: de Análisis de Requerimientos, de Investigación, del Conjunto de Datos, de Desarrollo, de Pruebas y de Documentación. A continuación, se detallan las tareas realizadas en cada una de ellas.

- Análisis de Requerimientos: se analizó el problema a resolver, así como los objetivos planteados para poder definir las fases posteriores a esta, las herramientas y la metodología a emplear.
- Investigación: se dio a la tarea de buscar antecedentes literarios en los cuales apoyar el presente trabajo.
- Conjunto de Datos: en esta fase se definieron los criterios y metodología de selección de la muestra ya antes detallada, se llevó a cabo dicha selección, se

[3] https://www.python.org/
[4] https://www.anaconda.com/distribution/

normalizaron los datos y se prepararon los archivos para ser empleados por el sistema.

- Desarrollo: se procede a crear la red neuronal y los módulos de entrenamiento y datos de prueba necesarios para revisar su correcto desempeño.
- Pruebas: se determinaron y ejecutaron los casos de prueba para corregir lo necesario.
- Documentación: en cada fase se fueron haciendo anotaciones, pero se decidió establecer una fase enteramente dedicada a la documentación para revisar todo lo realizado.

3.5 Diseño del Sistema

El funcionamiento de la red neuronal se divide en 7 etapas, siendo en descripción general las siguientes:

1. Generar los pesos aleatorios para el entrenamiento de la red.
    a. Se generan los pesos de acuerdo con una función de aleatoriedad.
    b. Se generan la cantidad adecuada de pesos en función de las entradas y salidas.
2. Cargar los datos de entrenamiento
    a. Los datos de entrenamiento se cargan a través de archivos guardados en la misma carpeta en la que se encuentra el código principal del entrenamiento.
3. Realizar el entrenamiento

a. Se ejecuta el entrenamiento de acuerdo con el algoritmo generado en el código de entrenamiento.
b. Se usan los datos de pesos aleatorios.
c. Se genera pesos entrenados que servirán de guía para el resto de las ejecuciones.

4. Guardar los resultados del entrenamiento
    a. Los resultados del entrenamiento incluyen pesos entrenados que usara el algoritmo para procesar entradas.
    b. Los pesos se almacenan en archivos dentro de en la misma carpeta en la que se encuentra el código principal del entrenamiento.
5. Recibir los datos de entrada del posible ataque a analizar
    a. Los datos procesan distintas variables de comportamiento para identificar el ataque.
    b. Se reciben los archivos en un formato de comas.
6. Identificar si se trata o no de un posible ataque, de serlo, identificar de cual tipo de ataque se trata.
7. Generar y guardar resultados.
    a. Existen 23 posibles resultados mismos que son guardados dentro de una variable del sistema.

Para optimizar el funcionamiento de la red neuronal se agregará un paso al funcionamiento en el cual se valida si ya existe un entrenamiento previo, esto puede considerarse como una etapa más, en caso de que este entrenamiento previo exista

se saltaran las etapas de la 1 a la 4 para de esta manera agilizar la ejecución total del programa.

El funcionamiento puede ser representado en términos generales por medio del siguiente diagrama de flujo:

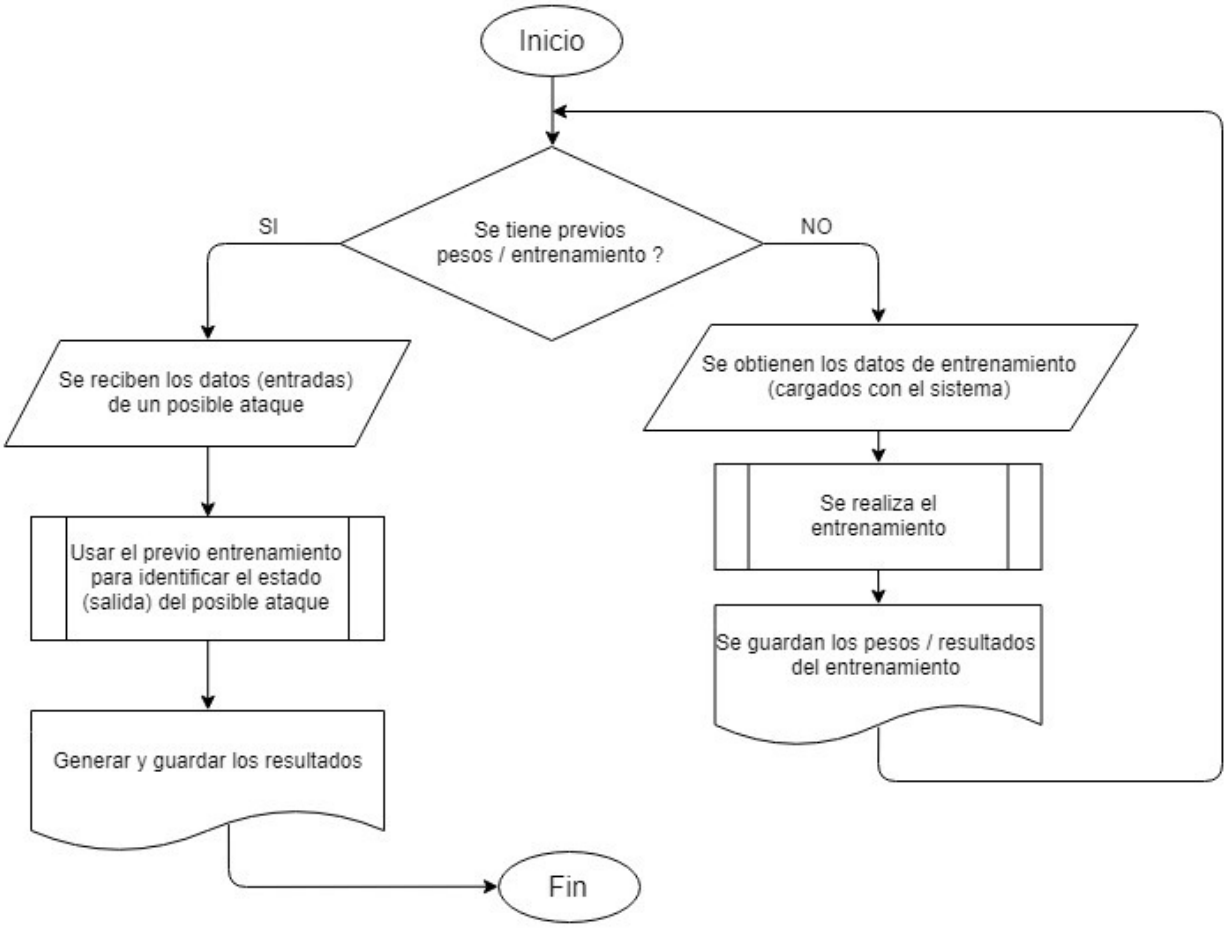


Figura 3.2 Flujo del sistema, Elaborada por S.A. Ordoñez, 2021

3.6 Detalle del Código

Pasando a la parte de la codificación, se optó por separar la parte de la red neuronal del flujo principal, esto solo para tener una mejor organización y claridad de este. Python ofrece muchas bondades, una de ellas es la capacidad de utilizar módulos con solo cargar los paquetes que los contienen. Se utilizaron tres paquetes numpy, para las operaciones matemáticas, scipy, para la optimización de la red e io para la creación y manipulación de los ficheros, dado que necesitábamos persistencia de datos.

3.7 Red Neuronal

Esta parte contempla dos clases: la de la red neuronal y la del entrenamiento. En la de red neuronal se contemplaron los siguientes métodos:

- *__init__:* método especial que inicializa los atributos del objeto aquí se detallaron las variables para las neuronas por capa.
- *randWeight:* se crean matrices con pesos aleatorios para el funcionamiento de la red. Aquí mismo hacemos uso del paquete *io* para almacenar los pesos en un archivo de texto.
- *dfltWeight:* define las variables de los pesos en base a pesos previamente definidos y almacenados en archivos.
- *sigmoid:* define la función de activación (sigmoide).
- *forward:* método que utiliza la red para “propagar” las entradas hacia la salida usando los pesos guardados en las variables W1 y W2 y la función de activación.}

- *costFunction:* está función calcula el error cuadrado medio del resultado obtenido comparando con el resultado esperado.
- *sigmoidPrime:* derivada de la función sigmoidal que se utilizará para la retropropagación.
- *costFunctionPrime:* ecuaciones resultantes de retro propagar la red.
- *getParams:* prepara los pesos para su manipulación haciéndolos vectores.
- *setParams:* devuelve los pesos a su estado “natural” (matrices) y los guarda en archivos (distintos a los del método *__init__*.
- *computeGradients:* guarda los gradientes de la retropropagación en dos variables.

En cuanto a la clase entrenadora:

- *__init__:* Crea una referencia local a la red.
- *callbackF:* función que se realiza después de cada iteración del optimizador.
- *costFunctionWraper:* función de apoyo para el método *train.*
- *train:* recupera las variables de entrada y salidas para poder usarlas a continuación en el método *optimize* importado del paquete *scipy.*
- Se detallaron métodos para la función de activación, su propagación, determinar su costo, su retro propagación y su entrenamiento. En cuanto al flujo principal se obtienen los datos necesarios para alimentar la red y se hace uso de los métodos detallados con anterioridad.

3.8 Flujo Principal

Se escribió el código sin agrupar en métodos o clases. Se utiliza la manipulación de ficheros para arrastrar los datos a las variables que utilizará la red neuronal para su correcto funcionamiento y hace uso de los métodos definidos en el otro archivo. Por último, categoriza cada ataque de acuerdo a S.T.R.I.D.E. y lo imprime en terminal.

Diccionario de NSL KDD

Tipos de Ataque

S/N	Name	Type
1.	Back	dos
2.	buffer_overflow	u2r
3.	ftp_write	r2l
4.	guess_passwd	r2l
5.	imap	r2l
6.	ipsweep	probe
7.	land	dos
8.	loadmodule	u2r
9.	multihop	r2l
10.	neptune	dos
11.	nmap	probe
12.	perl	u2r
13.	phf	r2l
14.	pod	dos
15.	portsweep	probe
16.	rootkit	u2r
17.	satan	probe
18.	smurf	dos
19.	spy	r2l
20.	teardrop	dos
21.	warezclient	r2l
22.	warezmaster	r2l

Características del Ataque

duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.
num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
serror_rate: continuous.
srv_serror_rate: continuous.
rerror_rate: continuous.
srv_rerror_rate: continuous.
same_srv_rate: continuous.
diff_srv_rate: continuous.
srv_diff_host_rate: continuous.
dst_host_count: continuous.
dst_host_srv_count: continuous.
dst_host_same_srv_rate: continuous.
dst_host_diff_srv_rate: continuous.
dst_host_same_src_port_rate: continuous.
dst_host_srv_diff_host_rate: continuous.
dst_host_serror_rate: continuous.
dst_host_srv_serror_rate: continuous.
dst_host_rerror_rate: continuous.
dst_host_srv_rerror_rate: continuous.

3.9 Pruebas y Errores

Para las pruebas que se hicieron se tomaron a través del proceso de codificación del sistema en donde desde que se empezó a generar la lógica de sistemas y después aplicando unos casos de prueba para poder asegurar el funcionamiento del sistema sabiendo de modificaciones y situaciones específicas que pudiera suceder en un futuro.

La primera sección de pruebas se realizó al ingresar los datos por primera vez, estos datos ingresados mediante archivos de texto permitieron la lectura de cada una de las líneas de manera independiente, pero al inicio se intentó probar por medio de un archivo CSV (delimitado por comas), pero la lectura de esta extensión de archivo genero problemas al correr el programa, se realizaron pruebas y se encontró que la mejor solución era modificar la extensión de este tipo de archivo a texto para que la lectura fuera permitida y a la vez no afectara la ejecución del sistema.

Las siguientes pruebas fueron enfocadas en la parte de lectura e impresión de los pesos generados para poder documentar que pesos aleatorios nos daban en cada ejecución y que pesos entrenados nos generaría, con esta información se permitió ver que tan eficaz era la función y que probabilidades de tasa de éxito se obtendría, de esta manera la ejecución próxima del sistema ejecuto la sección de entrenamiento, se nos dio en primera instancia valores los cuales se alejaban mucho del cero (0) y en donde las iteraciones eran mínimas, pero más que nada generó una advertencia la cual permitió notar que la utilización de pesos aleatorios en algunas ocasiones valores las pequeños que se acercaban tanto a cero influyendo en la funciones dando resultados muy poco asertivos y dando igual a una

tasa de éxito no satisfactoria, este error se generaba porque el lenguaje en el que se desarrolló el sistema las variables tienen asignadas un número menor de byte que no permite que los números muy pequeños sean almacenados los transformaba a cero, por lo que se decidió que las pruebas consiguientes se realicen mediante unos pesos almacenados a les llamaremos indicados, los cuales estos en aleatoriedad no presentan problemas en la ejecución de código y permite que el entrenamiento sea lo más cercano al cero, solucionando en si la advertencia del resultado variable y permitiendo tener más control sobre los resultados a revisar.

De ahí en adelante, se iniciaron las pruebas en base a la tabla de Casos de Prueba (Tabla 2), iniciando la aplicación en donde solo se percibieron errores de sintaxis en momento de algunas comparaciones para poder diferenciar si la red neuronal artificial ya está entrenada o necesita sacar los pesos aleatorios indicados.

Se inició la aplicación sin un archivo de datos para comprobar que es necesario crearlo desde un inicio para poder evitar este error, y que la condición revise si el valor obtenido del archivo de texto leído está vacío, prosiguiendo al entrenamiento y si este ya se encuentra con datos continúe con las pruebas sin necesidad de entrenar.

Comentando inicialmente estos errores tomándolos como principales se continuó con la lista de casos de prueba que se muestra a continuación (Tabla 2), en donde se detalla la fase de prueba individualmente.

Tabla 3.4
Casos de Prueba.

Fase	Nombre	Detalle
Entrenamiento	Iniciar la Aplicación	Se inicia la aplicación con el código creado de acuerdo al diagrama de flujo.
	Iniciar la Aplicación sin los datos de entrenamiento	Se inicia la aplicación sin los datos iniciales en el archivo de texto para probar el comportamiento y que error podría generar.
	Iniciar la Aplicación con un archivo incorrecto de entrenamiento	Se inicia la aplicación sin encontrar el archivo el cual contendría la información para entrenarlo.
Validación	Ejecutar la Aplicación con archivos incorrectos de pesos aleatorios	Se crea un archivo con pesos incorrectos para saber si estos podrían afectar el entrenamiento.

	Ejecutar la Aplicación con archivos incorrectos de pesos	Se agrega un archivo el cual sea difícil detectar ya que los pesos no serán correctos y esto debería provocar errores al iniciar el sistema.
Guardado	Ejecutar la Aplicación en un ambiente en el que no se han guardado resultados	Se realiza para probar la efectividad de guardado de los resultados obtenidos con el fin de poder realizar una observación de los mismos y sacar las conclusiones correspondientes.
	Ejecutar la Aplicación en un ambiente en el que ya se guardaron resultados	Se realiza un guardado para posteriormente verificar si el entrenamiento ha sido correcto y poder permitir en un futuro tener una tasa de éxito mayor.

Tabla 3.4 Casos de Prueba con sus diferentes fases, Elaborada por S.A. Ordoñez, 2021

CAPITULO 4 Resultados

4.1 Resultados esperados y complicaciones

Durante la ejecución del programa se obtuvieron resultados variables cambios debido a los pesos aleatorios generados y al número de iteraciones que se realizaban para que el algoritmo calculara el modelo óptimo.

Los resultados se entregan por parte del sistema de la siguiente manera:

Optimization terminated successfully.

Current function value: 0.000218

Iterations: 3960

Function evaluations: 3983

Gradient evaluations: 3983

Current function value: Valor de la función objetivo a optimizar obtenido por el algoritmo de entrenamiento.

Iterations: Número de veces que se repitió el ciclo de entrenamiento para obtener el resultado

Se imprime el valor de la variable yHat multiplicada por 23, para obtener el valor que cálculo para cada ataque, el valor de y multiplicado por 23 devuelve una guía de respuestas más acertadas. En base a estos datos se calculó el porcentaje de error de la siguiente manera, se calculó el porcentaje de acierto de cada caso:

$$\%\ acierto\ individual = \frac{Resultado}{Resultado\ Esperdo}\ x\ 100$$

Esta operación nos mostrara tanto valores inferiores como mayores al 100 por ciento, esto ya que existen valores identificados como un error, pero el error especifico, por lo tanto, se obtiene el porcentaje de error, restando de un cien por ciento el % de acierto, usando el valor absoluto para y limitando como máximo el cien para tener un valor claro:

$$\%\ de\ error = ABS[100 - \%ai]$$

Posteriormente se obtuvo un promedio de los resultados de porcentajes de error de cada caso:

$$\%\ de\ Error\ Promedio = \frac{1}{n}\sum_{l=1}^{n} e_i$$

Finalmente, para obtener el porcentaje de acierto se le resta el porcentaje promedio a un total de 100:

$$\%\ de\ Acierto = 100 - \%\ de\ Error$$

Con base en este cálculo se obtuvo el siguiente valor como porcentaje de acierto para cada caso:

Tabla 4.1

Porcentaje de Aciertos de cada caso

Porcentaje de Acierto						
Iteraciones	**200**	**400**	**800**	**1600**	**3200**	**Tope**
300 R	66.00	67.13	67.42	65.67	N/A	69.55
900 R	77.58	81.16	82.53	84.61	N/A	85.31
1500 R	82.70	85.77	87.12	87.74	87.20	88.00

Tabla 4.1 Resultados Obtenidos en porcentaje de aciertos por cada caso, Elaborada por S.A. Ordoñez, 2021

Los cuales son graficados de la siguiente manera:

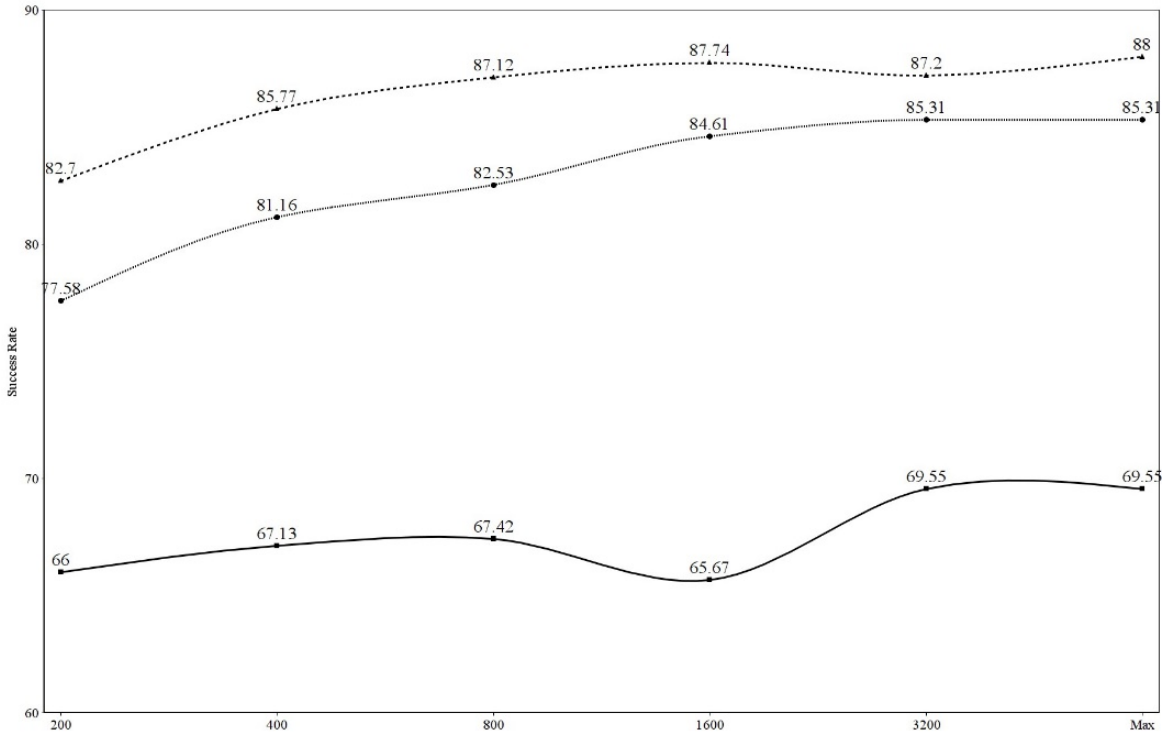


Figura 4. Gráficos en base a los resultados obtenidos en cada caso, Elaborada por S.A. Ordoñez, 2021

4.2 Análisis de resultados

Los porcentajes de acierto individuales de cada caso presentan variaciones, pero como se puede observar en la siguiente gráfica, se mantiene en torno al 100%, para los casos en los que se supera el 100% se identifican como casos en los que un error fue reconocido como tal, pero no como el correcto tipo de error. Con estos valores se puede decir que la red neuronal bajo estas condiciones de entrenamiento tiene un porcentaje de acierto del **88.0 %.**

Como se puede observar los resultados obtenidos al utilizar la cantidad de 1500, 1050 de entrenamiento y 450 para realizar la prueba, se puede percatar que, si bien el valor de la función es cercano a cero (0.000218), es decir, que el aprendizaje ha sido en su mayoría exitoso, el valor de las salidas (yHat) en respuesta a los ataques introducidos en el testX aun presentan error en algunas ocasiones de las salidas correctas (Y). Se puede determinar entonces que, aunque no tiene en porcentaje de éxito del 100% de éxito la red es suficientemente “inteligente” para identificar los ataques.

Para que las salidas sean más cercanas con una tasa de éxito mayor se tendría que utilizar una mayor cantidad de datos preseleccionados y clasificados al momento de entrenar la red neuronal artificial, así como también el porcentaje de ataques debería ser más equitativo ya que como se puede observar en la sección de materiales la cantidad de tipos de ataques varía mucho. Esto provoca que la diferencia de cantidades en casos de entrenamiento genere un posible error en las pruebas de identificación de ataques.

Se prevé que en futuras pruebas se pueda utilizar más de los datos obtenidos del conjunto de datos con el fin de probar más favorablemente que la red neuronal artificial puede ser más exacta, debiendo promover la utilización de este tipo de sistemas permitirá que la identificación de posibles amenazas sea más inteligente, convirtiendo los sistemas organizacionales en el área de seguridad de la información en sectores de alta tecnología especializadas, erradicando los futuros ataques, ya que se buscaría un posible autoaprendizaje de parte de los sistemas, detectando nuevos ataques y evitado arrojar falsos positivos.

Conclusiones del Capítulo.

El presente capitulo nos permite visualizar un modelo ajustable en el cual pudimos por medio del método de segmentación y clasificación de ataques reconocer un porcentaje representativo de ataques, por consiguiente se planean seguir ajustando los valores de aprendizaje para obtener un valor parecido al de otros autores, se tiene como objetivo incorporar algunos métodos para mezclar diversas tecnologías y herramientas de Inteligencia Artificial con la finalidad de poder incrementar la efectividad de nuestro sistema de seguridad.

CAPÍTULO 5 Conclusiones y trabajo futuros

5.1 Conclusiones

En la presente investigación se construyó un modelo de seguridad basado en Inteligencia Artificial, está fundamentado en la versatilidad que proveen las redes neuronales artificiales, la manipulación de información, el uso de diversos métodos de entrenamiento, las redes neuronales artificiales con una arquitectura Backpropagation, gracias a esta herramienta desarrollada se pudo identificar el comportamiento de los ataques de red establecidos por data sets previamente elaborados por diversos organismos encargados de análisis de riesgos.

El alcance esperado, consistía en crear un mecanismo de detección de Vulnerabilidades, buscando catalogar mediante el protocolo de clasificación STRIDE de Microsoft, los diversos ataques para identificar el tipo de ataque que se está llevando a cabo con la posibilidad de predecir y tomar medidas de prevención en nuestros sistemas de seguridad pero adicionalmente se obtuvo información de 23 tipos de ataque que pudieron ser reflejados como la base de la clasificación de STRIDE y los cuales pueden tener 1 o más de estos tipos de ataques con esto se identificaron los ataques durante el entrenamiento.

Dentro de la presente investigación se encontró un Dataset llamado KDDCUP99 que ha sido utilizado en múltiples investigaciones como un Dataset con millones de ataques, después de un proceso de búsqueda de información se encontró que el Dataset contiene una gran variante de ataques, pero muchos de estos utilizan la misma metodología de buscar vulnerabilidades, se encontraron registros duplicados y en base a esto se optimizo la base utilizando el Dataset NLS KDD obteniendo con esto optimizar hasta un 80% la cantidad de ataques dado que utilizaban el mismo método de ataque, este Dataset es muy utilizado y para los fines prácticos nos sirvió para analizar información depurada para su clasificación.

- Para el proceso de clasificación se utilizó la Herramienta STRIDE creada por Microsoft para segmentar los ataques utilizando NSL KDD.
- La Red neuronal fue entrenada modificando sus pesos de entrada y así mismo la cantidad de registros y de iteraciones que hacia esta misma.
- Los resultados obtenidos nos permitieron reentrenar la Red Neuronal hasta obtener el porcentaje de eficiencia más alto que nuestro sistema pudo obtener que fue el 88% de efectividad con 1500 datos.

Con esto se acepta nuestra Hipótesis primaria que establece que la Red Neuronal puede reducir el índice de error de reconocimiento de ataques con un entrenamiento y clasificación adecuado. Lo cual se demostró al hacer la experimentación y con los resultados obtenidos durante dicho proceso.

5.2 Resultados de experimentación

Es importante considerar que el resultado en la tabla 3, nos indica que mientras más iteraciones se realizaron a mayor cantidad de elementos para identificar aciertos el porcentaje fue en aumento, con lo cual obtenemos que nuestra red neuronal mientras más sea probada puede identificar una cantidad más grandes de ataques, buscando con esto la optimización en comparación a otras metodologías que obtienen porcentajes de acierto más altos, así mismo, se establecen los parámetros para obtener los resultados óptimos de este proceso.

Según la teoría propuesta coincidimos con los Autores como Wenke lee y Chimphlee que concuerdan que la lógica difusa implementada en Datasets aumenta el porcentaje de acierto, dado que nos da patrones de reconocimiento más específicos lo que permite un entrenamiento con mayor exactitud que lo que puede darnos una red neuronal, por lo tanto se establece que para próximas investigaciones se puede establecer más de un método de entrenamiento o unir diversos métodos para la identificación de ataques.

Así mismo se descartan teorías como las de Ghosh debido a que sus resultados fueron menos precisos que los de esta investigación, con lo cual nos alienta a desestimar el uso de redes neuronales de una sola capa y poder buscar combinación de diversos métodos, y se marca una tendencia a corroborar que los ataques cada vez tienen comportamientos más ambiguos lo que hace que un solo

método de detección no sea suficiente para detectar los comportamientos cada vez más complejos de las incidencias.

Se prevé que en futuras pruebas será posible utilizar más datos obtenidos del conjunto de datos para probar que la red neuronal artificial puede ser más precisa y debe promover el uso de este tipo de sistemas para la identificación de Posibles amenazas. Esto podría convertir los sistemas organizativos en el área de seguridad de la información en sectores especializados de alta tecnología, erradicando futuros ataques, ya que buscaría un posible autoaprendizaje de los sistemas, detectando nuevos ataques y evitando lanzar falsos positivos.

Además de esto, se podría usar un algoritmo evolutivo para evaluar el sistema, y así poder encontrar los datos que optimizan la red, descartando los conjuntos más débiles y, por lo tanto, producir mejores resultados cuando se trata de identificar los paquetes de información del computador.

En base a esto se propone el sistema como una medida alternativa usando Inteligencia Artificial con la capacidad de obtener un índice adecuado de confiabilidad basado en diferentes pruebas, buscando la manera de optimizarlo en diferentes entornos, así como en diferentes tipos de ataques y variaciones de estos.

La cantidad de ataques y la variedad de ataques nos invitan a seguir haciendo pruebas con diferentes variaciones de estos mismos, buscando reducir de manera adecuada la capacidad de nuestra red neuronal y poder lograr hacer más dinámico el entrenamiento de la misma.

5.3 Trabajos futuros

Como parte del proceso de enriquecimiento e implementación de la red neuronal dentro de ambientes controlados y en la búsqueda de poder optimizar el uso de la misma en diferentes entornos se prevé utilizarla misma red viendo el comportamiento de los ataques en sitios controlados.

Las pruebas de Monitoreo de Ataques se llevarán a cabo en 3 etapas diferentes, tratando de establecer los indicadores de comportamiento en las que nuestra herramienta podría funcionar. Los escenarios se mencionan a continuación.

1.- OSSIM Alien Vault: SIEM (Security information and event management) (Utilizar software externo para monitoreo)

Es una herramienta para monitorear registros de código abierto. Permite tener un entorno gráfico para llevar a cabo diversos ataques y mostrar los resultados de los mismos en un blog.

Para esto, las instalaciones para hacer las pruebas necesarias se instalarán en una Máquina Virtual.

Y poder ver el comportamiento del sistema para aprovechar y poder implementar prácticas de este sistema en nuestro proyecto. La implementación se muestra en la siguiente Figura 5.

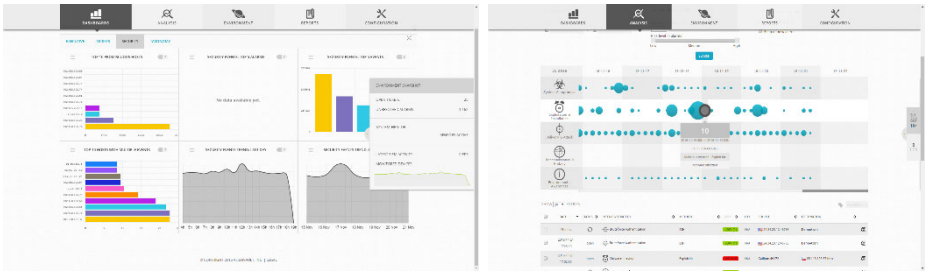


Figura 5. Instalación de OSSIM ALIEN VAULT en Máquina virtual, Elaborada por S.A. Ordoñez, 2021

2.- Ubuntu Server (Creación de entornos en diferentes sistemas operativos): Se instalará una máquina virtual con el sistema operativo Ubuntu en su versión de servidor, en el cual se instalarán los elementos básicos de cualquier servidor a nivel de empresa, como servidor de correo, DHCP, FTP, servicios web, servicios de bases de datos, etc. Para realizar pruebas de penetración en un entorno común utilizado por el usuario promedio.

3.- NETinVM: NETinVM (Redes virtuales para ambientes controlados) es una imagen de máquina virtual VMWare que contiene, lista para ejecutarse, una serie de máquinas virtuales Linux (UML) en modo Usuario. Cuando se inician, las máquinas virtuales UML forman una red de computadoras.

Además, se prevé utilizar diversos modelos de algoritmos evolutivos con la finalidad de optimizar cada vez más el proceso de fortalecimiento de nuestra red neuronal.

5.4 Recomendaciones

Dada la presente investigación se recomienda que la inversión en seguridad de la información pueda establecerse como un fundamento critico en la planeación estratégica de los diversos organismos que manejen información sensible, ya que cada vez nos enfrentamos a retos de ataques más inteligentes y que varían su comportamiento con patrones cada vez más difíciles de identificar y clasificar.

El uso de inteligencia para poder detectar estas amenazas se vuelve un parámetro fundamental ya que el dejar la responsabilidad a los responsables de sistemas de las organizaciones, puede no dar respuesta de manera adecuada a la variación de metodologías que buscan vulnerar nuestros sistemas.

Además podemos establecer como una necesidad primaria el tomar un interés en este tema, dados los resultados de los últimos reportes globales, donde vemos que el manejo y uso de información pasa a los primeros planos dentro de las empresas, organismos y hasta gobiernos, lo cual pone un foco de atención especial en este tema.

Y ante el déficit de expertos en seguridad necesitamos sistemas que puedan ser supervisados pero con una capacidad de hacer más robusto su aprendizaje para que detecten información que a los expertos pueden no identificar.

REFERENCIAS

.- de Pablos Heredero, C. (2004). *Informática y comunicaciones en la empresa. Biblioteca de economía y finanzas. Libros profesionales.* ESIC Editorial.

A. Goyal and C. Kumar (2007), "GA-NIDS: A Genetic Algorithm based Network Intrusion Detection System,".

A. K. Ghosh, and A. Schwartzbard. (1999). A Study in Using Neural Networks for Anomaly and Misuse Detection. *USENIX Security*.

Bagheri, E., Ghorbani, A. A., Lu, W., & Tavallaee, M. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. *CISDA*.

Casal, J., & Mateu, E. (2003). *Tipos de muestreo.* Barcelona: Rev. Epidem. Med. Prev. (2003), 1: 3-7 .

Charniak, Eugene, Mcdermott, Drew. (1986). *Introduction to Artificial Intelligence.*

CISCO. (2018). *Reporte Anual de Seguridad CISCO.*

de Pablos Heredero, C. (2004). *Informática y comunicaciones en la empresa. Biblioteca de economía y finanzas. Libros profesionales.* ESIC Editorial.

E. Jhordany Serna Valdivia and J. Mejía Miranda, "Proposal of a Intelligent Agent for Management and Mitigation in Cibersecurity Risk for IoT Environments," (2020), 9th International Conference On Software Process Improvement (CIMPS), 2020, pp. 148-154, doi: 10.1109/CIMPS52057.2020.9390114.

Frawley, W. J., Matheus, C. J., & Piatetsky-Shapiro, G. (1992). Knowledge Discovery in Databases: An Overview. *AI Magazine, 13*(3), 57-70.

Gacharná G., F. I. (2009). Hacker ético vs. delincuente informático: Una mirada en el contexto colombiano. *INVENTUM, 4*(6), 46-49. doi:10.26620/uniminuto.inventum.4.6.2009.46-49

J. K. Williams, J. Craig, A. Cotter, and J. K. Wolff. (2007). A Hybrid Machine Learning And Fuzzy Logic Approach to Cit Diagnostic Development. *AMS Fifth Conference on Artificial Intelligence*.

Jara, H., & Pacheco, F. G. (2012). *Ethical Hacking 2.0.* Buenos Aires: RedUsers Usershop.

JL McClelland, D. R. (1986). *Parallel distributed processing.*

Li, W. (2004). Using genetic algorithm for network intrusion detection. *Proc. of the United States Department of Energy Cyber Security*, 1-8.

M. Luck, P. McBurney and C. Preist (2003), Agent Technology: Next Generation Computing, AgentLink II.

Mata, M. C., & Macassi, S. (1997). *Cómo elaborar muestras para los sondeos de audiencias* (Vol. V). Quito: ALER.

Mieres, J. (2009). *Debilidades de seguridad comúnmente explotadas.* Obtenido de http://proton.ucting.udg.mx/tutorial/hackers/hacking.pdf

O. Cordon, P. Kazienko and B. Trawinski. (2011). Special Issue on Hybrid and Ensemble Methods in Machine Learning. *New Generation Computing*, 241-244.

Pavlova, Galya & Tsochev, Georgi & Yoshinov, Radoslav & Trifonov, Roumen & Manolov, Slavcho. (2017). Increasing the level of network and information security using artificial intelligence. 83-88. 10.15224/978-1-63248-131-3-25.

Russell, S. /. (2004). *Inteligencia Artificial. Un Enfoque Moderno / 2 Ed.* PEARSON.

Tyugu, E. (2011). Artificial Intelligence in Cyber Defense. *3rd International Conference on Cyber Conflict*.

W. Chimphlee, A. H. Abdullah, M. N. M. Sap, S. Srinoy, and S. Chimphlee. (2006). Anomaly-based intrusion detection using fuzzy rough clustering. *ICHIT'06*, 329-334.

W. Lee, S. J. Stolfo, and K. W. Mok. (1999.). "Mining in a data-flow environment: Experience in network intrusion detection. *ACM*, 114-124.

W. Li. (2004), "Using Genetic Algorithm for Network Intrusion Detection," Proc. of the United States Department of Energy Cyber Security Group

WEF. (2020). *The Global Risk Report 2020.*

Wenke Lee, S. J. Stolfo and K. W. Mok. (1999). A data mining framework for building intrusion detection models. *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344)*, 120-132.

Z. Ma and A. Kaban. (2013). K-Nearest-Neighbours with a novel similarity measure for intrusion detection. *2013 13th UK Workshop on Computational Intelligence (UKCI), Guildford*, 266-271.

Forbes Mexico . (21 de mayo de 2019). Obtenido de Fraude cibernético y robo de datos: riesgo global: https://www.forbes.com.mx/fraude-cibernetico-y-robo-de-datos-riesgo-global/

World Economic Forum . (21 de Diciembre de 2020). Obtenido de https://es.weforum.org/reports

CAPÍTULO 6 Anexos

Código Entrenamiento de Red Neuronal

Archivo: main.py

 # Inicio Imports

```
# -*- coding: utf-8 -*-
import numpy as np
from redNeuronal import Neural_Network, trainer #computeNumericalGradient
from io import open
```

 # Fin Imports

 # Inicio del programa

```
# Revisar que pasa si no existe el archivo
fichero = open('pesosEntrenadosW1.txt', 'w')
fichero.close()
fichero = open('pesosEntrenadosW1.txt', 'r')
entrada1 = fichero.read()
fichero.close()
fichero = open('pesosEntrenadosW2.txt', 'w')
fichero.close()
fichero = open('pesosEntrenadosW2.txt', 'r')
entrada2 = fichero.read()
fichero.close()

RN = Neural_Network()
RN.randWeight()

# While ¿no cuente con previo pesos /entrenamiento?
while entrada1 != "" or entrada2 != "":
    ## Recibir datos de entrenamiento
    fichero = open('trainX.txt', 'r')
    lineas = fichero.readlines()
    fichero.close()
    X = []
```

```
    for l in range(len(lineas)):
        linea = lineas[l]
        linea = linea.replace("\n", "").split(",")
        for n in range(len(linea)):
            linea[n] = float(linea[n])
        X.append(linea[:])

    fichero = open('trainy.txt', 'r')
    lineas = fichero.readlines()
    fichero.close()
    y = []
    for l in range(len(lineas)):
        linea = lineas[l]
        linea = linea.replace("\n", "").split(",")
        for n in range(len(linea)):
            linea[n] = float(linea[n])
        y.append([linea[0]])

    X = np.array(X, dtype=float)
    y = np.array(y, dtype=float)
    y = y/23

    ## Entrenamiento y guarda pesos
    T = trainer(RN)
    T.train(X,y)

    fichero = open('pesosEntrenadosW1.txt', 'r')
    entrada1 = fichero.read()
    fichero.close()
    fichero = open('pesosEntrenadosW2.txt', 'r')
    entrada2 = fichero.read()
    fichero.close()
# Recibir datos de entrada y avanzar en la red
fichero = open('testX.txt', 'r')
lineas = fichero.readlines()
fichero.close()
X = []
for l in range(len(lineas)):
    linea = lineas[l]
    linea = linea.replace("\n", "").split(",")
    for n in range(len(linea)):
        linea[n] = float(linea[n])
    X.append(linea[:41])
```

```
fichero = open('testy.txt', 'r')
lineas = fichero.readlines()
fichero.close()
y = []
for l in range(len(lineas)):
    linea = lineas[l]
    linea = linea.replace("\n", "").split(",")
    for n in range(len(linea)):
        linea[n] = float(linea[n])
    y.append([linea[0]])
X = tuple(X)
y = tuple(y)

X = np.array(X, dtype=float)
y = np.array(y, dtype=float)
y = y/23

yHat = RN.forward(X)
# Generar y guardar resultados
rYHat = np.around(yHat*23)
norm, spoof, tamp, repu, info, deni, elev = 0,0,0,0,0,0,0
for i in range(len(rYHat)):
    au = int(rYHat[i])
    if au == 0 or au == 1:
        norm += 1
    if au == 4 or au == 5 or au == 7 or au == 8:
        spoof += 1
    if au == 12 or au == 13 or au == 17:
        tamp += 1
    if au == 3 or au == 18:
        repu += 1
    if au == 14 or au == 19 or au == 22:
        info += 1
    if au == 2 or au == 6 or au == 9 or au == 10 or au == 11 or au == 20:
        deni += 1
    if au == 15 or au == 16 or au == 21 or au == 23:
        elev += 1
print("\nSe encontraron los siguientes paquetes:\n{} normal\n{} spoofing\n \
{} tampering\n{} repudiation\n{} disclosure\n{} denial\n \
{} elevation".format(norm, spoof, tamp, repu, info, deni, elev))
```

 # Fin Programa

```
# -*- coding: utf-8 -*-
import numpy as np
from redNeuronal import Neural_Network, trainer #computeNumericalGradient
from io import open
```

```
fichero = open('pesosEntrenadosW1.txt', 'w')
fichero.close()
fichero = open('pesosEntrenadosW1.txt', 'r')
entrada1 = fichero.read()
fichero.close()
fichero = open('pesosEntrenadosW2.txt', 'w')
fichero.close()
fichero = open('pesosEntrenadosW2.txt', 'r')
entrada2 = fichero.read()
fichero.close()
```

```
RN = Neural_Network()
RN.randWeight()
#RN.dfltWeight()

# While ¿no cuente con previo pesos /entrenamiento?
while entrada1 is "" or entrada2 is "":
    ## Recibir datos de entrenamiento
    fichero = open('trainX.txt', 'r')
    lineas = fichero.readlines()
    fichero.close()
    X = []
    for l in range(len(lineas)):
        linea = lineas[l]
        linea = linea.replace("\n", "").split(",")
        for n in range(len(linea)):
            linea[n] = float(linea[n])
        X.append(linea[:])

    fichero = open('trainy.txt', 'r')
    lineas = fichero.readlines()
    fichero.close()
    y = []
    for l in range(len(lineas)):
        linea = lineas[l]
        linea = linea.replace("\n", "").split(",")
        for n in range(len(linea)):
            linea[n] = float(linea[n])
        y.append([linea[0]])

    X = np.array(X, dtype=float)
    y = np.array(y, dtype=float)
    y = y/23

    ## Entrenamiento y guarda pesos
    T = trainer(RN)
    T.train(X,y)

    fichero = open('pesosEntrenadosW1.txt', 'r')
    entrada1 = fichero.read()
    fichero.close()
    fichero = open('pesosEntrenadosW2.txt', 'r')
    entrada2 = fichero.read()
    fichero.close()
```

```
# Recibir datos de entrada y avanzar en la red
fichero = open('testX.txt', 'r')
lineas = fichero.readlines()
fichero.close()
X = []
for l in range(len(lineas)):
    linea = lineas[l]
    linea = linea.replace("\n", "").split(",")
    for n in range(len(linea)):
        linea[n] = float(linea[n])
    X.append(linea[:41])

fichero = open('testy.txt', 'r')
lineas = fichero.readlines()
fichero.close()
y = []
for l in range(len(lineas)):
    linea = lineas[l]
    linea = linea.replace("\n", "").split(",")
    for n in range(len(linea)):
        linea[n] = float(linea[n])
    y.append([linea[0]])
X = tuple(X)
y = tuple(y)

X = np.array(X, dtype=float)
y = np.array(y, dtype=float)
y = y/23

yHat = RN.forward(X)
```

```python
# Generar y guardar resultados
rYHat = np.around(yHat*23)
norm, spoof, tamp, repu, info, deni, elev = 0,0,0,0,0,0,0
for i in range(len(rYHat)):
    au = int(rYHat[i])
    if au == 0 or au == 1:
        norm += 1
    if au == 4 or au == 5 or au == 7 or au == 8:
        spoof += 1
    if au == 12 or au == 13 or au == 17:
        tamp += 1
    if au == 3 or au == 18:
        repu += 1
    if au == 14 or au == 19 or au == 22:
        info += 1
    if au == 2 or au == 6 or au == 9 or au == 10 or au == 11 or au == 20:
        deni += 1
    if au == 15 or au == 16 or au == 21 or au == 23:
        elev += 1
print("\nSe encontraron los siguientes paquetes:\n{} normal\n{} spoofing\n \
{} tampering\n{} repudiation\n{} disclosure\n{} denial\n \
{} elevation".format(norm, spoof, tamp, repu, info, deni, elev))
```

 # Inicio Imports

```
# Red Neuronal - Backpropagation with BFGS as optimizer
import numpy as np
from scipy import optimize
from io import open
import pickle
```

 # Fin Imports

 # Inicio Programa

 # Inicio Clase Neural_Network

```
class Neural_Network():
    def __init__(self):
        #Capas de la red = Neuronas por capa
        self.inputLayerSize = 41
        self.hiddenLayerSize = 20
        self.outputLayerSize = 1

    def randWeight(self):
        #Se crean matrices con los pesos de los parametros aleatorios
        self.W1 = np.random.randn(self.inputLayerSize,self.hiddenLayerSize)
        self.W2 = np.random.randn(self.hiddenLayerSize,self.outputLayerSize)

        #Se guarda en ficheros los pesos
        fichero = open('binInicialesW1.pckl', 'wb')
        pickle.dump(self.W1, fichero)
        fichero.close()
        self.P1 = str(self.W1)
        fichero = open('pesosInicialesW1.txt', 'w')
        fichero.write(self.P1)
        fichero.close()

        fichero = open('binInicialesW2.pckl', 'wb')
        pickle.dump(self.W2, fichero)
        fichero.close()
        self.P2 = str(self.W2)
        fichero = open('pesosInicialesW2.txt', 'w')
        fichero.write(self.P2)
        fichero.close()


    def dfltWeight(self):
```

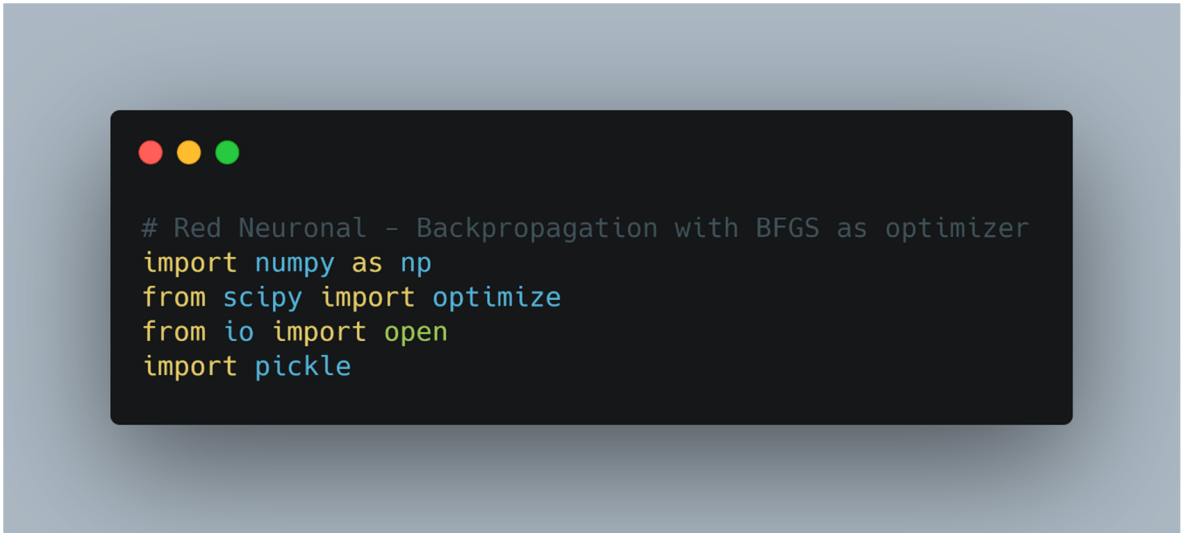
```
    #Se crean matrices con los pesos de los parametros predefinidos
    fichero = open('binInicialesW1.pckl', 'rb')
    self.W1 = pickle.load(fichero)
    fichero.close

    fichero = open('binInicialesW2.pckl', 'rb')
    self.W2 = pickle.load(fichero)
    fichero.close

def sigmoid(self, z):
    """Funcion de activacion a escalar, vector o matriz"""
    #1/(1+eulier^-z)
    return 1/(1+np.exp(-z))

def forward(self, X):
    """Propaga las entradas en la red"""
    self.z2 = np.dot(X, self.W1) #Producto entradas - pesos1
    self.a2 = self.sigmoid(self.z2) #Aplica funcion sigmoide a lo anterior
    self.z3 = np.dot(self.a2, self.W2) #Productos capa oculta - pesos2
    yHat = self.sigmoid(self.z3) #Capa de salida
    return yHat

def costFunction(self, X, y):
    """Calcula costo -MSE- para la X y y dadas"""
    self.yHat = self.forward(X)
    #Funcion de costo
    J = 0.5*sum((y-self.yHat)**2)
    return J

def sigmoidPrime(self,z):
    """Gradiente de funcion sigmoide"""
    return np.exp(-z)/((1+np.exp(-z))**2)

def costFunctionPrime(self, X, y):
    #Indica la direccion de la funcion del costo
    #Calcula la derivada con respecto a los pesos para una X y y dadas
    self.yHat = self.forward(X)
    #Derivada de salidas a capa escondida
    delta3 = np.multiply(-(y-self.yHat), self.sigmoidPrime(self.z3))
    dJdW2 = np.dot(self.a2.T, delta3)
    #Derivada de capa escondida a entradas
    delta2 = np.dot(delta3, self.W2.T)*self.sigmoidPrime(self.z2)
    dJdW1 = np.dot(X.T, delta2)
    return dJdW1, dJdW2
```

```
    #Funciones de apoyo para interactuar con otras clases:
    def getParams(self):
        #Convierte los pesos en vector:
        params = np.concatenate((self.W1.ravel(), self.W2.ravel()))
        return params

    def setParams(self, params):
        #Prepara los pesos usando un vector de parametro simple
        W1_start = 0
        W1_end = self.inputLayerSize * self.hiddenLayerSize
        self.W1 = np.reshape(params[W1_start:W1_end], (self.inputLayerSize , self.hiddenLayerSize))
        W2_end = W1_end + self.hiddenLayerSize*self.outputLayerSize
        self.W2 = np.reshape(params[W1_end:W2_end], (self.hiddenLayerSize, self.outputLayerSize))

        #Se guarda en ficheros los pesos
        self.P1 = str(self.W1)
        fichero = open('pesosEntrenadosW1.txt', 'w')
        fichero.write(self.P1)
        fichero.close()
        self.P2 = str(self.W2)
        fichero = open('pesosEntrenadosW2.txt', 'w')
        fichero.write(self.P2)
        fichero.close()

    def computeGradients(self, X, y):
        dJdW1, dJdW2 = self.costFunctionPrime(X, y)
        return np.concatenate((dJdW1.ravel(), dJdW2.ravel()))
```

 # Fin class Neural_Network
 # Inicio class trainer

```
class trainer():
    def __init__(self, N):
        #Crea una referencia local a la red:
        self.N = N

    def callbackF(self, params):
        self.N.setParams(params)
        self.J.append(self.N.costFunction(self.X, self.y))

    def costFunctionWrapper(self, params, X, y):
        self.N.setParams(params)
        cost = self.N.costFunction(X, y)
        grad = self.N.computeGradients(X,y)
        return cost, grad
```

```
    def train(self, X, y):
        #Crea una variable interna par la funcion de llamada:
        self.X = X
        self.y = y

        #Crea una lista vacia a los costos guardados:
        self.J = []

        params0 = self.N.getParams()

        options = {'maxiter': 200, 'disp' : True}
        _res = optimize.minimize(self.costFunctionWrapper, params0, jac=True, method='BFGS', \
                                 args=(X,y), options=options, callback=self.callbackF)

        self.N.setParams(_res.x)
        self.optimizationResults = _res

# Fin clase trainer
def computeNumericalGradient(N, X, y):
    """Solo es para comprobar que costFunctionPrime este bien definida """
    paramsInitial = N.getParams()
    numgrad = np.zeros(paramsInitial.shape)
    perturb = np.zeros(paramsInitial.shape)
    e = 1e-4

    for p in range(len(paramsInitial)):
        #Asigna vector de perturbacion
        perturb[p] = e
        N.setParams(paramsInitial + perturb)
        loss2 = N.costFunction(X, y)

        N.setParams(paramsInitial - perturb)
        loss1 = N.costFunction(X, y)

        #Calcular Gradiente numerico
        numgrad[p] = (loss2 - loss1) / (2*e)

        #Regresa el valor cambiado a cero:
        perturb[p] = 0

    #Regresa parametros a valor original:
    N.setParams(paramsInitial)

    return numgrad
```


# Red Neuronal - Backpropagation with BFGS as optimizer
import numpy as np
from scipy import optimize
from io import open
import pickle

```python
class Neural_Network():
    def __init__(self):
        #Capas de la red = Neuronas por capa
        self.inputLayerSize = 41
        self.hiddenLayerSize = 20
        self.outputLayerSize = 1

    def randWeight(self):
        #Se crean matrices con los pesos de los parametros aleatorios
        self.W1 = np.random.randn(self.inputLayerSize,self.hiddenLayerSize)
        self.W2 = np.random.randn(self.hiddenLayerSize,self.outputLayerSize)

        #Se guarda en ficheros los pesos
        fichero = open('binInicialesW1.pckl', 'wb')
        pickle.dump(self.W1, fichero)
        fichero.close()
        self.P1 = str(self.W1)
        fichero = open('pesosInicialesW1.txt', 'w')
        fichero.write(self.P1)
        fichero.close()

        fichero = open('binInicialesW2.pckl', 'wb')
        pickle.dump(self.W2, fichero)
        fichero.close()
        self.P2 = str(self.W2)
        fichero = open('pesosInicialesW2.txt', 'w')
        fichero.write(self.P2)
        fichero.close()


    def dfltWeight(self):
        #Se crean matrices con los pesos de los parametros predefinidos
        fichero = open('binInicialesW1.pckl', 'rb')
        self.W1 = pickle.load(fichero)
        fichero.close

        fichero = open('binInicialesW2.pckl', 'rb')
        self.W2 = pickle.load(fichero)
        fichero.close

    def sigmoid(self, z):
        """Funcion de activacion a escalar, vector o matriz"""
        #1/(1+eulier^-z)
        return 1/(1+np.exp(-z))

    def forward(self, X):
        """Propaga las entradas en la red"""
        self.z2 = np.dot(X, self.W1) #Producto entradas - pesos1
        self.a2 = self.sigmoid(self.z2) #Aplica funcion sigmoide a lo anterior
        self.z3 = np.dot(self.a2, self.W2) #Productos capa oculta - pesos2
        yHat = self.sigmoid(self.z3) #Capa de salida
        return yHat

    def costFunction(self, X, y):
        """Calcula costo -MSE- para la X y y dadas"""
        self.yHat = self.forward(X)
        #Funcion de costo
        J = 0.5*sum((y-self.yHat)**2)
        return J

    def sigmoidPrime(self,z):
        """Gradiente de funcion sigmoide"""
        return np.exp(-z)/((1+np.exp(-z))**2)

    def costFunctionPrime(self, X, y):
        #Indica la direccion de la funcion del costo
        #Calcula la derivada con respecto a los pesos para una X y y dadas
        self.yHat = self.forward(X)
        #Derivada de salidas a capa escondida
        delta3 = np.multiply(-(y-self.yHat), self.sigmoidPrime(self.z3))
        dJdW2 = np.dot(self.a2.T, delta3)
        #Derivada de capa escondida a entradas
        delta2 = np.dot(delta3, self.W2.T)*self.sigmoidPrime(self.z2)
        dJdW1 = np.dot(X.T, delta2)
        return dJdW1, dJdW2

    #Funciones de apoyo para interactuar con otras clases:
    def getParams(self):
        #Convierte los pesos en vector:
        params = np.concatenate((self.W1.ravel(), self.W2.ravel()))
        return params

    def setParams(self, params):
        #Prepara los pesos usando un vector de parametro simple
        W1_start = 0
        W1_end = self.inputLayerSize * self.hiddenLayerSize
        self.W1 = np.reshape(params[W1_start:W1_end], (self.inputLayerSize , self.hiddenLayerSize))
        W2_end = W1_end + self.hiddenLayerSize*self.outputLayerSize
        self.W2 = np.reshape(params[W1_end:W2_end], (self.hiddenLayerSize, self.outputLayerSize))

        #Se guarda en ficheros los pesos
        self.P1 = str(self.W1)
        fichero = open('pesosEntrenadosW1.txt', 'w')
        fichero.write(self.P1)
        fichero.close()
        self.P2 = str(self.W2)
        fichero = open('pesosEntrenadosW2.txt', 'w')
        fichero.write(self.P2)
        fichero.close()

    def computeGradients(self, X, y):
        dJdW1, dJdW2 = self.costFunctionPrime(X, y)
        return np.concatenate((dJdW1.ravel(), dJdW2.ravel()))
```

```python
class trainer():
    def __init__(self, N):
        #Crea una referencia local a la red:
        self.N = N

    def callbackF(self, params):
        self.N.setParams(params)
        self.J.append(self.N.costFunction(self.X, self.y))

    def costFunctionWrapper(self, params, X, y):
        self.N.setParams(params)
        cost = self.N.costFunction(X, y)
        grad = self.N.computeGradients(X,y)
        return cost, grad

    def train(self, X, y):
        #Crea una variable interna par la funcion de llamada:
        self.X = X
        self.y = y

        #Crea una lista vacia a los costos guardados:
        self.J = []

        params0 = self.N.getParams()

        options = {'maxiter': 200, 'disp' : True}
        _res = optimize.minimize(self.costFunctionWrapper, params0, jac=True, method='BFGS', \
                                 args=(X,y), options=options, callback=self.callbackF)

        self.N.setParams(_res.x)
        self.optimizationResults = _res
```

Casos de Prueba

Fase	Nombre
Entrenamiento	Iniciar la Aplicación
	Iniciar la Aplicación sin los datos de entrenamiento
	Iniciar la Aplicación con un archivo incorrecto de entrenamiento
	Iniciar la Aplicación con archivos incorrectos de entrenamiento
Validación	Ejecutar la Aplicación con archivos incorrectos de pesos aleatorios
	Ejecutar la Aplicación con un archivo incorrecto de pesos aleatorios
	Ejecutar la Aplicación con archivos incorrectos de pesos
	Ejecutar la Aplicación con un archivo incorrecto de pesos
	Ejecutar la Aplicación especificando el índice de la línea
	Ejecutar la Aplicación sin especificar el índice de la línea
Guardado	Ejecutar la Aplicación en un ambiente en el que no se han guardado resultados
	Ejecutar la Aplicación en un ambiente en el que ya se guardaron resultados

Clasificación STRIDE

	S	T	R	I	D	E
ATTACK_TYPE	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privileges
normal	0	0	0	0	0	0
neptune	0	0	0	0	1	0
warezclient	0	0	1	0	0	0
ipsweep	1	0	0	0	0	0
portsweep	1	0	0	0	0	0
teardrop	0	0	0	0	1	0
nmap	1	0	0	0	0	0
satan	1	0	0	0	0	0
smurf	0	0	0	0	1	0
pod	0	0	0	0	1	0
back	0	0	0	0	1	0
guess_passwd	0	1	0	0	0	0
ftp_write	0	1	0	0	0	0
multihop	0	0	0	1	0	0
rootkit	0	0	0	0	0	1
buffer_overflow	0	0	0	0	0	1
imap	0	1	0	0	0	0
warezmaster	0	0	1	0	0	0
phf	0	0	0	1	0	0
land	0	0	0	0	1	0
loadmodule	0	0	0	0	0	1
spy	0	0	0	1	0	0
perl	0	0	0	0	0	1

Clasificación STRIDE por TIPO

ATTACK_TYPE	ATTACK_CATEGORY	STRIDE_ATTACK_TYPE
normal	NORMAL	NORMAL
neptune	DOS	DENIAL OF SERVICE
warezclient	R2L	REPUDIATION
ipsweep	PROBING	SPOOFING
portsweep	PROBING	SPOOFING
teardrop	DOS	DENIAL OF SERVICE
nmap	PROBING	SPOOFING
satan	PROBING	SPOOFING
smurf	DOS	DENIAL OF SERVICE
pod	DOS	DENIAL OF SERVICE
back	DOS	DENIAL OF SERVICE
guess_passwd	R2L	TAMPERING
ftp_write	R2L	TAMPERING
multihop	R2L	INFORMATION DISCLOSURE
rootkit	U2R	ELEVATION OF PRIVILEGES
buffer_overflow	U2R	ELEVATION OF PRIVILEGES
imap	R2L	TAMPERING
warezmaster	R2L	REPUDIATION
phf	R2L	INFORMATION DISCLOSURE
land	DOS	DENIAL OF SERVICE
loadmodule	U2R	ELEVATION OF PRIVILEGES
spy	R2L	INFORMATION DISCLOSURE
perl	U2R	ELEVATION OF PRIVILEGES

Tipos de Ataque

S/N	Name	Type
23.	Back	dos
24.	buffer_overflow	u2r
25.	ftp_write	r2l
26.	guess_passwd	r2l
27.	imap	r2l
28.	ipsweep	probe
29.	land	dos
30.	loadmodule	u2r
31.	multihop	r2l
32.	neptune	dos
33.	nmap	probe
34.	perl	u2r
35.	phf	r2l
36.	pod	dos
37.	portsweep	probe
38.	rootkit	u2r
39.	satan	probe
40.	smurf	dos
41.	spy	r2l
42.	teardrop	dos
43.	warezclient	r2l
44.	warezmaster	r2l

Características del Ataque

duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.

num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
serror_rate: continuous.
srv_serror_rate: continuous.
rerror_rate: continuous.
srv_rerror_rate: continuous.
same_srv_rate: continuous.
diff_srv_rate: continuous.
srv_diff_host_rate: continuous.
dst_host_count: continuous.
dst_host_srv_count: continuous.
dst_host_same_srv_rate: continuous.
dst_host_diff_srv_rate: continuous.
dst_host_same_src_port_rate: continuous.
dst_host_srv_diff_host_rate: continuous.
dst_host_serror_rate: continuous.
dst_host_srv_serror_rate: continuous.
dst_host_rerror_rate: continuous.
dst_host_srv_rerror_rate: continuous.

Registros para entrenamiento de Red Neuronal

Tipo	total real	porcentaje	total_900	total_realista_900	porcentaje	total_1500	total_realista_1500	porcentaje
1	67343	53.45828074	481.1245267	470	52.22222222	791.6666667	788	52.5333333
2	41214	32.7165345	294.4488105	285	31.66666667	483.3333333	479	31.9333333
3	890	0.706500599	6.358505394	6	0.666666667	10	10	0.66666667
4	3599	2.856961412	25.71265271	25	2.777777778	41.66666667	42	2.8
5	2931	2.326689052	20.94020147	20	2.222222222	33.33333333	33	2.2
6	892	0.708088241	6.37279417	6	0.666666667	10	10	0.66666667
7	1493	1.185174601	10.66657141	10	1.111111111	16.66666667	17	1.13333333
8	3633	2.883951323	25.95556191	25	2.777777778	41.66666667	42	2.8
9	2646	2.100450096	18.90405087	18	2	30	30	2
10	201	0.159558001	1.436022005	2	0.222222222	1.666666667	3	0.2
11	956	0.758892779	6.830035008	6	0.666666667	10	10	0.66666667
12	53	0.042072508	0.378652568	2	0.222222222	1.666666667	3	0.2
13	8	0.006350567	0.057155105	2	0.222222222	1.666666667	3	0.2
14	7	0.005556746	0.050010717	2	0.222222222	1.666666667	3	0.2
15	10	0.007938209	0.071443881	2	0.222222222	1.666666667	3	0.2
16	30	0.023814627	0.214331642	2	0.222222222	1.666666667	3	0.2
17	11	0.00873203	0.078588269	2	0.222222222	1.666666667	3	0.2
18	20	0.015876418	0.142887762	2	0.222222222	1.666666667	3	0.2
19	4	0.003175284	4	4	0.444444444	4	4	0.26666667
20	18	0.014288776	0.128598985	2	0.222222222	1.666666667	3	0.2
21	9	0.007144388	0.064299493	2	0.222222222	1.666666667	3	0.2
22	2	0.001587642	2	2	0.222222222	2	2	0.13333333
23	3	0.002381463	3	3	0.333333333	3	3	0.2
TOTAL	125973	100	908.935701	900	100	1500	1500	100

Registros para entrenamiento 70/30

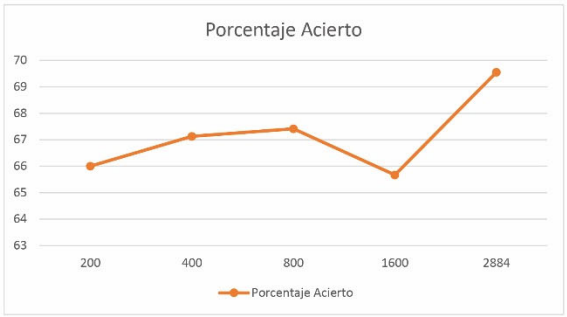
tipo	TOTAL 1500	70% 1500	REAL 70% 1500	30% 1500	REAL 30% 1500	TOTAL 900	70% 900	real 70%	30% 900	real 30%
1	788	551.6	552	236.4	236	470	329	331	141	139
2	479	335.3	336	143.7	143	285	199.5	201	85.5	84
3	10	7	7	3	3	6	4.2	4	1.8	2
4	42	29.4	30	12.6	12	25	17.5	18	7.5	7
5	33	23.1	23	9.9	10	20	14	14	6	6
6	10	7	7	3	3	6	4.2	4	1.8	2
7	17	11.9	12	5.1	5	10	7	7	3	3
8	42	29.4	29	12.6	13	25	17.5	18	7.5	7
9	30	21	21	9	9	18	12.6	13	5.4	5
10	3	2.1	2	0.9	1	2	1.4	1	0.6	1
11	10	7	7	3	3	6	4.2	4	1.8	2
12	3	2.1	2	0.9	1	2	1.4	1	0.6	1
13	3	2.1	2	0.9	1	2	1.4	1	0.6	1
14	3	2.1	2	0.9	1	2	1.4	1	0.6	1
15	3	2.1	2	0.9	1	2	1.4	1	0.6	1
16	3	2.1	2	0.9	1	2	1.4	1	0.6	1
17	3	2.1	2	0.9	1	2	1.4	1	0.6	1
18	3	2.1	2	0.9	1	2	1.4	1	0.6	1
19	4	2.8	3	1.2	1	4	2.8	3	1.2	1
20	3	2.1	2	0.9	1	2	1.4	1	0.6	1
21	3	2.1	2	0.9	1	2	1.4	1	0.6	1
22	2	1.4	1	0.6	1	2	1.4	1	0.6	1
23	3	2.1	2	0.9	1	3	2.1	2	0.9	1
	1500	1050	1050	450	450	900	630	630	270	270

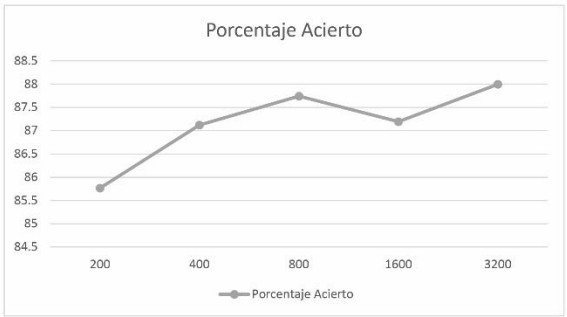
Resultados Obtenidos con diferente número de registros

300 Registros					
Iteraciones	200	400	800	1600	2884
Porcentaje Error	33.998888	32.8723632	32.5848855	34.3293789	30.4459767
Porcentaje Acierto	66.001112	67.1276368	67.4151145	65.6706211	69.5540233

900 Registros					
Iteraciones	200	400	800	1600	1955
Porcentaje Error	22.4214337	18.8362186	17.4664538	15.3929354	14.6889749
Porcentaje Acierto	77.5785663	81.1637814	82.5335462	84.6070646	85.3110251

1500 Registros						
Iteraciones	200	400	800	1600	3200	3960
Porcentaje Error	17.3010949	14.232817	12.8779833	12.257294	12.8043861	12.0029996
Porcentaje Acierto	82.6989051	85.767183	87.1220167	87.742706	87.1956139	87.9970004





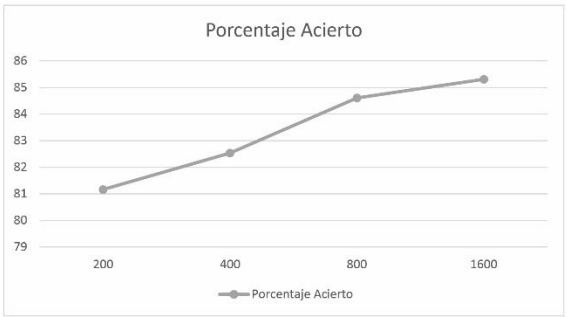
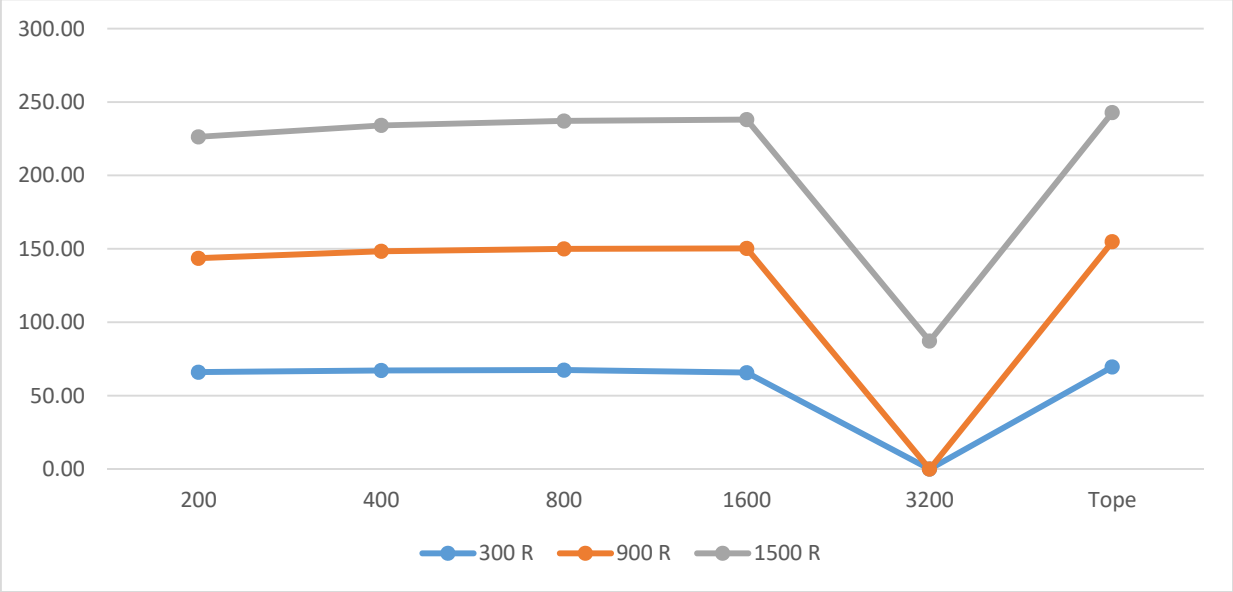


Tabla combinada de Resultados

300 Registros					
Iteraciones	200	400	800	1600	2884
Porcentaje Acierto	66.001112	67.1276368	67.4151145	65.6706211	69.5540233

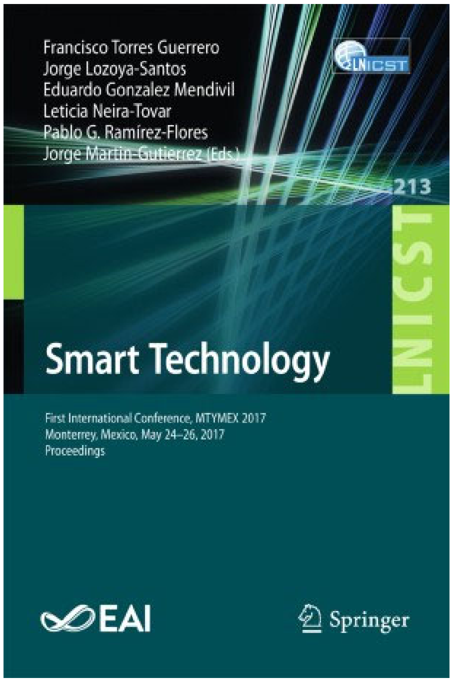
900 Registros					
Iteraciones	200	400	800	1600	1955
Porcentaje Acierto	77.5785663	81.1637814	82.5335462	84.6070646	85.3110251

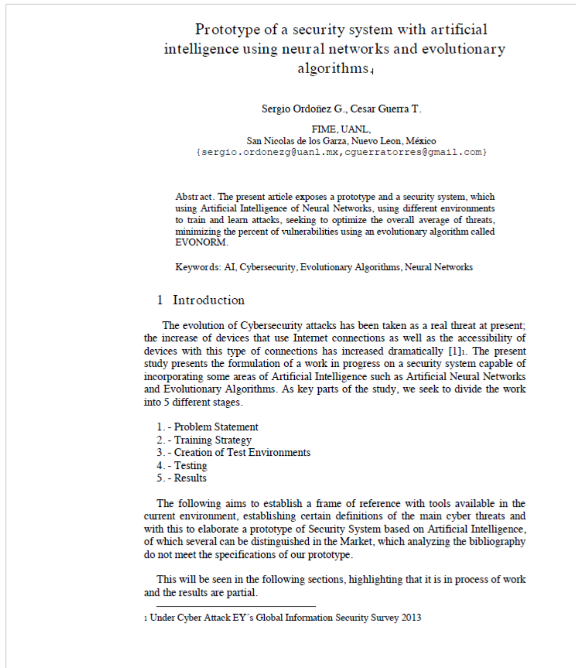
Porcentaje de Acierto						
Iteraciones	200	400	800	1600	3200	Tope
300 R	66.00	67.13	67.42	65.67	N/A	69.55
900 R	77.58	81.16	82.53	84.61	N/A	85.31
1500 R	82.70	85.77	87.12	87.74	87.20	88.00



Publicaciones elaboradas

First International Conference MTY/MEX 2017





Prototype of a security system with artificial intelligence using neural networks and evolutionary algorithms

Sergio Ordoñez G., Cesar Guerra T.

FIME, UANL,
San Nicolas de los Garza, Nuevo Leon, México
{sergio.ordonezg@uanl.mx, cguerratorres@gmail.com}

Abstract. The present article exposes a prototype and a security system, which using Artificial Intelligence of Neural Networks, using different environments to train and learn attacks, seeking to optimize the overall average of threats, minimizing the percent of vulnerabilities using an evolutionary algorithm called EVONORM.

Keywords: AI, Cybersecurity, Evolutionary Algorithms, Neural Networks

1 Introduction

The evolution of Cybersecurity attacks has been taken as a real threat at present; the increase of devices that use Internet connections as well as the accessibility of devices with this type of connections has increased dramatically [1][1]. The present study presents the formulation of a work in progress on a security system capable of incorporating some areas of Artificial Intelligence such as Artificial Neural Networks and Evolutionary Algorithms. As key parts of the study, we seek to divide the work into 5 different stages.

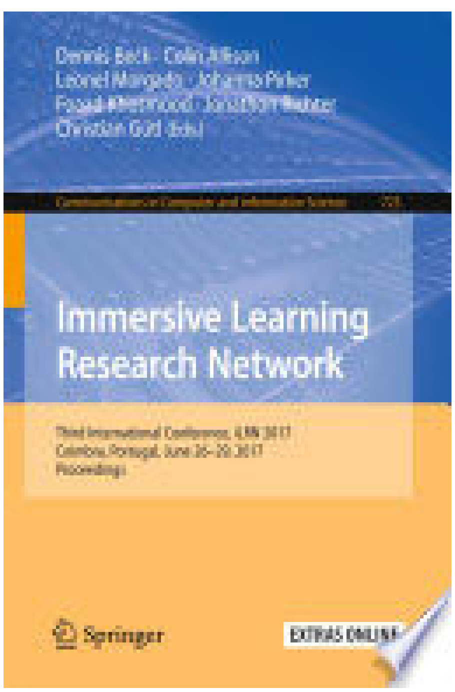
1. - Problem Statement
2. - Training Strategy
3. - Creation of Test Environments
4. - Testing
5. - Results

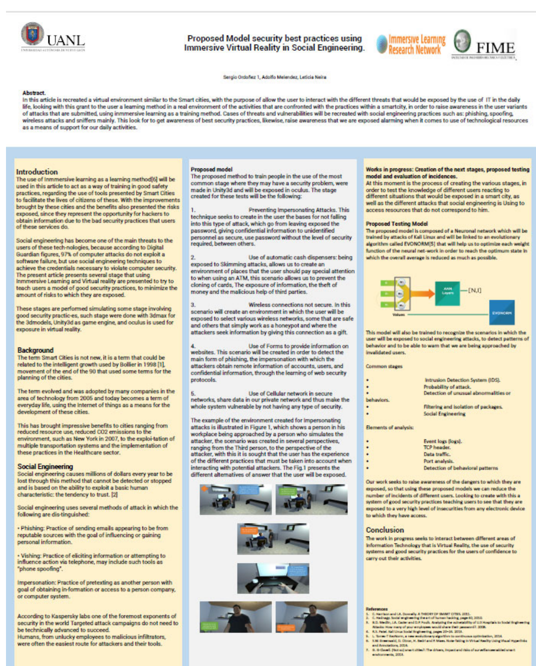
The following aims to establish a frame of reference with tools available in the current environment, establishing certain definitions of the main cyber threats and with this to elaborate a prototype of Security System based on Artificial Intelligence, of which several can be distinguished in the Market, which analyzing the bibliography do not meet the specifications of our prototype.

This will be seen in the following sections, highlighting that it is in process of work and the results are partial.

1 Under Cyber Attack EY´s Global Information Security Survey 2013

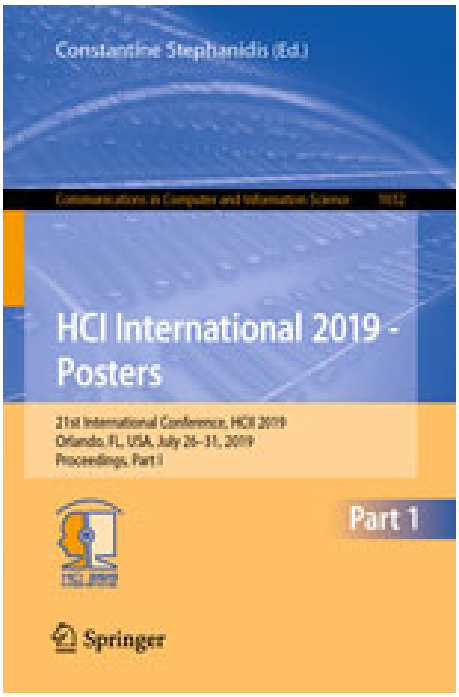
Immersive Learning Research Network 2017





HCII: International Conference on Human-Computer Interaction

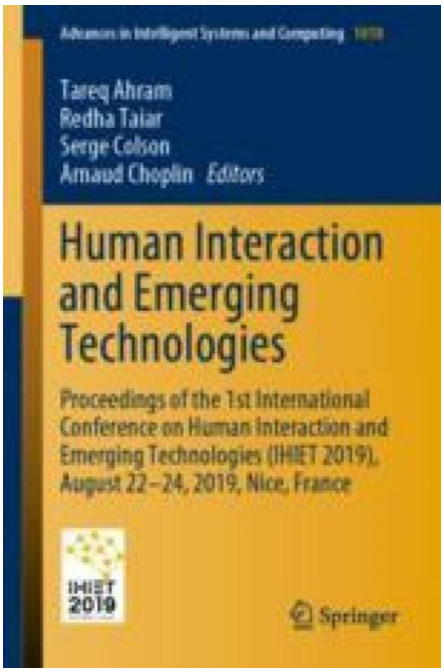
Título: How Tour Video Games Benefit Students: Study Case Freshman Engineering School



- Autores: Leticia Neira-Tovar
- Sergio Ordoñez
- Francisco Torres-Guerrero

IHIET: International Conference on Human Interaction and Emerging Technologies

Título: Security System Based on STRIDE Using Artificial Intelligence



Autores: Sergio Ordoñez, Juan Mendoza, Omar Gonzalez, Miguel Obregon, Joaquin Lara, Mexico