Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema híbrido (centralizado y descentralizado) de detección de intrusiones basado en inteligencia artificial.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Francisco Javier Fernández González

Tutor: Alberto Miguel Bonastre Pina

2019-2020

Prólogo

Agradecimiento:

Doy las gracias a todas las personas que me han ayudado en el camino que me ha permitido llegar al punto donde estoy hoy. A todos aquellos que me han apoyado cuando lo he necesitado, los que me han apoyado y han creído en mí, porque es por todos ellos que he podido desarrollar este proyecto.

Pero en especial, quiero agradecérselo a toda mi familia, a mis padres por guiarme siempre hacia el camino correcto y ayudarme a levantarme cada vez que me caía, a mi hermana pequeña por tratar de ser un referente para ella y dejarme que le enseñe que las cosas se consiguen con esfuerzo, a mi pareja por estar ahí siempre y darme fuerzas y apoyarme en todas las decisiones que he tomado.

También se lo dedico a mis amigos, que también han sufrido mi ausencia y aún así me han dado ánimos y energía para llegar hasta aquí.

No debo olvidar a todos los profesores que he tenido, tanto en el colegio como en la carrera, todos ellos me han hecho crecer y llegar hasta donde estoy ahora. Pero en especial al tutor de este proyecto, nunca se ha rendido conmigo y me ha apoyado hasta en su enfermedad.

Por todo esto y mucho más, muchas gracias a todos, de corazón.

Les dedico mi trabajo a todas estas personas y en especial a mi abuelo que no puede estar aquí para verlo.

Resumen

En este proyecto se propone la definición, especificación e implementación de un sistema de detección de intrusiones en redes. Para ello se seleccionará un conjunto de ataques de entre los más habituales. Para cada uno de ellos se analizarán los parámetros significativos del tráfico de las redes que puedan inducir a pensar que se está produciendo un ataque con este perfil. Tras el estudio teórico, se implementará un sistema de alarma híbrido centralizado/distribuido basado en técnicas de Inteligencia Artificial (IA). El sistema consiste en un conjunto de nodos detectores (sensores) que analizarán de forma pasiva el tráfico que transcurre por el segmento de la red que observan. Estos nodos dispondrán de capacidades de IA por lo que podrán, de forma autónoma, tomar decisiones en función del tráfico observado acerca de la necesidad de alertar a los administradores del sistema de una posible intrusión. El sistema se complementa con un elemento centralizado (monitor) que recibirá datos de los nodos sensores y buscará en ellos indicios de intrusión aplicando técnicas computacionalmente más complejas de Inteligencia Artificial. Este dispositivo dispondrá de una visión completa de la topología de la red y, por tanto, mayor capacidad de detección. Todo ello durante el funcionamiento habitual de la red... Por sencillez y coste, la comunicación entre sensores y monitor se realizará empleando la red monitorizada. Para minimizar la intrusión sobre la misma, este tráfico se adaptará al estado de carga de la red, decidiendo transmitir datos parametrizados – realizando un preproceso del tráfico observado para detectar los parámetros relevantes del mismo - o bien las trazas completas para que el nodo monitor pueda detectar intrusiones de una forma más detallada.

La ventaja más significativa del sistema de detección de intrusiones propuesto es la posibilidad de detectar una serie de ataques predefinidos anteriormente mediante un mecanismo híbrido centralizado/distribuido basado en técnicas de inteligencia artificial.

Palabras clave: sistema de detección de intrusiones; Inteligencia artificial; Redes de computadores; Análisis de redes; Carga de red; Ingeniería Informática; Deep Learning; Predicciones; Automatización; Detección; Ciberseguridad;

Abstract

This project proposes the definition, specification, and implementation of a network intrusion detection system. To do this, a set of attacks will be selected from among the most common. For each of them, the significant parameters of the traffic of the networks that may lead to think that an attack with this profile is taking place will be analyzed. After the theoretical study, a hybrid centralized / distributed alarm system based on Artificial Intelligence (AI) techniques will be implemented. The system consists of a set of detector nodes (sensors) that will passively analyze the traffic that passes through the network segment they observe. These nodes will have AI capabilities so they can autonomously make decisions based on observed traffic about the need to alert system administrators of a possible intrusion. The system is complemented with a centralized element (monitor) that will receive data from the sensor nodes and will search them for signs of intrusion by applying computationally more complex Artificial Intelligence techniques. This device will have a complete view of the network topology and, therefore, greater detection capacity. All this during the normal operation of the network ... For simplicity and cost, communication between sensors and monitor will be carried out using the monitored network. To minimize intrusion on it, this traffic will adapt to the load status of the network, deciding to transmit parameterized data - performing a pre-processing of the observed traffic to detect its relevant parameters - or the complete traces so that the monitor node can detect intrusions in a more detailed way.

The most significant advantage of the proposed intrusion detection system is the possibility of detecting a series of previously predefined attacks using a hybrid centralized / distributed mechanism based on artificial intelligence techniques.

Keywords: intrusion detection system; Artificial intelligence; Computer networks; Network analysis; Network load; Informatics Engineering; Deep Learning; Predictions; Automation; Detection; Cybersecurity.

Tabla de contenidos

Tabla de figuras

1. Capítulo 1

1.1. Introducción

Las nuevas tecnologías ya se han integrado en nuestra vida cotidiana, y se encuentran presentes tanto en hogares, como en hospitales, fábricas y demás aspectos del día a día. Es por ello por lo que puede considerase imprescindible garantizar su correcto funcionamiento. Uno de los factores que amenazan este correcto funcionamiento son los ataques informáticos. Por tanto, es necesario el desarrollo e implementación de nuevos dispositivos que tengan la capacidad de identificar intentos de ataques de la forma más ágil, segura y eficiente posible.

Dado el crecimiento exponencial que están teniendo los sistemas y la facilidad de acceso a la información, no es posible evitar completamente la aparición de una intrusión en nuestro sistema. Este TFG pretende obtener una novedosa aproximación a la implementación de un sistema completo de detección de amenazas.

Un IDS [1] (Intrusion Detection System, o Sistema de detección de intrusiones) es un conjunto de recursos hardware o software que se incorporan a una red informática, doméstica, corporativa o troncal) con el propósito de observar el tráfico que circula por ella y detectar posibles amenazas o ataques informáticos que atenten contra la seguridad. El objetivo principal de este trabajo consiste en implementar un IDS novedoso, como se indicará más adelante.

1.2. Motivación

Desde un punto de vista personal la temática del TFG ha sido elegida por su capacidad innovadora y la cantidad de aprendizaje que me iba a aportar. Desde el momento de iniciar la carrera ya tenía claro que quería centrar mi aprendizaje en IoT o en ciberseguridad. Entre mis alternativas para desarrollo de este proyecto, tenía dos opciones planteadas, continuar con mi proyecto de emprendimiento personal, el desarrollo de un sistema de domotización de viviendas para personas de la tercera edad, incluyendo detección de patrones y predicción de comportamientos, o el que se presenta en este documento. Decantarme por uno no implicaba dejar el otro de lado. Por tanto, me he centrado en el desarrollo de éste y cuando finalice la parte entregable, trabajaré centrándome en el otro con el objetivo de buscar un producto comercial.

Respecto a la motivación profesional, tras haber realizado ya unas prácticas en el grupo Eleven-Paths de Telefónica y con la posibilidad de entrar a trabajar con ellos definitivamente con la carrera finalizada, es más interesante la realización del TFG que he elegido, ya que me ha aportado muchos más conocimientos en el ámbito de la ciberseguridad que el de domotización que era únicamente IoT.

La decisión de centrarme en el sistema de detección de intrusiones ha sido debido a que me aportaría un aprendizaje mayor y la cantidad de documentación que se puede encontrar es mucho más reducida. Por tanto, el apoyo del tutor es lo que me iba a permitir llevar a cabo el desarrollo de este sistema, el facilitarme documentación en la que basarme para desarrollarlo y enseñarme el funcionamiento de técnicas de Deep Learning [2], ya que mi conocimiento al inicio del proyecto sobre este aspecto es completamente nulo.

También es importante mencionar la mejora tecnológica que implica este TFG para ello mencionamos las diferencias que encontramos sobre lo que hay en el mercado actualmente y que hacen de nuestro producto superior a la competencia. Esto no implica que las grandes corporaciones no cuenten con dispositivos construidos ad hoc similares a este. Estos aspectos los veremos comentados en profundidad en siguientes capítulos.

- Adaptabilidad al entorno en el que sea configurado.
- Mayor precisión que los elementos ya funcionales.
- Adaptabilidad en tiempo real a la carga de la red.
- Capacidad de funcionamiento de modo distribuido.
- Alta escalabilidad.
- Fácil modificación y reimplementación.

Otro aspecto fundamental para la elección es desde la perspectiva de un punto de vista social. El factor determinante ha sido la finalidad del trabajo. Encontramos que cada vez tenemos más dispositivos inteligentes en los hogares y empresas y no somos conscientes de los datos que están transmitiendo sobre nuestra red y todos los elementos que encontramos en el mercado para detectar intrusiones no son accesibles para el usuario medio. Por tanto, la posibilidad de diseñar un sistema para todos los públicos capaz de incrementar la seguridad tecnológica del hogar es un producto fundamental con el que ya deberíamos contar y que lamentablemente en estos momentos no tenemos disponible.

1.3. Objetivos

El objetivo principal consiste en abordar la implementación de un sistema de detección de intrusiones híbrido con técnicas de inteligencia artificial basado en Deep Learning .

Para ello son necesarios objetivos secundarios:

- Objetivo 1: Análisis detallado del problema y la solución.
- Objetivo 2: Caracterización de los ataques a detectar.
- Objetivo 3: Diseño e implementación de los monitores.
- Objetivo 4: Verificación del funcionamiento del sistema conjunto.

La secuencia de pasos a seguir es la siguiente:

Objetivo 1:

- Caracterizar el problema, fijando especificaciones. Delimitar los escenarios de uso de nuestra solución.
- Estudiar otras soluciones existentes a este problema.
- Identificar y estudiar las tecnologías a emplear.

Objetivo 2:

- Seleccionar los ataques a analizar en función del escenario de uso.
- Fijar los parámetros relevantes que permitirán identificar cada ataque.
- Generar un banco de pruebas que permita obtener capturas de los ataques a identificar.
- Obtener diferentes capturas de cada uno de los ataques.

Objetivo 3:

- Creación y entrenamiento de los módulos de IA para la detección de ataques.
- Definir el protocolo de comunicaciones entre Monitores.
- Diseñar e implementar los Monitores Locales.
- Diseñar e implementar del Monitor Global.
- Incorporar los monitores en el banco de pruebas

Objetivo 4:

- Verificar el correcto funcionamiento con variaciones de los ataques ya considerados.

Una vez definidos los objetivos del proyecto pasamos a desglosarlos para explicar lo que se pretende conseguir con más exactitud.

Para resolver el primer objetivo, primeramente, analizamos el problema en detalle y propusimos una serie de soluciones posibles. Vimos los posibles casos en los cuales este iba a poder ser utilizado y las principales diferencias entre estos escenarios. Hecho esto, observamos cuáles eran los dispositivos que había en el mercado en este momento y de que maneras se enfrentaban al problema que habíamos planteado, de manera que los caracterizamos y valoramos para evaluar las ventajas que tendría nuestro sistema y los mínimos que debíamos cubrir para ser un sistema funcional.

El siguiente paso fue ver de qué manera podíamos construirlo, cuáles son las tecnologías que debemos usar para resolverlo y de qué manera debemos usarlas para cumplir los requisitos del sistema.

En cuanto a la resolución del segundo objetivo, debemos estudiar cuales son, dependiendo de los escenarios, los ataques principales que debemos cubrir ya que sería

inviable para un trabajo fin de grado generar un sistema que cubriera el 100% de los ataques. Una vez tuvimos los ataques seleccionados, lo siguiente fue seleccionar la tecnología más adecuada para detectarlos, y una vez elegidas la técnica, cuál era la mejor manera de alimentar a nuestro sistema para hacerlo capaz de detectarlos mediante Deep Learning.

Para poder entrenar a nuestro sistema generamos un entorno desde el cual pudiéramos obtener capturas de todos los ataques que seleccionamos de manera segura. Sobre el mismo, ejecutamos un sniffer de red y obtuvimos los datos que necesitábamos y las guardamos para posteriormente entrenar al modelo de inteligencia artificial.

Para resolver el tercer objetivo, lo primero fue preparar el entorno de desarrollo para poder trabajar con un sistema de Deep Learning, Una vez tuvimos el sistema comenzamos con la creación del sistema bajo los parámetros que habíamos decidido previamente.

Creado el modelo de inteligencia artificial, el siguiente paso ha sido entrenarlo. Para ello usamos todo el conjunto de capturas de ataques que habíamos obtenido al cumplir el objetivo anterior.

Teniendo disponible el modelo pudimos definir de qué manera debían comunicarse los monitores de nuestro sistema, tanto la manera de identificar al monitor global como la forma en que deben enviarse los mensajes para que sea éste quien ejecute los algoritmos de IA más pesados.

Tras las comunicaciones, implementamos ambos monitores, tanto el local como el global, cumpliendo las especificaciones de diseño y siguiendo el patrón de comunicaciones que acabábamos de definir.

Incorporamos y configuramos los monitores en nuestro sistema de pruebas que habíamos creado en el objetivo dos, de manera que pudieran actuar de forma autónomo como si se tratara de un caso de uso real.

Por último, pasamos a la fase de pruebas y verificación del sistema, generamos todos los ataques que habíamos identificado previamente y comprobamos si nuestro sistema es capaz de detectarlos y notificarlos de la forma que se había previsto.

1.4. Impacto esperado

El impacto esperado de este proyecto debe ser analizado desde diferentes perspectivas.

Como todo proyecto de ciberseguridad, el objetivo principal es la consecución de que todos los entornos, ya sean industriales o familiares, sean más seguros y vulnerables a una menor cantidad de amenazas. Para lograr esto, lo primero y fundamental será la creación de un sistema autónomo e independiente capaz de detectar los ataques e

intrusiones más frecuentes. Además, el impacto fundamental que se desea obtener en este trabajo es la posibilidad de que, en los hogares del ciudadano medio, al igual que instalan antivirus en sus equipos, puedan tener un dispositivo dentro de sus redes que les permita detectar intrusiones en sus redes y el robo de datos o el uso de sus dispositivos personales como “Bot nets”.

En este momento ya existen herramientas que realizan esta función, aunque habitualmente en ámbitos profesionales o empresariales y a precios elevados. Este trabajo busca dar acceso a una herramienta diferente, un sistema hardware que sea capaz de incrementar la seguridad en redes, de manera asequible y disponible para todos los hogares.

El otro punto fundamental, es el cambio de paradigma en cuanto al desarrollo de IDSs. Consiste en dejar de trabajar con reglas y pasar a emplear Deep Learning, que nos proporciona unos resultados mucho más precisos, detectando una mayor cantidad de ataques dependiendo del entrenamiento que reciba el modelo.

En la parte relacionada con las ODS (objetivos de desarrollo sostenible) [3], podemos centrarnos en el apartado del punto 8 “Trabajo decente y crecimiento económico”, concretamente en el apartado 8.2 “Modernización de la tecnología”. Mediante este sistema conseguimos una mayor adaptabilidad sin aumentar los costes, además de unos resultados mucho más precisos, lo que finalmente termina en un crecimiento económico. Además del punto 9 “Industria, innovación e infraestructuras”, que corresponde con el fundamento principal del proyecto: generar un sistema innovador que realice una serie de funciones de una manera más eficiente de lo que se están haciendo actualmente.

Si analizamos el impacto desde los diferentes usuarios que van a emplear el sistema podemos identificar dos grandes bloques. Por un lado, los administradores de estas redes, los cuales van a poder realizar su trabajo de una forma más rápida y eficiente y por otro lado los usuarios de estas redes, que sin perder rendimiento van a estar mucho más y mejor protegidos ante cualquier amenaza.

Si bien las capacidades del sistema presentado en este TFG son limitadas, esta línea de trabajo permitirá – y es la intención del autor – el desarrollo de un nuevo IDS mucho más completo, que identifique un rango mayor de ataques con una baja sobrecarga del sistema, basado en técnicas de Deep Learning. La utilización de estas tecnologías punteras permitirá la evolución de este sistema para adaptarse a nuevas amenazas de forma sencilla.

Finalmente, se espera que se pueda terminar desarrollando un producto comercial, que pueda ser comprado directamente por usuarios particulares o por administradores de redes, y que permita aumentar en gran medida el nivel de seguridad de la red por un precio asequible, contando con unas técnicas de predicción altamente eficientes bajo el modelo de Deep Learning.

1.5. Metodología

La metodología empleada para la realización de este proyecto ha sido orientada para lograr con éxito todos y cada uno de los objetivos planteados.

El primer paso ha sido estudiar las posibles soluciones disponibles frente al problema de inseguridad de la mayor parte de las redes actuales. De este estudio, tras mucho pensar, explorar y probar algunas de las soluciones que ya existen, se ha constatado la necesidad del sistema objeto de este TFG.

En segundo lugar, y puesto que los ámbitos de aplicación de las redes de computadores son muy diferentes, se ha estudiado en qué situaciones es aplicable el IDS propuesto, teniendo en cuenta en su diseño la necesidad de dotarlo de una capacidad de adaptación sencilla a la mayor cantidad posible de entornos y situaciones.

Una vez definidas las características del IDS a implementar, focalizamos en las herramientas que disponemos para realizarlo, analizando diversas tecnologías que tenemos a nuestro alcance.

El siguiente paso es la identificación de los diferentes perfiles de ataque a detectar, y la caracterización de los mismos, teniendo en cuenta su gravedad y su incidencia en la actualidad.

Tras estos análisis, con toda la documentación preparada y reunida, se procede a construir los monitores locales y el monitor global, incluyendo sus comunicaciones. Teniendo en cuenta que se trata de una aproximación, construiremos todo el entorno mediante máquinas virtuales.

El siguiente paso consiste en implementar los ataques para conseguir recrear sus perfiles y obtener un conjunto de trazas de cada ataque que permita entrenar a la herramienta de IA basada en Deep Learning.

A continuación, se entrenará la IA mediante estas trazas para crear un módulo que, frente a una captura real, sea capaz de identificar los ataques y activar las alertas oportunas.

Por último, se incluirán los módulos de IA en los nodos monitores (ML y MG) dentro de un entorno de pruebas para verificar las capacidades de detección del sistema frente a los ataques considerados y variaciones de los mismos.

1.6. Estructura

Durante las siguientes páginas entraremos en detalle sobre todos los aspectos que hemos mencionado anteriormente. Comenzaremos por definir el contexto tecnológico y

observar que elementos tenemos en el mercado. Continuaremos comentando qué aspectos deberíamos mejorar y con qué objetivo. Lo siguiente será la explicación detallada de la propuesta de valor, en qué nos centramos y por qué. Hecho esto, realizaremos un análisis del problema que esperamos solucionar, teniendo en cuenta diferentes ámbitos, como pueda ser la seguridad, el marco legal, la protección de datos y los riesgos inherentes al proyecto. El siguiente paso consta de un plan de trabajo detallado con las estimaciones de esfuerzo del mismo. Proseguiremos con el presupuesto del sistema, aunque de una forma un poco abierta ya que depende en gran medida del entorno en el que vaya a ser construida la herramienta.

Visto todo esto, pasamos al punto central de la solución, es decir, su diseño, con el estudio de las tecnologías apropiadas y el desarrollo e implementación del mismo.

Esta memoria finaliza considerando las conclusiones del proyecto, la relación del mismo con los estudios cursados, las perspectivas de futuro del proyecto y finalmente las referencias.

Al final del documento incluiremos como anexo un glosario de los términos empleados, con documentación y explicaciones adicionales para contextualizar más aun el trabajo.

2. Capítulo 2

2.1. Estado del arte

2.1.1. Herramientas IDS

En este punto vamos a poner atención al contexto tecnológico que encontramos en este momento. Lo que más abunda en este momento son soluciones software que nos permitan, instar sensores en nuestras máquinas o leer los datos que estamos utilizando a nivel individual, buscando obtener la mayor parte de los datos a analizar de los equipos de los usuarios mostrándolos posteriormente en un monitor central. Debido a que encontramos en el mercado gran cantidad de productos similares vamos a analizar los más utilizados y aquellos cuyos rendimientos están por encima del resto, no obstante, en la siguiente tabla podremos encontrar un listado de los más utilizados actualmente con una breve descripción.

Snort	Es una herramienta de barrido de puertos y análisis de paquetes cuyo objetivo es responder ante anomalías de la red.
Suricata	Es un motor de red multiplataforma de alto rendimiento que cuenta con una serie de herramientas que le permiten actuar tanto como un IDS como un IPS
Bro	Es un sistema de detección de intrusiones para UNIX cuyo funcionamiento se basa en analizar todo el tráfico de la red en busca de actividad poco habitual. Lo que hace que sea diferente al resto es su forma de detección, ya que sus reglas están basadas en scripts en lenguaje nativo que se encargan de detectar, generar logs o eventos y acciones a nivel de sistema operativo.
Kismet	Es un sniffer y un sistema de detección de intrusiones que funciona únicamente con redes inalámbricas 802.11.

OSSEC	Son un conjunto de herramientas de intrusos que pretenden desde los registros de eventos del sistema operativo, comprobar la integridad del mismo, auditar los registros de los equipos Windows, detección de rootkits, alerta en tiempo real y respuesta activa a ataques.
Tripwire	Son un conjunto de herramientas que pretenden detectar intrusiones en un sistema desde diferentes ámbitos, tanto los registros de los equipos como los paquetes de red.
Samhain	Es un sistema que proporciona verificación de integridad de archivos y monitorización de archivos de registro, detección de rootkits, barrido de puertos, detección de ejecutables no autorizados y procesos ocultos.
NGIPS	Es un sistema, de la mano de CISCO, que cuenta con un conjunto de herramientas que pretenden detectar intrusiones basado en un sistema de redes de reglas altamente actualizado.

Tabla 1 Descripción de productos en el mercado

El primero y tal vez el más antiguo es Snort [4], una herramienta openSource de análisis de paquetes. De hecho, es un paquete fundamental a integrar en otros tipos de sistemas para identificar gran cantidad de ataques y vulnerabilidades. No cuenta con entorno gráfico y debe estar cargada en cada uno de los equipos de la red para asegurar un éxito frente a todos los ataques.

Este sistema cuenta con una serie de componentes que le permiten trabajar y cumplir las necesidades para las que ha sido diseñado. Su primer componente es un decodificador de paquetes y preprocesador de los mismos, que toma el paquete de todos los interfaces de red que tiene disponibles y con esto traduce el paquete y lo transforma en la base para la que se le han creado sus reglas de detección, además, durante este preprocesado ya analiza las cabeceras de los paquetes y en caso de detectar alguna anomalía lo notifica. El siguiente elemento es un motor de detección, que se basa en comparar este paquete preprocesado que ha conseguido con los listados con los que cuenta en las redes de reglas. Por último, cuenta con un sistema de Logging que le permite mostrar las alertas necesarias según la detección que haya logrado.

Una gran ventaja de esta herramienta es que cuenta con una gran cantidad de plugins de salida que le permiten adaptarse a gran cantidad de sistemas para almacenar estas alertas como pueda ser MYSQL o Rsyslog.

Snort puede funcionar de tres maneras diferentes, la primera y principal, como sniffer, en este modo monitorizará en tiempo real todo el tráfico que pase por la red y toda la actividad para la cual este haya sido preparado para monitorizar. El segundo modo es como packet logger, cuando funciona de esta manera únicamente almacena toda

la información que puede recibir para posteriormente analizarla. Por último, el modo IDS, que puede monitorizar o bien por pantalla o bien en un sistema basado en log y sobre toda la información que ha recaudado aplicar las reglas de detección cargadas previamente para lograr que detecte una serie de patrones haciendo matching con estas.

El siguiente que compararemos será Suricata [5], otra herramienta openSource robusta de detección de intrusiones. Es un sistema centralizado al igual que Snort, que además puede ejecutarse sobre el mencionado anteriormente con unas técnicas y mejoras de eficiencia destacables.

En cuanto a las características principales de este sistema podemos hablar del procesamiento multihilo, lo que le hace destacar sobre las demás ya que cuenta con la capacidad de ejecutar varios procesos de forma simultánea. Esto hace que aumente en gran medida su rendimiento porque puede capturar y analizar 4 paquetes simultáneamente, decodificarlos y reensamblarlos para que coincidan con las bases de sus redes de reglas. Comparar estos paquetes mediante pattern matching con las reglas especificadas y procesar los eventos y mostrar las salidas de las alertas.

La siguiente característica a destacar es su detección de protocolo automática que le da acceso a analizar protocolos que los IDS habitualmente no analizan como puedan ser TLS o FTP. Esto es gracias a que cuenta con claves para detectarlos todos y trabajar sobre cada uno de ellos.

Suricata también cuenta con una gestión de estadísticas personalizadas que permite en tiempo real o mediante almacenamiento tener acceso a los datos y análisis de rendimiento.

La última apuesta de valor de Suricata es un log de HTTP, que independientemente de que sean caracterizadas como elementos sospechosos son almacenadas para poder analizarlas posteriormente y en caso de aparecer un problema poder revisar la procedencia del mismo.

Vistas estas herramientas pasamos a una plataforma concreta como pueda ser OSSEC [6], la cual se basa en el análisis de forma distribuida en todos los dispositivos de la red, en este caso llamados agentes, los cuales se encargan de procesar los datos y enviarlos a un servidor. El modelo de detección es completamente diferente, funciona como una plataforma completa, no como un sistema independiente basado en la supervisión del Host.

El motor de análisis está preparado para correlacionar y analizar registros de dispositivos y formatos diferentes, entre los principales podemos destacar sistemas basados en Unix, servidores FTP o Servidores de correo entre muchos otros. Esto aporta a la plataforma una versatilidad muy grande.

Además, ofrece la posibilidad de una gestión centralizada de políticas en varios sistemas operativos. Cuenta con centralización de servicio de registro de eventos mediante Rsyslog, mecanismos de respuesta activa con una herramienta llamada fail2ban, sistema de monitorización de ficheros, detectores de kits de intrusiones y sistemas de alertas y análisis.

El apartado fundamental de este sistema es su capacidad integradora de muchas otras herramientas de código abierto que le permite cumplir los objetivos de detección en muchos tipos de sistemas.

OSSEC es una plataforma completa que requiere de una ardua configuración y un alto nivel de conocimientos para ser utilizada que integra una gran cantidad de sistemas y permite realizar todos tipos de análisis basados en el host empleando redes de reglas y cuenta con un completo sistema de notificaciones y alertas.

Podríamos continuar con análisis de todos y cada uno de los sistemas existentes y continuaríamos viendo las mejoras sustanciales del sistema propuesto, no obstante, además de lo ya visto anteriormente en las comparaciones que hemos presentado, nuestro sistema cuenta con un factor más diferencial y es la capacidad de actuar sin ningún tipo de configuración ni supervisión.

Una vez contextualizado el nivel tecnológico de los sistemas que existen en este momento podemos pasar a un punto más profundo en este aspecto. Actualmente, por suerte o por desgracia, es muy sencillo encontrar tutoriales en internet para, de manera sencilla poner en riesgo la integridad de una red. Esto se ilustra muy claramente en la cita de *Gene Spafford:* "El único sistema completamente seguro es aquel que está apagado, encerrado en un bloque de cemento y sellado en una habitación rodeada de alambradas y guardias armados [7]". Ya que evidentemente todo Software que esté conectado se puede piratear, debemos añadir un esfuerzo extra en la detección de esos ataques para poder proporcionar una respuesta adecuada en el mínimo tiempo posible.

Además, no debemos perder de vista el factor más importante en la inseguridad de todo sistema, este es el factor humano y error que tiene asociado, ya que hemos llegado a un punto donde el eslabón más débil de una red son los usuarios de la misma. Esto nos lo ilustra *Kevin Mitnick: "Las empresas invierten millones en firewalls, cifrado y dispositivos para acceder de forma segura, y es dinero malgastado, porque ninguna de estas medidas corrige el eslabón más débil de la cadena de seguridad: la gente que usa y administra los ordenadores".*

Esto nos lleva al siguiente punto y nos da que pensar sobre la inversión que se está produciendo en estos momentos en muchos países en el ámbito de la ciberseguridad, ya que lamentablemente, como dice *Richard Clarke: "Si gastas más en café que en seguridad de TI, serás pirateado. Además, mereces ser pirateado."* Aunque no queramos desear el mal a nadie el contexto actual no podemos negar que es este y que, aunque se endurezca la política de protección de datos o se establezcan delitos digitales aún estamos muy lejos de una política de ciberseguridad universal.

2.1.2. Inteligencia Artificial y Redes Neuronales

Es sabido que simular un comportamiento inteligente en un sistema informático es algo posible y que cada vez se consigue un comportamiento más similar al de un cerebro humano real con la finalidad de reproducir su inteligencia.

Como una definición más detallada podríamos decir que la inteligencia artificial es el diseño de una serie de elementos que cuando los ejecutamos sobre una arquitectura hardware nos proporcionan unos resultados similares al comportamiento de la inteligencia humana.

Debemos tener en cuenta, que para lograr esta inteligencia artificial debemos aunar varias áreas multidisciplinares, como son la informática, la computación, las matemáticas, la lógica y la filosofía, y hacer que todas ellas trabajen en un único sistema capaz de resolver problemas.

Los inicios de la inteligencia artificial basado en Redes neuronales se remontan a los años 40 del siglo pasado, cuando McCulloch y Pitts [8] en 1943 realizan el primer modelo matemático de una neurona y 15 años más tarde Rosenblatt [9] implementa el modelo de perceptrón, el cual implica las bases de lo que conocemos actualmente como Redes neuronales.

En la ilustración 2 podemos observar la estructura de dicho modelo:

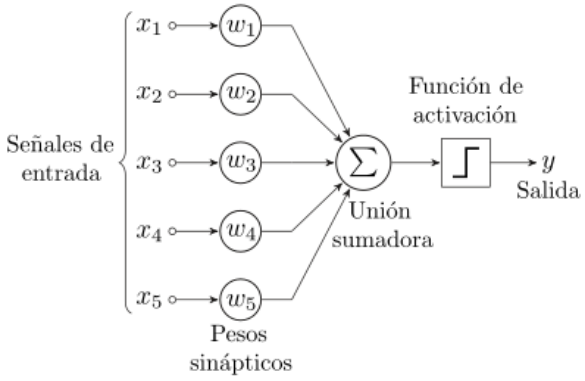

Ilustración 1 Funcionamiento de Perceptrón

El funcionamiento del modelo es el siguiente: Cada neurona cuenta con una serie de entradas. Puesto que cada entrada tiene una importancia para la salida final se le asigna a cada una de ellas un peso sináptico que le adapta su valor. Posteriormente suma estos productos y los compara con el valor umbral esperado por la función de activación.

La siguiente etapa interesante fue al final de la década de los 90, cuando se produjo un aumento significativo de la capacidad computacional de los computadores que permitió la aparición de gran cantidad de datos que podíamos comenzar a analizar, de forma que comenzaron a estudiarse algoritmos que permitiesen analizar y trabajar estos datos. Fue finalmente en 1997 cuando un ordenador de IBM, Deep Blue [10], derrotó al campeón de ajedrez Kaspárov en una partida.

Fue posteriormente con el aumento de uso de las redes y la difusión de internet cuando obtuvimos nuevamente una gran cantidad de datos que podíamos explotar, de hecho, grandes empresas como Apple comenzaron a comprar Start Ups que le

permitieron desarrollar elementos con inteligencias artificiales muy potentes como podría ser Siri.

Actualmente podemos identificar tres grandes bloques en los sistemas de inteligencia artificial. El primero de ellos son los sistemas que piensan como seres humanos, estos, imitan la forma de pensar, un ejemplo de esto son las Redes neuronales que imitan el funcionamiento del sistema nervioso humano. El siguiente bloque son los sistemas que actúan de forma racional, estos se encargan de reconocer el entorno y actuar según lo que han observado imitando el comportamiento de una persona, el ejemplo más claro que podemos encontrar son los robots. Por último, los sistemas que piensan de modo racional, estos son capaces de imitar la lógica de los humanos, estos son empleados en mejorar la calidad de ciertos servicios.

También debemos tener en cuenta que existen una serie de divisiones dentro del ámbito de la inteligencia artificial. El modelo convencional está basado únicamente en un análisis estadístico y formal del comportamiento. Éste emplea un razonamiento basado en casos, es decir en función de los resultados a ciertos problemas concretos es capaz de tomar decisiones más complejas, de modo que son capaces de actuar como sistemas expertos. El modelo computacional pretende el aprendizaje interactivo, para ello, pueden ser empelados elementos como la máquina de vectores de soporte [11], que permite encontrar patrones de forma eficiente, las Redes neuronales [12], los modelos de probabilidad de Markov [13], que pretende basar el aprendizaje en eventos probabilísticos, los sistemas difusos que buscan lograr razonamientos bajo incertidumbre y la computación evolutiva [14] que aplica conceptos inspirados en la biología.

Una vez contextualizado todo el avance de la inteligencia artificial vamos a centrarnos en un área concreta de la misma, en concreto la que ha sido utilizada para el desarrollo de nuestro sistema, las Redes neuronales, esta rama pretende construir programas que generen conocimiento sin necesidad de la supervisión humana.

Este tipo de inteligencia pretende imitar el modelo de aprendizaje que seguimos los humanos y para ello han sido desarrolladas gran cantidad de plataformas de la mano de las principales empresas tecnológicas que nos facilitan el trabajo con la misma.

El siguiente paso es la clasificación de estos sistemas. Para ello podemos definir dos grupos fundamentales, uno según la estructura de la red y otro según el tipo de aprendizaje.

Dentro del análisis según la topología debemos comprobar la forma de agrupación de las neuronas utilizadas, pero principalmente podemos definir las redes monocapa y las redes multicapa. Las redes monocapa son las más sencillas de todas, e incluso pueden implementarse por hardware. En estas redes las neuronas se interconectan todas entre sí y existe la posibilidad de que conexiones sobre si mismas (conexiones autorecurrentes), un ejemplo de ellas es la maquina estocástica de Boltzmann [33]. Las redes multicapa están formadas por distintas capas de neuronas, normalmente ordenadas en función de la manera en la que reciben la señal de entrada hasta que se genera la de salida, dentro de estas encontramos las redes con conexiones hacia adelante, esto implica que una capa solamente va a tener conexiones con las

siguientes en la dinámica de la computación y las redes con conexiones hacia atrás, que cuentan con una estructura inversa.

Si nos centramos en el análisis en función del tipo de aprendizaje, primeramente, debemos entender que es lo que entendemos como proceso de aprendizaje, este, es lo que entendemos como el proceso de presentación de los patrones a aprender a la red, y el ajuste de los pesos de las conexiones usando una regla de aprendizaje. Esta regla se basa en fórmulas matemáticas que a su vez emplean técnicas como minimización de errores u optimizaciones basadas en funciones de energía. El proceso consiste en, de forma iterativa, pasarle a la red una serie de patrones de muestra y este se encarga de ajustar los pesos hasta que logra la convergencia y es capaz de representarlos de una forma suficientemente similar al resultado esperado. En cuanto a este aprendizaje, hay 3 maneras de hacerlo interactuar con la red, mediante aprendizaje supervisado [15], aprendizaje no supervisado [16] y aprendizaje reforzado [17].

En las siguientes ilustraciones encontraremos un esquema de cada uno de estos tipos de aprendizaje en las cuales podremos observar cuál es su funcionamiento exacto:

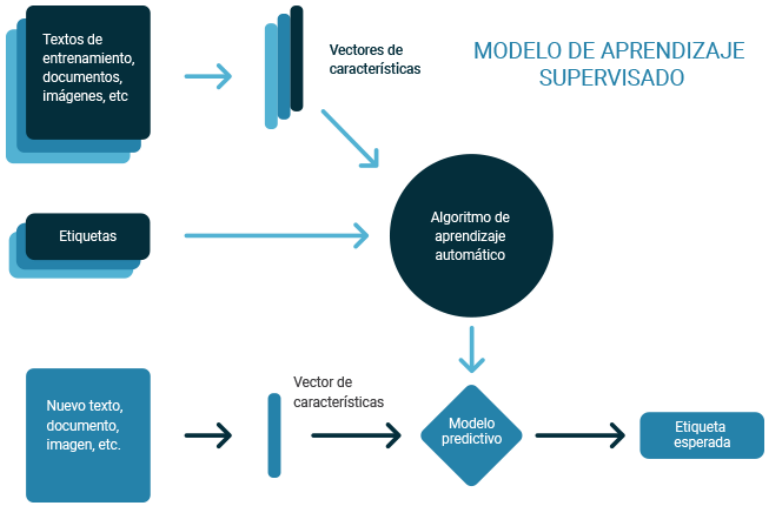

Ilustración 2 Aprendizaje supervisado

Tal y como vemos en la ilustración 2 el procedimiento del Aprendizaje supervisado está basado en una serie de algoritmos que trabajan con datos etiquetados con las respuestas que debe obtener la inteligencia artificial, de forma que debe encontrarse la función para la cual los datos de entrada obtengan como la etiqueta como función de salida. El algoritmo es entrenado con un conjunto de datos y así aprende a asignar la etiqueta de salida adecuada a un nuevo valor, es decir, es capaz de predecir el valor de salida.

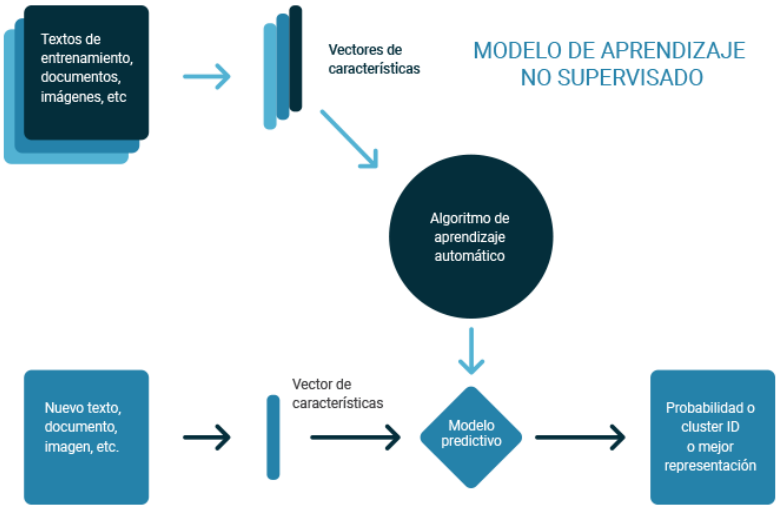

Ilustración 3 Aprendizaje no supervisado

En la ilustración 3 observamos que, para realizar un Aprendizaje no supervisado, este lo encontramos cuando no tenemos los datos etiquetados, por tanto, se pretende explorar los mismos para buscar una serie de patrones que simplifiquen el análisis. De manera que buscamos representar los datos que tenemos de una forma más estructurada.

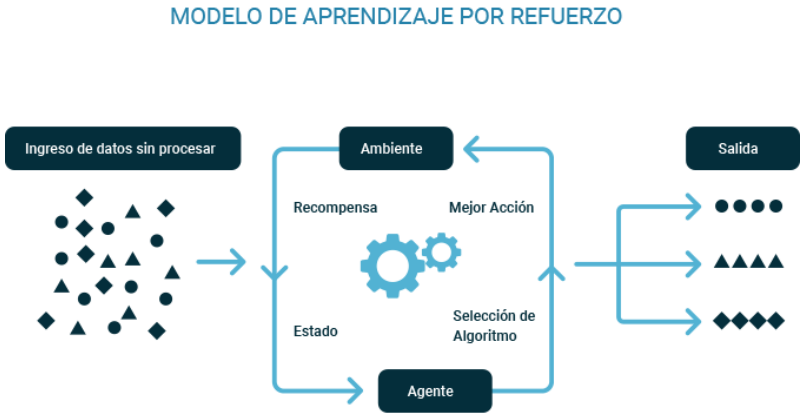

Ilustración 4 Aprendizaje por refuerzo

Por último, en la ilustración 4 encontramos el aprendizaje por refuerzo que pretende mejorar la respuesta del proceso de retroalimentación. El algoritmo aprende de su entorno y su información de entrada es la retroalimentación que obtiene como respuesta de sus acciones. De forma que su aprendizaje esta principalmente basado en prueba y error.

En concreto para nuestro desarrollo vamos a emplear un tipo concreto de red neuronal denominada aprendizaje profundo o Deep Learning, en la cual cada capa extrae características cada vez de un nivel más completo hasta obtener su respuesta final. Según se profundiza en la red neuronal, las funciones simples que son empleadas para el reconocimiento de patrones se combinan entre sí para obtener funciones más complejas.

El aprendizaje profundo puede emplear cualquiera de los modelos presentados en las ilustraciones anteriores, no obstante, vamos a trabajar con Aprendizaje supervisado etiquetando los datos y pasándoselos al modelo.

Para el desarrollo del aprendizaje profundo vamos a emplear el conjunto de librerías de TensorFlow, esta, es una herramienta de software abierto, lanzada por Google en 2015 que ha logrado un gran éxito comercial.

Esta herramienta está basada en Redes neuronales, de manera que puede relacionar datos en una red de forma simultánea como si fuera un cerebro humano. Esta, puede emplearse para el reconocimiento de patrones numéricos, patrones en textos o incluso reconocimiento de imágenes rápidamente gracias a si gran capacidad como conjunto de Redes neuronales.

Además, cuenta con un conjunto de herramientas y librerías externas que la hacen una herramienta muy equilibrada en cuanto a flexibilidad y escalabilidad y por tanto una de las mejores herramientas que encontramos en el mercado.

Para poder emplear eta librería podemos hacerlos a diferentes niveles, el que va a ser utilizado es el de más alto nivel, que nos permitirá conseguir nuestro objetivo de una manera sencilla.

El primer paso es la preparación de los datos de entrada de la red neuronal, una vez preparados los datos y normalizados para que tengan toda la misma forma podemos pasar a crear el modelo.

Con modelo nos referimos a una serie de operaciones matemáticas que son utilizadas para ajustar lo pesos de las variables de entrada y ponderarlas con el fin de que converjan y nos den la respuesta esperada.

El siguiente paso es la construcción y compilación del modelo, en el que indicaremos cual va a ser su optimizador y su función de pérdida. Para definir este modelo es necesario tener en cuenta el formato de los datos de entrada y las etiquetas para que este pueda ser empleado adecuadamente.

Hecho esto podemos entrenar al modelo pasándole las iteraciones que queremos que realice junto con los datos que hemos preparado anteriormente para el entrenamiento.

Finalmente podemos evaluar el modelo para ver si el resultado que ha obtenido es adecuado para el sistema en el que va a ser utilizado.

2.1.3. Categorización de ataques

En este apartado se pretende realizar una categorización de los ataques principales que una red puede sufrir, describiendo el funcionamiento de los mismos y la

manera de observarlos en la red. No se indica ni la forma de realizar el ataque ni la forma de prevenirlo, ya que son temas que se escapan a la temática de este proyecto.

Gran parte de los ataques que vamos a analizar son técnicas que pretenden dejar a un servidor inoperativo, englobadas en el tipo Denegación de Servicio (DoS – Denegation of Service).

El primer tipo de técnica que vamos a analizar es el ACK flood. Esta técnica se basa en la suplantación de la dirección IP del atacado y se lanzan mensajes del tipo petición de conexión (TCP-SYNC) a múltiples servidores, de este modo, todos estos servidores contestarán con asignaciones de conexión (TCP-SYN-ACK) al equipo que ha sido atacado.

Para poder observar este tipo de ataque lo que debemos buscar en nuestra red es la aparición de un gran número de aberturas de conexión desde una misma IP o, en caso de ser modificada, la aparición de paquetes de establecimiento de conexión altamente similares. Este ataque podemos catalogarlo como técnica empleada para realizar un ataque de denegación de servicio.

Otra técnica muy utilizada dentro de este segmento es el CAM Table [18] Overflow, que se aprovecha de la existencia de una tabla en todos los equipos que se encarga de asociar temporalmente direcciones de red a puertos, de forma que si se sobrecarga conseguimos que éste inunde la red.

Para detectarlo simplemente tenemos que observar si aparecen en los paquetes analizados gran cantidad de direcciones MAC aleatorias que el equipo registrará.

Las últimas técnicas que vamos a investigar de denegación de servicio son el ping de la muerte y el ping flood. El primero de ellos se basa en enviar un paquete del tipo ICMP más grande de lo que un equipo está preparado para soportar esto provoca que este se quede bloqueado. Aunque este ataque ya está obsoleto y todos los sistemas actualizados tienen medidas para contrarrestarlos, si observamos que un equipo de nuestro sistema está realizando este tipo de ataques podríamos catalogarlo como sospechoso. El segundo es la inundación de ICMP, donde el atacante bloquea el servicio por hacerle recibir una cantidad infinita de estos paquetes.

La detección de ambos es muy sencilla. Únicamente hay que observar los paquetes ICMP, o bien de gran tamaño y o bien en gran número, enviados a un único equipo.

Otro gran bloque de técnicas tiene el objetivo de, durante una comunicación legítima entre cliente y servidor, que el intruso se sitúe entre ambos, obteniendo acceso a todo su tráfico. Pueden ser catalogadas como partes de un ataque Men in the Middle.

Las principales son DHCP [19] Spoofing, ARP [20] Poisoning y posteriormente la aplicación de ingeniería social.

El DHCP Spoofing genera un servidor DHCP falso dentro de la red y pretende reasignarles las direcciones IP al equipo de la víctima redirigiendo todo el tráfico a través de una dirección IP determinada.

Para detectarlo, debemos buscar paquetes del tipo DHCP ACK (respuesta del servidor DHCP indicando que está validad la conexión), que provengan de direcciones IP no esperadas o que sean enviados de forma reiterada.

El ARP Poisoning se basa en engañar al equipo víctima para suplantar al Router o al proxy, haciendo que el atacado envíe los paquetes a través del equipo atacante. Para se envía una serie de paquetes ARP al equipo de la víctima para tratar de considere que el Router ha cambiado de dirección y que modifique la dirección MAC a la cual envía el tráfico de salida de la red.

Para detectarlo podemos observar paquetes ARP masivos a un equipo concreto con unas direcciones no esperadas y que no podemos encontrar típicamente en la red.

Una vez conseguido esto se pueden emplear técnicas de ingeniería social para convencer al usuario de que permita acceder a sus datos privados, ya que al acceder a todo el tráfico controla todos los datos que son enviados y recibidos.

Para detectarlo lo más sencillo y eficiente es detectar la base del ataque, este puede iniciar con uno de los procedimientos explicados anteriormente.

Otra herramienta habitualmente empleada por usuarios maliciosos son los ataques del tipo escaneo de puertos, por lo que también es importante detectarlos. Se basan en descubrir qué puertos tienen abiertos (esperando peticiones de clientes) las máquinas conectadas a una red y suelen ser la base de gran cantidad de ataques.

Para detectar este escaneo únicamente debemos observar la red a nivel global y comprobar que se esté realizando un barrido de puertos.

Hemos analizado estos ataques porque son la base de gran cantidad de ataques más complejos y dañinos. Por tanto, en ellos basaremos nuestro sistema de detección de intrusiones.

La siguiente tabla resumen resume estos ataques y algunos más, también significativos.

Tipo de Ataque	Detalles
ACK flood	Técnica de DoS que bloquea un servidor con peticiones de sincronización ya que este se bloquea esperando recibir los ACK de estas conexiones.
DHCP Spoofing	Técnica MiTM que mediante un servidor DHCP falso busca redirigir todo el tráfico del usuario a través de su sistema.
ARP Poisoning	Técnica de MiTM que busca redirigir el tráfico de un usuario a través de tu sistema.
Ataque de secuencia TCP	Técnica empleada para predecir la secuencia de TCP con el objetivo de secuestrar una sesión
Ataque LOKI	Es un software cliente servidor que permite esconder datos dentro del tráfico de la red

Ataque Man-In-The-Middle	Conjunto de técnicas que pretenden situarte entre un usuario de la red e internet teniendo acceso a todo su tráfico
CAM Table Overflow	Técnica de DoS que pretende atacar a la tabla CAM de los conmutadores con el objetivo de desbordarla provocando que el equipo actúe como un inundador de la red
DoS	Conjunto de técnicas que pretenden hacer que un servicio quede inoperativo
Escaneo de puertos	Técnica de reconocimiento que pretende observar que puertos hay abiertos en los equipos de una red. Suele ser el inicio de gran cantidad de ataques
FTP Bounce	Técnica que pretende hacer que a través de un FTP y mediante el comando PORT enviar archivos a los usuarios a atacar
ICMP Tunneling	Técnica utilizada para generar una conexión cifrada con una computadora de forma que se hace posible evitar los firewalls
Ingeniería Social	Conjunto de técnicas que pretenden hacer, sin que el usuario sea consciente que renuncien a que su información sea confidencial
KeyLoggers	Herramienta empleada para registrar las pulsaciones del teclado del usuario atacado
OS Finger Printing	Técnica empleada para conocer qué sistema operativo está siendo ejecutado en una máquina remota para conocer cuáles son sus debilidades.
Ping de la muerte	Técnica de DoS que se basa en el envío de un paquete ICMP de un tamaño que el sistema operativo no pueda manejar.
Ping Flood	Técnica de DoS que pretende enviar una cantidad ilimitada de paquetes ICMP al usuario víctima
Root kit	Software que pretende obtener el control de nivel de administrador en un sistema sin ser detectado. Para poder usarlos deben ser acompañados de otras técnicas como las ya mencionadas
TCP Session Hijacking	Técnica empleada para robarle a un usuario una sesión de TCP existente aprovechándose que la autenticación se realiza al inicio de la sesión

Tabla 2 Algunos ataques de red

2.2. Crítica al estado del arte

En cierto modo, todos somos conscientes de las necesidades que existen en cuanto a ciberseguridad en la red, pensando tanto en el ámbito doméstico como en el corporativo, por tanto, es responsabilidad de todos nosotros evolucionar y lograr mejoras fundamentales en este sector.

Es por esto por lo que es imprescindible el cambio de paradigma que se pretende enfocar en este TFG y la consecución de los objetivos previamente enumerados, es más, debemos ver más allá, siendo sabido que en más del 64% de las empresas españolas [21] no se emplea ninguna técnica de detección de intrusiones, ¿Cómo será la situación en nuestros hogares? En qué circunstancias almacenamos nuestros datos, como dijo *Fabián Chiera : "Si tienes bien en claro que lo importante son los datos, deberías proteger tus datos. Independientemente de dónde estén o en que formato estén."*.

Debemos tener en cuenta que la ciberseguridad empieza en nosotros, que nosotros tenemos la clave para lograr que los delitos informáticos sean menos frecuentes y nuestra información y datos mucho más seguros.

Presento una serie de gráficas que ilustran el nivel de ciberseguridad partiendo de la base explicada. La fuente es Google

En la siguiente figura procederemos a mostrar la evolución de la utilización de comunicaciones cifradas vía HTTPS durante los últimos años en diferentes entornos.

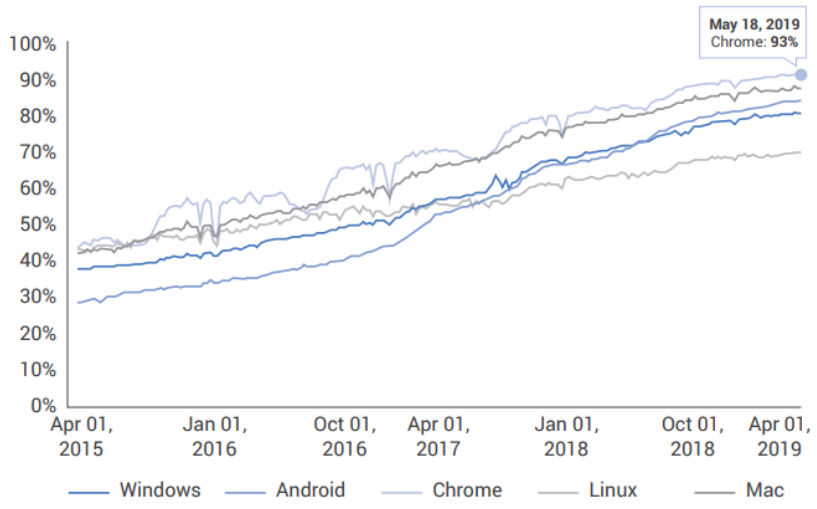

Ilustración 5 Porcentaje de uso de HTTPS por plataformas

Como vemos en la gráfica, han aumentado las conexiones web y nuestros datos viajan cifrados de extremo a extremo e impiden que terceros puedan espiarnos capturando el tráfico en tránsito. Por ejemplo, en marzo de 2015 solo el 45% de las páginas visitadas en Chrome utilizaba el protocolo https. Este porcentaje se elevó al 93% en mayo de 2019. No obstante, en plataformas como Linux, aún no alcanzaba el 80%, lo que deja claro que seguimos navegando de forma no segura.

Y además tras observar estos datos seguimos con unas inversiones ridículas en materias de ciberseguridad como observamos en la siguiente figura. En esta se muestra el aumento generalizado de la inversión en aspectos de la ciberseguridad por las empresas, no obstante, esta sigue siendo muy reducida.

Tal y como vemos en la figura, el gasto mundial en productos y servicios de ciberseguridad alcanzó en 2018 los 114 millones de dólares americanos, lo que supuso un

aumento del 12.4% respecto al año anterior y en 2019 un 8,7% más y aunque las previsiones sigan al alza, la inversión no es suficiente dado el impacto que pueden tener una brecha de ciberseguridad.

Segmento de mercado	2017	2018	2019
Servicios de seguridad	52,315	58,92	64,237
Protección de la infraestructura	12,583	14,106	15,337
Equipo de seguridad de red	10,911	12,427	13,321
Gestion de acceso de la identidad	8,823	9,768	10,578
Software de seguridad para el consumidor	5,948	6,395	6,661
Gestión integrada de riesgos	3,949	4,347	4,712
Seguridad de datos	2,563	3,063	3,524
Seguridad de aplicaciones	2,434	2,742	3,003
Otro software de seguridad de la información	1,832	2,079	2,285
Seguridad en la nube	185	304	459
Total	101,544	114,152	124,116

Fuente: Gartner.

Ilustración 6 Gasto en materia de ciberseguridad en millones de USD [22]

Observando en detalle, podemos ver la forma en la que está repartida la inversión. Es curioso como el aspecto que mayor inversión recibe es el de servicios de seguridad, lo que implica la contratación de empresas que cubran estos aspectos subcontratando los servicios en vez de que sea la propia empresa la que los controle.

Si observamos este aumento durante los últimos tres años, se aprecia que la distribución es significativamente uniforme, ya que todos los ámbitos aumentan de media un 1,339%, si bien es cierto que debemos destacar los que se ve que crecen ligeramente más despacio y los que sobresalen de este porcentaje creciendo más rápido que los demás.

Debemos tener en cuenta que, aunque la media es de 1,339% la mediana es de 1.22% lo que provoca que el aumento de la mayoría de las inversiones esté por debajo de la media. Además, la distancia es significativamente grande siendo de 1.29%, lo que implica, para la poca variabilidad que encontramos en los datos, que el aumento en seguridad en la nube es un dato anómalo.

El que está ligeramente por debajo de la media es el de software de seguridad para el consumidor, exactamente en un 1,2% lo que está por debajo de la media, lo que provoca que los usuarios finales siempre sean el último eslabón de la cadena y por tanto los que más desprotegidos están. Por tanto, sería conveniente que aumentara la oferta de productos para este ámbito y por tanto aumentara la inversión global.

No es de extrañar que al otro lado de la balanza aparezcan aspectos como el de ciberseguridad en la nube, que está un 1,8% por encima de la media y la seguridad en los datos que están un 1,02% por encima de esta.

Para poder explicar esto, debemos entender que la nube es una tecnología relativamente nueva, lo que implica que su crecimiento será muy superior al observado en el resto de los aspectos ya que deberá llegar al punto de equilibrio. No obstante, pese a que la cantidad de los datos en la nube crece de manera exponencial, su inversión no lo hace, lo que únicamente puede terminar en un aumento en los ataques a estos datos desde puntos no esperados que provoquen una gran cantidad de pérdidas.

Respecto a la seguridad en los datos, que es un aspecto que está directamente ligado con lo ya mencionado, es normal que se despunte su crecimiento. Además, el uso del IoT y la industria 4.0 y la aplicación de inteligencia artificial a estos campos implican que se incrementará de forma notable la cantidad de datos recogidos. Una vez más, la inversión no es suficiente y puede preverse que esto traerá lamentables consecuencias.

Por tanto, dada la situación global y siendo España uno de los países con menor número de expertos en ciberseguridad, es fundamental darles luz y priorizar proyectos de fin de grado con este tipo de intereses, de manera que, desde la base podamos seguir creciendo y evolucionando en el mundo de las tecnologías. La base del conocimiento está en las universidades y, tal como vemos en la siguiente ilustración, centros de investigación en ciberseguridad sí que tenemos y son de calidad.

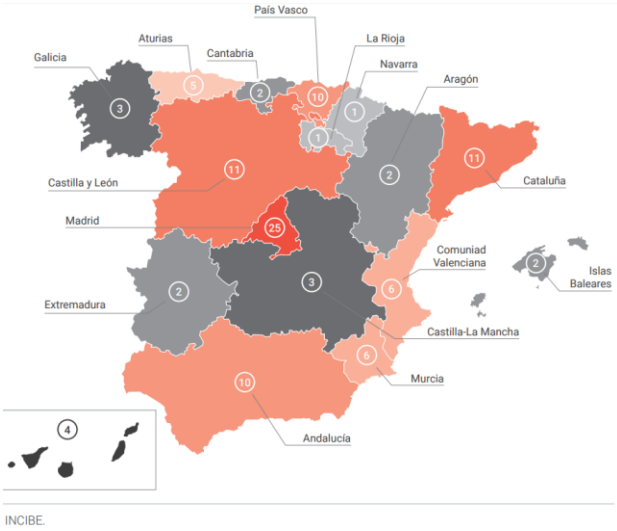

Ilustración 7 Localización de los centros de ciberseguridad en España [23]

Sobre esta figura podemos ver que contamos con 104 equipos de investigación en ciberseguridad repartidos por todo el territorio nacional, siendo la comunidad de Madrid la que cuenta con la mayor cantidad de expertos. Esto provoca que la mayor parte de soluciones de gran calibre sean implementadas desde allí. No obstante, no debemos olvidar que la mejor forma de afrontar estos problemas sería que se trabajase de forma común entre todos estos centros en colaboración y no en competencia entre ellos. En el resto de las comunidades también contamos con una gran cantidad de expertos en la materia, pero al no estar en la capital no se les dan siempre las mismas oportunidades.

Respecto a las temáticas de investigación en ciberseguridad en España puede decirse que son abundantes, pese a que, como vemos en el mapa, los equipos son reducidos (únicamente 1302 investigadores). Las temáticas se centran en la creación de

sistemas fiables y actualizables, privacidad, procesado de datos e infraestructuras críticas, temas altamente importantes para la resolución de problemas complejos. Por el contrario, viendo los puntos fuertes de crecimiento de la inversión, se echa de menos una mayor dedicación a temas alrededor de la nube o la seguridad de los datos más que el procesado de los mismos.

En síntesis, pese a que la situación global es muy pesimista, no debemos olvidar que este año hemos sido forzados a una digitalización rápida y que como vemos el avance en los gráficos durante los últimos años ha sido positivo (las cifras de este año no están publicadas, pero sabemos que producirán un impacto muy superior), y la expectativa es que en los próximos diez años lograremos estar mucho más protegidos.

2.3. Propuesta de valor

Como ya hemos comentado en apartados anteriores de este documento, el objetivo de este trabajo es crear un sistema de detección de intrusiones en redes con un enfoque muy diferente. Buscamos un cambio de paradigma, es decir, dejar de lado las redes de reglas e incorporar técnicas avanzadas de inteligencia artificial (*Deep Learning*) para evaluar la agresividad del tráfico de la red. De esta manera, en todo momento monitorizaremos en tiempo real nuestro sistema y podremos identificar todos los ataques para los que el modelo haya sido entrenado, proporcionando un porcentaje de probabilidad de estar sufriendo cada uno de los ataques.

El monitor propuesto utiliza la propia red monitorizada para intercambiar información entre los dispositivos. Para evitar sobrecargar la red, especialmente en aquellos momentos en que su utilización sea elevada, el sistema propuesto en este proyecto se adaptará a la capacidad de carga de la red. Cuando la utilización de la red lo permita, cada monitor local enviará los paquetes capturados al monitor global. Sin embargo, cuando esta utilización sea elevada, cada monitor local procesará los paquetes recibidos para extraer únicamente los parámetros significativos para cada tipo de ataque, y por tanto reduciendo el tráfico intercambiado.

Otro aspecto importante es que el IDS no depende de las estaciones de usuario, es decir, se ofrece uno o varios dispositivos hardware responsables de la monitorización. Incluso en redes pequeñas, que únicamente disponen de un segmento de red, un dispositivo podría incluir tanto el monitor local como el monitor global.

La escalabilidad de la solución permite incrementar el número de monitores locales para cubrir todos los segmentos de red necesarios. Con la configuración adecuada, podría incrementarse el número de monitores globales, responsabilizándose cada uno de ellos de la detección de ataques en un subconjunto de la red.

No podemos olvidar el factor coste, ya que se pretende desarrollar un sistema para todos los usuarios, lanzar un sistema que pueda ser asequible y que su uso realmente contribuya a la reducción de brechas de seguridad.

Otro factor diferencial en este proyecto es la capacidad de los nodos de trabajar tanto coordinadamente – proporcionando una visión global que proporciona una mayor detección de ataques – como de forma independiente. Cada monitor local ejecuta sus rutinas de detección y, si alguno de ellos detecta un ataque, lo notificarán al monitor global que lo notificará al administrador.

Estos dispositivos hardware serán fácilmente instalables, ya que, previamente entrenados en laboratorio, el usuario únicamente tendrá que conectarlos a la red. Cada monitor local encontrará al monitor global de su red si necesidad de ningún tipo de configuración previa y volcará sobre el mismo la información obtenida.

En conclusión, el sistema propuesto presentará innegables ventajas (Inteligencia Artificial, nula intrusión en las estaciones, escalabilidad y facilidad de instalación) sobre la mayoría de los IDS del mercado, y por tanto puede suponer una revolución en el mismo, al obtener mejores resultados y más específicos de una forma mucho más sencilla.

3. Capítulo 3

3.1. Análisis del problema

Tal y como hemos visto desde el primer punto de la memoria, el problema fundamental a tratar es lo poco seguras que son las redes hoy en día. Por tanto, debemos trabajar en los aspectos relacionados con la ciberseguridad buscando la protección a los usuarios proporcionándoles una serie de facilidades para hacer que sus redes sean más seguras.

Si tuviésemos que definir cuáles serían los requisitos de esta solución, aplicable tanto los hogares, como las empresas, serían los siguientes:

Por un lado, debemos brindarles un sistema que garantice una seguridad real en lo que están haciendo y, por tanto, notificarles al recibir una amenaza. Además, este sistema deberá ser muy sencillo y la configuración del mismo deberá ser completamente transparente para el usuario final.

Es muy importante que el IDS propuesto no consuma los recursos del equipo de los usuarios ni sobrecargue excesivamente sus redes, minimizando la penalización para los usuarios.

Además, el sistema propuesto deberá analizar diferentes segmentos de la red simultáneamente para poder monitorizar completamente la red y poder tomar decisiones sobre el cómputo global de lo que está ocurriendo en esta red, para ello empleamos un sistema hibrido que nos facilita este acceso.

Otro aspecto a destacar es que el número de falsos positivos debe ser muy reducidos, ya que ponemos foco en la comodidad del usuario final y pretendemos ofrecerle un sistema sobre el que tenga que intervenir en la mínima cantidad de ocasiones posibles.

Para evitar que con la aparición de nuevos ataques nuestro sistema sea inservible el producto desarrollado será fácilmente actualizable. Cada vez que un nuevo ataque sea catalogado, entrenaremos a nuestro modelo y se distribuirán las actualizaciones de éste entre todos los dispositivos de forma automática. Este proceso resultará sencillo, ya que únicamente deberemos añadir capturas etiquetadas de estos ataques a nuestro modelo al volver a entrenarlo.

Por tanto, para cubrir esta serie de necesidades es necesario crear un producto que sea capaz monitorizar la red del cliente de manera pasiva, sin instalar software en sus equipos. Además, este sistema deberá adaptarse a la carga de la red para la comunicación entre dispositivos.

En cuanto a la reducción de falsos positivos y negativos, son necesarias técnicas de detección más fiables y específicas. La mayor parte de los sistemas que podemos encontrar actualmente en el mercado trabaja en base a reglas aplicadas sobre las capturas, y generalmente sobre paquetes concretos. Para el sistema de Deep Learning no vamos a evaluar la posibilidad de ataque sobre paquetes concretos, sino que van a ser aplicadas sobre las trazas completas, de manera que seremos capaces de analizar el tráfico en su conjunto.

En cuanto a los dispositivos hardware deben ser de bajo coste (computadores monoplaca), de manera que este sistema sea asequible para todos los usuarios. Además, estos deben poder ser empleados por usuarios sin conocimientos, por industrias y por pequeñas corporaciones, de manera que deben ser de instalación sencilla y que no requieran ningún tipo de configuración.

Según vimos en los capítulos anteriores y comentaremos en el apartado de trabajo futuro, las perspectivas de este trabajo son muy amplias. No obstante, en este punto se pretende abrir una línea de investigación, dejar una base sobre la cual se pueda seguir estudiando y trabajando para finalmente lograr incrementar la seguridad de los usuarios gracias a dispositivos como éste.

Resumiendo, los requisitos de la solución son:

4. Alta garantía de detección.
5. Baja intrusión.
6. Acceso a diferentes segmentos de red.
7. Ser fácilmente actualizable.
8. Fácil instalación.
9. No necesidad de configuración.
10. Bajo coste

3.2. Análisis de seguridad

Al tratarse de una medida adicional de seguridad, el sistema propuesto en este trabajo puede ser a su vez objetivo de ataques maliciosos, a la par que desempeña un papel importante en la política global de ciberseguridad. Es por ello por lo que resulta fundamental estudiar de forma detallada la seguridad del mismo.

Primeramente, mencionamos los puntos principales que deben tenerse en cuenta para desarrollar adecuadamente este análisis.

- Evaluar la estabilidad y actualización automática del sistema, para evitar agujeros.
- Confidencialidad: comunicaciones cifradas entre ML y MC.
- Autenticación: Uso de certificados digitales en ambos extremos.

Para ello dividiremos las amenazas en diferentes grupos: riesgos conocidos, riesgos predecibles y riesgos impredecibles, siendo los últimos los que incluyen una solución más compleja.

Debemos tener en cuenta que al tratarse de una aproximación no ha sido implementado el aspecto de confidencialidad, para ello sería necesario hacer que los dispositivos se comunicarán mediante comunicaciones cifradas, es decir, empleando la capa SSL (secure socket layer).

En cuanto a los riesgos predecibles, podemos destacar la posibilidad de que algún posible atacante, sobre todo de redes domésticas o de pequeñas corporaciones, genere un punto de acceso falso y consiga redirigir todo el tráfico a través de él. Esto es algo que escapa completamente al alcance de este proyecto. No obstante, en fases más avanzadas de este trabajo, podrían dotarse de certificados digitales a los elementos de forma que no fuera posible la suplantación.

Por último, respecto a los riesgos impredecibles, sabemos que pueden crearse ataques nuevos para los cuales nuestro modelo no haya sido entrenado, por esto, debemos estar pendientes de actualizaciones y cada vez que aparezca un nuevo ataque, incorporar las actualizaciones para que nuestros dispositivos lo detecten lo antes posible.

Para facilitar la comprensión de las amenazas a la seguridad para que el lector los pueda diferenciar los hemos organizado en el siguiente diagrama de calor. No obstante, dada la complejidad y ambición del proyecto, en fases más avanzadas del desarrollo debería hacerse un análisis de riesgos más detallado con mucha más perspectiva que la aproximación que pretende este TFG.

Catastrófico	Avería Hardware	Falso punto de acceso	Nuevos ataques
Moderado		Detección de vulnerabilidades en el sistema	Adición de segmentos de red no cubiertos
Insignificante			
	Raro	Posible	Seguro

Tabla 3 Análisis de seguridad

Atendiendo a la tabla 3, vemos que lo que más debemos tener en cuenta es la constante actualización del servicio de manera que podamos estar prevenidos de la mayor cantidad de ataques posibles reduciendo al máximo el impacto que esto pueda causar.

Una vez previstas todas estas problemáticas procedemos a analizar el marco legal y ético de la solución.

3.3. Análisis del marco legal y ético

En los siguientes apartados profundizaremos en aspectos del desarrollo que debemos tener en cuenta para lograr nuestros objetivos de manera ética, legal y sostenible.

3.3.1. Análisis de protección de datos

Al monitorizar de forma pasiva las redes del usuario (doméstico o empresarial), el sistema propuesto manejará información especialmente delicada, tanto datos personales como datos confidenciales del mismo. Por tanto, resulta muy importante proteger y no permitir el acceso indebido a estos datos.

Pese a que sería muy interesante que el administrador tuviera acceso a los datos recogidos por el sistema, para ofrecer a los usuarios un servicio mucho más personalizado, se ha decidido, al menos a priori, que el sistema va a trabajar de manera cerrada una vez sea instalado y montado. Podrá recibir actualizaciones sobre el modelo, pero, en ningún caso, se va a acceder a los datos del cliente. Para ello, el sistema no almacenará ningún tipo de información sobre lo que monitoriza. Únicamente, en tiempo real, notificará al administrador del sistema en caso de detectar algún ataque. En este caso, serán visibles por el administrador únicamente las tramas comprometidas, para que este pueda determinar exactamente de donde proviene el problema y ponerle una solución. Para evitar que este tenga acceso a información confidencial, las tramas serán preprocesadas y se mantendrán únicamente las cabeceras relevantes.

Es importante remarcar que en ningún caso los datos de aplicación que analiza el sistema deberían ser visibles para el administrador.

3.3.2. Propiedad intelectual

En cuanto a la decisión del tipo de licencias que van a emplearse en el sistema que se va a desarrollar tenemos muchas posibilidades. No obstante, para poder lograr un desarrollo lo más ágil posible y contar con la comunidad más grande disponible trabajando en el proyecto con el objetivo de conseguir que cada vez las redes sean más seguras, la licencia principal deberá ser Creative Commons. De este modo, aunque el sistema sea comercializado para gran cantidad de usuarios sin conocimientos, para los

usuarios experimentados podrán trabajar en las mejoras de este y poderlas utilizar libremente.

De esta manera, contaremos con una gran comunidad de usuarios que nos ayudarán a mejorar nuestro producto y a la vez tanto para corporaciones como para usuarios sin conocimientos podremos vender un sistema completamente funcional que sea transparente para ellos.

Gracias a esto cumpliremos la motivación social del proyecto, ya que podremos llegar a todos los grupos de usuarios que puedan utilizar este sistema y realmente lograremos una mejora en cuanto al panorama en el que se encuentra la ciberseguridad en este momento.

3.3.3. Otros aspectos legales

Respecto a aspectos legales específicos del proyecto sería necesario establecer el lugar donde queramos instalar nuestro sistema. Es decir, en caso de querer instalarlo en un espacio público deberemos cumplir una serie de auditorías específicas que para un uso doméstico no sería necesario (ISO 27001, ISO 27032, ISO 27033...). De todos modos, voy a mencionar, a priori, cuáles serían los aspectos legales que se deberían cumplir.

Estos aspectos legales los podemos agrupar en tres grupos principales [24], seguridad informática, delitos informáticos y protección de datos personales.

Respecto al bloque de seguridad informática, debemos destacar el cumplimiento de la estrategia de ciberseguridad de la unión europea, que pretende aplicar una serie de medidas que pretenden promover los derechos de los ciudadanos basándose en la defensa contra delitos informáticos. También se tienen en cuenta las medidas impuestas en la directiva del parlamento europeo y del consejo del 6 de julio de 2016, que incluyen aspectos como la fiabilidad y la seguridad de las redes. Respecto al código de Derecho de la ciberseguridad, como el Real Decreto 381/2015, de 14 de mayo, por el que se establecen medidas contra el tráfico no permitido y el tráfico irregular con fines fraudulentos en comunicaciones electrónicas, que entre otros es el que mayor implicación directa adquiere sobre este TFG. Por último, hay que considera los aspectos referentes al bloque de código de la seguridad privada, cuyo fin es satisfacer las necesidades de seguridad de todos aquellos usuarios que decidan instalar componentes de seguridad de empresas privadas controlando aspectos como lo relacionado con la protección y tratamiento de datos.

En cuanto al bloque de delitos informáticos, es interesante poner atención sobre el código penal y la legislación complementaria que contiene entre otras normas la Ley 5/2014, de 4 de abril, de Seguridad Privada.

Para la protección de datos personales debe aplicarse el código de protección de daos de carácter personal que incluye medidas como las aportadas en el punto de

protección de datos y el código de derecho al olvido, que es el derecho que tiene el titular de un dato personal, a borrarlo y suprimir su información sí que deje ningún rastro.

Además de la legislación vigente en España y la UE, desde el punto de vista de certificación de calidad son aplicables diversas normas, de entre las que se destacan:

El primero, y desde mi punto de vista fundamental, es la ISO 27001, que contempla la normativa general que asegura la protección de la información y de los datos de carácter personal, que como hemos visto en el apartado 3.3.1 es algo que debemos tener muy en cuenta.

También hay que considerar la norma ISO 27032, que define las guías en el ámbito de la ciberseguridad y se centra en cubrir los espacios o huecos no cubiertos por normas anteriores de seguridad en este ámbito conceptual más amplio, en el que aparecen nuevos ataques y los riesgos asociados a éstos.

La última normativa a mencionar es la ISO 27033, orientada hacia la gestión de la seguridad, aplicaciones de servicios y/o redes, seguridad de los dispositivos de red y a la seguridad de información que se transmite mediante enlaces de comunicaciones, objetivo fundamental de este proyecto.

Dentro de este apartado se ha diferenciado entre normativa legal, de obligado cumplimiento, y normativa ISO de calidad. El trabajo presentado cumple, a entender del autor, las normativas legales existentes. También se ha atendido a las recomendaciones de las normativas ISO de calidad a la hora de especificar las prestaciones del sistema, si bien sería necesario un proceso de auditoría para garantizar el cumplimiento completo de las mismas.

3.3.4. Ética

El mayor dilema que aguarda este TFG es el análisis de datos en la personalización del modelo para adecuarlo al funcionamiento de la red en la que va a ser utilizado, ya que este, va a ser mucho más eficiente si se va retroalimentando de forma supervisado y aprendiendo sobre la red en la cual está siendo utilizado. No obstante, esto facilitaría en gran medida el oportunismo de los administradores de sistemas para traficar con los datos de los clientes, ya que este sistema va a estar continuamente monitorizando la red y va a contener toda la información que atraviesa la misma.

Otro aspecto que analizar es el tipo de licencia que queremos atribuirle al proyecto, ya que este surge para, como hemos visto anteriormente, cubrir una necesidad cada vez más importante, que es la seguridad de todos los usuarios de la red. Es por esto, que no considero adecuado obtener un beneficio económico del mismos. No obstante, para asegurar la continuidad del proyecto y dar la posibilidad de que siga creciendo y dando un servicio de mayor calidad, considero, que, efectivamente, debería finalizar en un sistema comercial, pero con una parte de código abierto que puedan utilizar los usuarios experimentados.

Respecto a las capturas de los paquetes de los ataques, es evidente que deben realizarse en máquinas virtuales de redes supervisadas sin poner en riesgo la integridad de los datos ni de las redes de ningún usuario y que todas las prácticas que han sido realizadas hasta el momento han sido con fines educativos y tampoco se ha visto comprometida la seguridad de ninguna red.

Ningún sistema informático ha sido dañado en el rodaje de este TFG.

3.4. Análisis de riesgos

Respecto al desarrollo del proyecto debemos tener en cuenta una serie de aspectos inherentes al mismo, que en caso de ocurrir no permitirían el desarrollo correcto de este.

El primero de ellos es el cumplimiento de plazos, en este punto englobamos el desarrollo del sistema completo con sus protocolos de comunicación y el modelo de Deep Learning que deben estar listos en la entrega del presente documento.

El siguiente riesgo que debemos controlar es que el modelo que generemos sea funcional y converja, es decir, que tenga una tasa de aciertos válida. Además, este elemento debe poder ser implementado dentro del sistema que hemos generado y ser funcional junto con este.

Otro riesgo que no debemos olvidar es que el funcionamiento del entorno de pruebas sea el esperado y responda adecuadamente a todas las situaciones para las que ha sido diseñado y nos permita testear el software. Dado el modelo de sistema que planteamos en los capítulos anteriores, debemos tener en cuenta la aparición de falsos positivos y de falsos negativos ya que es imposible obtener una red neuronal perfecta y que no genere estos errores. Para tratar de minimizar la aparición de este tipo de problemas, nuestro modelo deberá ser entrenado con un número de paquetes muy elevado para conseguir tasas de acierto próximas a la unidad.

También debemos controlar que en caso de que el equipo donde se está desarrollando del sistema falle que el proyecto pueda ser concluido y no se pierda información relevante.

Otro aspecto que no debemos olvidar es que, al querer integrar el software en un sistema empotrado, este debe ser suficientemente potente para permitir la instalación en el mismo. Además, el modelo debe ser lo suficientemente eficiente como para poderlo usar en dispositivos de bajo coste.

Debemos tener en cuenta también el caso de encontrarnos redes altamente sobrecargadas, de forma que nuestro dispositivo debe adaptarse a la red donde vaya a ser instalado.

Presentamos, al igual que en análisis de seguridad, un mapa de calor que nos permite observar cómodamente los riesgos ya planteados.

Catastrófico	Que el usuario apague el sistema	No convergencia del Deep Learning	Desarrollo fuera de plazo
Moderado	Redes altamente sobrecargadas	Hardware insuficiente	Falsos negativos
Insignificante		Eficiencia mejorable en los sistemas	Falsos positivos
	Raro	Posible	Seguro

Tabla 4 Análisis de riesgos

3.4.1. Análisis de soluciones posibles

Una vez realizado el análisis detallado del problema, el objetivo planteado consiste en crear una solución que permita detectar intrusiones en las redes y prevenirnos de ciberataques. Para esto se plantean una serie de posibilidades.

La primera de ellas es la creación de un software a instalar en el computador del usuario para que se ejecutara de manera paralela al antivirus, identificando problemas específicos de nuestro equipo. Esta solución aporta únicamente una visión local del problema , ya que dependiendo del punto de la red en el que se produzca el ataque podría no ser detectado. Además, introduce sobrecarga en las estaciones de trabajo del usuario, lo cual no resulta aceptable en muchos casos. Finalmente, algunos usuarios empresariales limitan el software a instalar, y por lo que la solución no es viable.

La segunda opción consiste en instalar un único monitor situado a la altura del Router con la potencia suficiente para analizar toda la red. No obstante, en este caso tampoco se es capaz de monitorizar la red completa, dependiendo de la topografía de la misma. Sería una solución que nos proveería de un marco de trabajo más fiable pero no llegaría a ser suficiente para el nivel de protección que pretende este sistema.

La tercera posibilidad es buscar crear un sistema únicamente distribuido, incluyendo una serie de dispositivos hardware distribuidos por la red, estos contarían con una gran observabilidad local, ya que podrán analizar adecuadamente los datos que reciban, no obstante, no cuentan con una perspectiva global de la red y por tanto una gran variedad de ataques dejaría de estar contemplada.

La cuarta solución es la creación de un sistema distribuido híbrido que sea capaz de analizar en tiempo real el tráfico de la red pero que además tenga una plataforma central desde donde se gobiernen todos los dispositivos hardware y a su vez reciba los datos y los analice buscando paquetes no confiables. De esta manera contamos con una buena observabilidad local y además una visión global que permite el análisis completo.

Para la implementación de cualquiera de las soluciones anteriores podemos optar por varias opciones. La primera de ellas es el preprocesamiento de la información, es decir, si el sistema analizará las tramas tal como han sido transmitidas (formato RAW) o se realizará un procesamiento previo antes de analizarlas.

Otra decisión consiste en el método de análisis de los datos. Como ya hemos visto, la mayor parte de sistemas que podemos encontrar en el mercado están basados en reglas, lo que nos permite definir una serie de patrones, que, en caso de cumplirse, identificarán el ataque que han registrado. Por otro lado, existe la opción de analizar los paquetes mediante una red neuronal, de forma que, en vez de ceñirse a un patrón, el tráfico es evaluado en su conjunto. Como inconvenientes de este segundo método, es necesario entrenar un modelo de red neuronal de forma previa, si bien los resultados obtenidos permiten un nivel de precisión muy superior al obtenido mediante reglas. Otro inconveniente de este método es su mayor coste computacional, que hace muy complicado ejecutarlas de forma no intrusiva en los computadores de usuario.

Por último, viendo que las mejores opciones son la tercera y la cuarta, deberemos de analizar en ellas que en caso de existir muchos dispositivos en la red de qué manera vamos a realizar las conexiones entre los mismos, para ello tenemos dos opciones, por un lado, que los dispositivos estén en constante comunicación y que los análisis los hagan en conjunto. Esto, en la opción tres aportaría la visión global que le falta. No obstante, esto provocaría sobrecargas enormes ya que deberían enviarse todos los paquetes a todos los nodos. Por tanto, debemos trabajar para minimizar el impacto el mínimo posible. En cuanto a la cuarta opción sabemos que los dispositivos deberán mandar la información al nodo central, estás comunicaciones también podrán sobrecargar la red, por tanto, deberá contar con un sistema de comunicaciones inteligente.

De entre todas estas opciones, en el siguiente apartado se toman las decisiones a aplicar en la solución propuesta

3.4.2. Solución propuesta

Hemos visto en los análisis previos que el problema a tratar resulta extraordinariamente complejo. Por tanto, en este trabajo se presentará una versión reducida de las funcionalidades óptimas, con la intención de marcar una línea de trabajo que oriente futuras implementaciones. El IDS propuesto se denominará AHIDS (Affordable Hybrid Intrusion Detection System)

Vamos a comenzar por la arquitectura de AHIDS. Como hemos visto entre las posibles opciones, con todos los planteamientos, la solución más completa es la creación

de un sistema híbrido distribuido / centralizado, de manera que se consiga monitorizar correctamente toda la red.

El sistema propuesto está formado por diferentes dispositivos Hardware, que denominaremos monitores locales, los cuales se comunican con un dispositivo central (monitor global) al que le transmiten, dependiendo la capacidad de carga de la red, la información que están monitorizando.

Cada uno de estos nodos monitoriza el tráfico que pasa por su segmento de la red y evalúa la carga de la misma. En función de la información que registren, las capturas se envían al nodo central en su totalidad (formato RAW), para que éste analice el conjunto de tráfico observado en todos los nodos y determine la probabilidad de estar sufriendo un ataque informático, o bien, en caso de que la utilización de la red supere un determinado umbral, y con el fin de no sobrecargar la red observada, los monitores locales preprocesarán los datos capturados para obtener los indicadores fundamentales de cada paquete capturado y transmitirlos al nodo central.

Un ejemplo de este sistema podría ser el que encontramos en la siguiente ilustración:

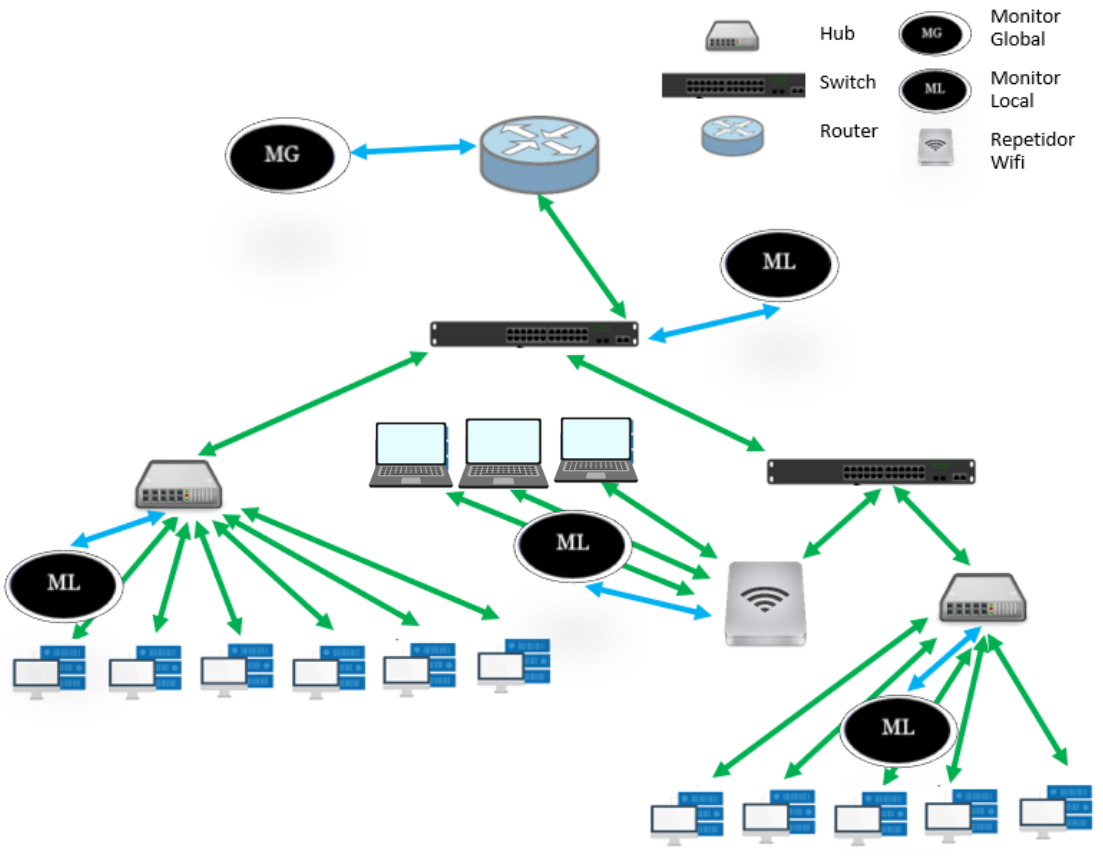

Ilustración 8 Topología de red planteada

En este caso concreto los ML (Monitores locales) serán los dispositivos encargados de la monitorización pasiva de las tramas del sistema y el MG (Monitor global) será el monitor principal encargado del análisis exhaustivo (posteriormente detallaremos los elementos de la figura).

Cada uno de los nodos cuenta con un pequeño modelo de Deep Learning capaz de detectar, en los paquetes recibidos, indicios de estar sufriendo un ataque, , de forma que realizará una primera evaluación de los datos que monitoriza antes de enviarlos.

El monitor global cuenta con una funcionalidad más elaborada de Deep Learning que permite un diagnóstico más preciso sobre una mayor cantidad de información, ya que dispone del conocimiento que implica contar con todos los paquetes que pasan por la red. En caso de detección de un posible evento o ataque malicioso, el MG podrá centrar la escucha en alguno de los nodos, analizando por separado las capturas del mismo, y avisar al administrador del sistema de posibles brechas de seguridad dentro de la red indicando dónde ha sido detectado.

La selección de qué ataques o eventos van a ser monitorizados depende de las aplicaciones de la red a monitorizar. Así, los patrones a detectar serán distintos en redes domésticas, domóticas, industriales, empresariales, etc. Por ello, el entorno propuesto debe ser fácilmente adaptable a su entorno.

El número de monitores locales necesarios dependerá de la topología de la red, siendo conveniente la instalación de un dispositivo en cada segmento de red. De esta forma se posibilita la monitorización de la mayor cantidad de datos posible. Debemos tener en cuenta que estos dispositivos deben tener suficiente capacidad para poder procesar todos los datos monitorizados.

No obstante, el hecho de introducir un gran número de dispositivos aumentará la carga de la red, de manera que los dispositivos deberán ser más potentes para poder procesar todo el tráfico que sea enviado y, además, la infraestructura de comunicaciones con la que cuente el lugar donde se instale deberá poder soportar el tráfico habitual de la red y el que generen estos monitores locales.

Debemos tener en cuenta que, aunque el modelo de nuestro sistema es pasivo, ya que únicamente se realizará el trabajo de detección y alerta. No obstante, dada la precisión que se puede lograr con los nuevos algoritmos, en un futuro podrían realizarse adaptaciones para responder a las amenazas que pueda detectar. De esta manera, dejaría de actuar como IDS y comenzaría a actuar como un IPS, es decir, sería una solución activa dentro de nuestra red. Este aspecto se comentará en las futuras ampliaciones del proyecto.

Desde el punto de vista del tratamiento de los datos, éste se realizará tanto de forma distribuida – en cada uno de los monitores locales con información local – como de forma centralizada en el monitor global, basándose en la información obtenida en todos los monitores locales. De esta forma es posible aplicar técnicas complejas sobre el estado completo de la red.

Para reducir la intrusión sobre la red observada causada por las comunicaciones entre monitores locales y monitor global, los monitores evaluarán constantemente la utilización de la red, de forma que, en caso de detectar una carga elevada que pueda verse influida por el tráfico del IDS, los paquetes capturados sean preprocesados antes de enviarlos al nodo central, de manera que este hará el procesamiento y los evaluará para detectar posibles ataques, en cualquiera de ambos formatos.

A pesar de que la mayoría de los IDSs existentes emplean reglas de comportamiento para detectar intrusiones y ataques, en este trabajo hemos optado por emplear técnicas de inteligencia artificial, en concreto una red neuronal con aprendizaje basado en Deep Learning. Con ello esperamos, por un lado, aumentar la capacidad de detección, al plantear algoritmos que evalúan todo el tráfico generado en la red a monitorizar en su conjunto (reducir el número de falsos negativos), sin incrementar significativamente el número de falsos positivos.

Por tanto, la solución propuesta consiste en un conjunto de monitores locales, capaces de capturar el tráfico que circula por su segmento de red en modo promiscuo y evaluar mediante una red neuronal la probabilidad de que se esté produciendo uno de los ataques predeterminados. En caso de que esta probabilidad supere el umbral predefinido, se generará una alarma dirigida al administrador del sistema.

De forma paralela, los monitores locales evaluarán la utilización de la red. Si la utilización es baja, reenviarán los paquetes recibidos al monitor global, que podrá reevaluar el conjunto de todos los paquetes de la red. Por el contrario, si la utilización es alta, el monitor local procesará los paquetes para extraer parámetros significativos de los mismos, enviando únicamente estos parámetros al monitor global para su evaluación.

Aunque el desarrollo de este trabajo no sea el sistema definitivo, se pretende que el trabajo de investigación realizado permita que en un futuro se pueda seguir avanzando en esta línea de trabajo.

3.4.3. Plan de trabajo

Para implementar el plan de trabajo ha sido utilizada la herramienta Microsoft Project para acompañar el proyecto durante todas sus fases. A continuación, analizaremos el diagrama de Gantt y explicaremos los trabajos realizados en cada una de las fases.

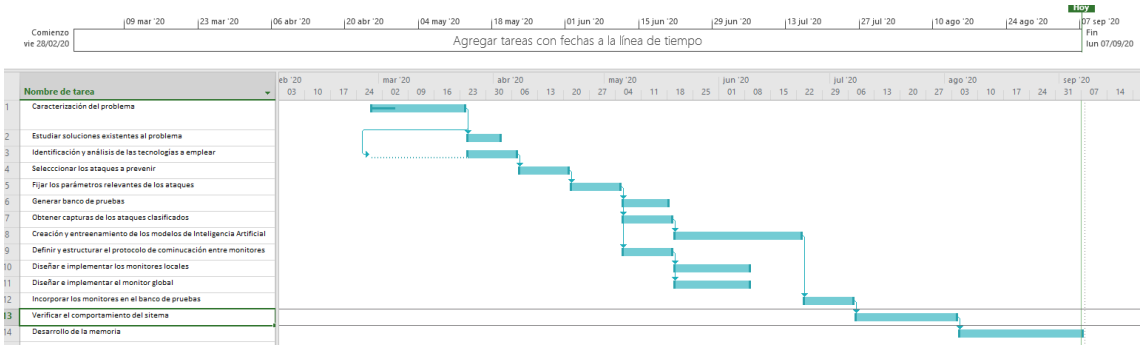

Ilustración 9 Diagrama Gantt

3.4.4. Presupuesto

Ilustración 10 Diagrama de Gantt

Tal como se aprecia en el diagrama de Gantt, el tiempo empleado en el TFG es tres meses. Considerando una dedicación parcial de 10 horas semanales al principio y dedicación completa al final, el número de horas es aproximadamente 425, contando el periodo de formación en las materias que existía desconocimiento, toda la parte de investigación y por supuesto, la parte del desarrollo de la aproximación al sistema completo que se busca crear con esta nueva línea de trabajo.

Por tanto, contando a que la hora sea pagada a 12€ podríamos calcular el coste del desarrollo de este TFG en 5100€, en caso de ser desarrollado por un ingeniero informático Junior sin experiencia.

A este coste había que añadir la parte de creación de los nodos e implementación de los recursos hardware, en caso de dispositivos ad-hoc. En caso de adquirirlos, podríamos contar con unos 80€ de coste por nodo y unos 150€ de coste del sistema central.

Para decantarnos por estos precios se ha realizado un estudio de especificaciones de los componentes que podemos utilizar de las posibilidades que hay en el mercado y como nodos las primeras pruebas deberían realizarse con las placas Beagle Board BBB01-SC-505 y el módulo central con una Beagle Bone AI.

Viendo el estado del proyecto es posible que sea necesaria la introducción de más elementos en un futuro para asegurar el funcionamiento correcto del sistema. Por tanto, este presupuesto también es una aproximación.

4. Capítulo 4

4.2. Diseño de la solución

Siguiendo las especificaciones y decisiones de diseño que se han adoptado en apartados previos, el diseño completo de la solución parte de un esquema básico, similar al presentado en la ilustración 8

En este esquema vamos a contar con un conjunto de nodos monitor local que van a trabajar como *sniffer* monitorizando toda la información que circula por su segmento de la red, y un monitor global que va a procesar la información de todos los monitores locales para detectar si la red que estamos monitorizando ha recibido algún ataque de alguno de los tipos definidos.

Además, deberemos crear un modelo de inteligencia artificial que sea capaz de, entrenándolo previamente, detectar estos ataques.

Como hemos visto en el apartado del estado del arte, existe una gran variedad de ataques que podríamos detectar en nuestras redes y en el sistema completo se debería contar con una base que incluya una gran parte de estos, no obstante, para el caso práctico únicamente hemos preparado un dataset de un conjunto de ataques del tipo Port Scan, y otro conjunto del tipo MiTM, ya que haciendo funcionar nuestro sistema con esos ataques será suficiente para comprobar la funcionalidad del mismo. No obstante, se investiga sobre más tipologías de ataque para facilitar la continuación del trabajo sobre esta temática.

El hecho de elegir esos ataques concretamente ha sido por dos motivos. El primero es que son los más frecuentes y pueden llegar a ser muy peligrosos y el segundo es que son la base de muchos otros ataques más complejos y que, si somos capaces de prevenirlos desde el principio podemos conseguir que la red sea más segura.

4.3. Arquitectura del sistema

En cuanto a la arquitectura del sistema pasamos a especificar el formato que va a tener el sistema que hemos seleccionado para cumplir todos los objetivos que se han definido en los pasos previos de la memoria.

Como se ha anticipado, el sistema diseñado cuenta con dos grandes bloques fundamentales, por un lado, unos equipos que funcionarán como monitores locales, que llevarán cargado un programa que les permita observar todo el tráfico que haya en la red

a su alcance. Para ello emplearán su tarjeta de red y un sniffer que haga la función de la captura de los datos. Además, cuentan con una función para buscar al monitor global, conocer su IP, autoconfigurarse y otra que le permite evaluar el estado de carga de la red, de forma que en función de la carga puede actuar de la forma que le corresponda.

El segundo elemento del sistema va a ser un nodo central que se va a encargar de recibir toda la información enviada por los sensores, bien en formato bruto (raw) o bien ya preprocesada por el monitor local, la procesará y la almacenará para posteriormente pasársela al modelo de inteligencia artificial y que éste proporcione un resultado en función de la confiabilidad de la red.

En la siguiente ilustración podremos observar una aproximación de lo que podría ser la representación del sistema que se pretende utilizar.

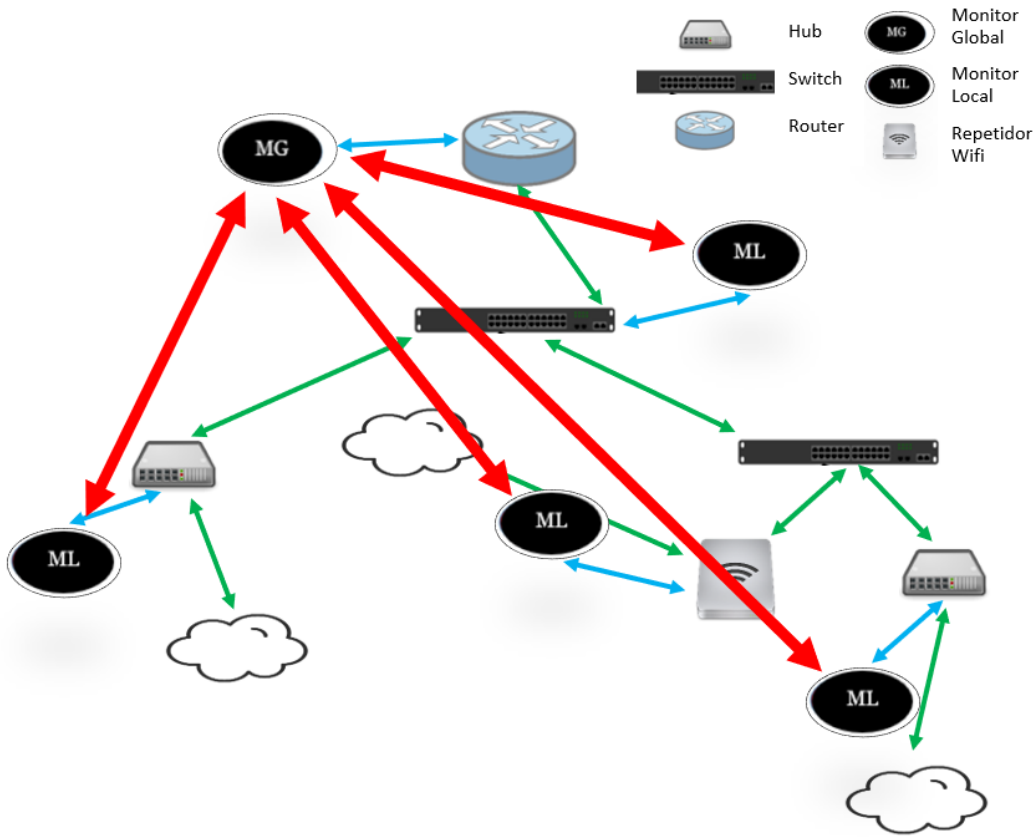

Ilustración 11 Comunicación entre monitores

Como vemos en la ilustración disponemos de una serie de monitores locales, como hemos comentado antes, que transmitirán la información adquirida al monitor global formando una topología en estrella.

Por otro lado, podemos observar en la ilustración 2 la representación del modelo de inteligencia artificial que será empleado en el proyecto.

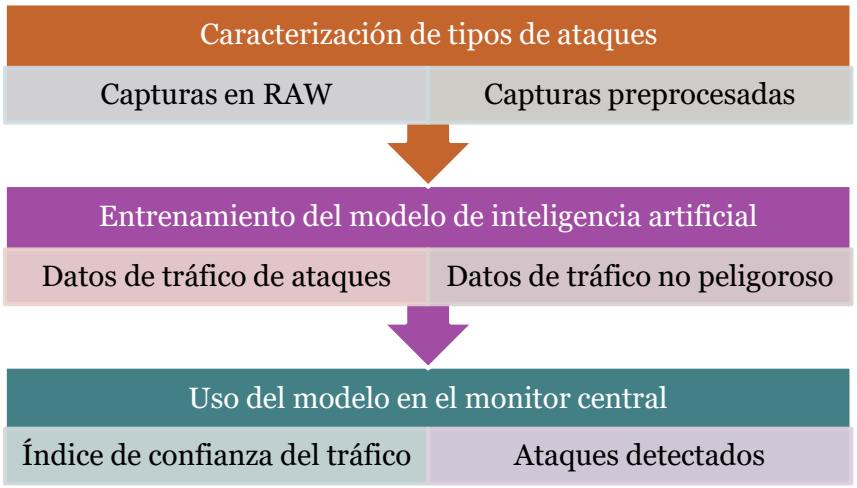

Ilustración 12 Representación inteligencia artificial

Tal y como vemos en esta ilustración, la arquitectura es muy clara. Primeramente, se realiza es una caracterización de los tipos de ataque que podemos encontrar. En el estado del arte podemos encontrar la caracterización de todos los ataques, en este paso generaremos datos sobre los ataques seleccionados.

La información capturada en cada uno de los ML se transmite al MG en dos formatos diferentes, en función de la carga de la red. Por tanto, es necesario que los módulos de IA puedan aplicarse tanto a datos en formato RAW, como a capturas preprocesadas, obteniendo los parámetros que consideramos necesarios para funcionar con el mínimo de información.

Las capturas se etiquetan para indicar si responden a ataques o a datos de tráfico real no peligroso, de manera que podamos entrenar a nuestro modelo para que realice predicciones fiables de lo que está ocurriendo.

Por último, emplearemos este modelo que hemos creado para, en tiempo real, monitorizar todo el tráfico capturado sobre la red.

4.4. Diseño detallado

En este apartado vamos a entrar en profundidad en todos los aspectos que hemos incluido en la arquitectura.

Vamos a comenzar desgranando los monitores locales (ML). Éstos realizarán varias funciones. La primera y principal será la de adquisición de datos – monitorización – que se dedica a capturar el tráfico de su segmento de red. Para esto emplearemos la tarjeta de red en modo promiscuo.

La segunda función de los ML consiste en detectar la utilización de la red, de manera que podamos adaptar la cantidad de datos transmitidos por el ML en función de la misma. Para reducir el tamaño, el ML también podrá preprocesar los paquetes capturados y transmitir únicamente los parámetros más relevantes de cada paquete en vez de su contenido completo.

El ML dispone de IA, diagnóstico de ataques y alerta a través del MG mediante el sistema de comunicación.

Finalmente, el ML dispone de rutinas de comunicaciones que le permite conectarse con el nodo principal y compartir con éste toda la información monitorizada.

Cada una de estas funciones se ejecuta en paralelo con el resto, por lo que la implementación se basa en varios hilos de ejecución que se ejecutan de forma concurrente.

Para poder realizar y comprobar el estado de todo el sistema, vamos a suponer que se trata de una red del tipo Gigabit Ethernet y emplearemos esta para calcular la carga de la red, por tanto, únicamente deberemos saber cuál es la dirección IP del monitor global, cosa que descubriremos a partir del sistema de instalación automática.

Este sistema de instalación se basará en un mecanismo “Plug & Play” que le permitirá, una vez conectado en la red, descubrir la IP asignada al monitor global y autoconfigurarse en base a ese parámetro de manera que si éste se desconecta vuelva a buscarlo y encontrarlo.

En la siguiente ilustración encontraremos un resumen de los elementos principales de este bloque.

Ilustración 13 Resumen diseño del monitor local

El siguiente elemento a analizar será Monitor Global. Éste cumple tres funciones principales. La primera de ellas será la habilitación de un servidor que permita recibir las conexiones de diferentes clientes, de manera que estos le envíen la información de las capturas realizadas.

El segundo bloque es el conjunto de elementos que preparan los datos monitorizados para suministrarlos al modelo de inteligencia artificial, ya que éste requiere un formato específico de datos para poder procesarlos. Este formato – vectorización – se describe en detalle en el desarrollo de la solución. También debe unificar los datos recibidos de todos los ML en un único vector para ser analizados por la IA.

El tercer y último bloque es el del uso del modelo de inteligencia artificial para el análisis y la interpretación de los resultados que obtengamos sobre el mismo.

Los atributos principales de este programa son la lista de paquetes recibidos y procesados, una lista auxiliar de procesamiento y el puerto en el que vamos a iniciar el sistema.

En la ilustración 11 podemos observar la imagen resumen de este elemento y en la ilustración 12 el mapa detallado de un ejemplo de sistema.

Ilustración 14 Resumen diseño monitor global

En este mapa observamos, por un lado, las conexiones entre los elementos y por otro lado los datos enviados en un posible caso de uso.

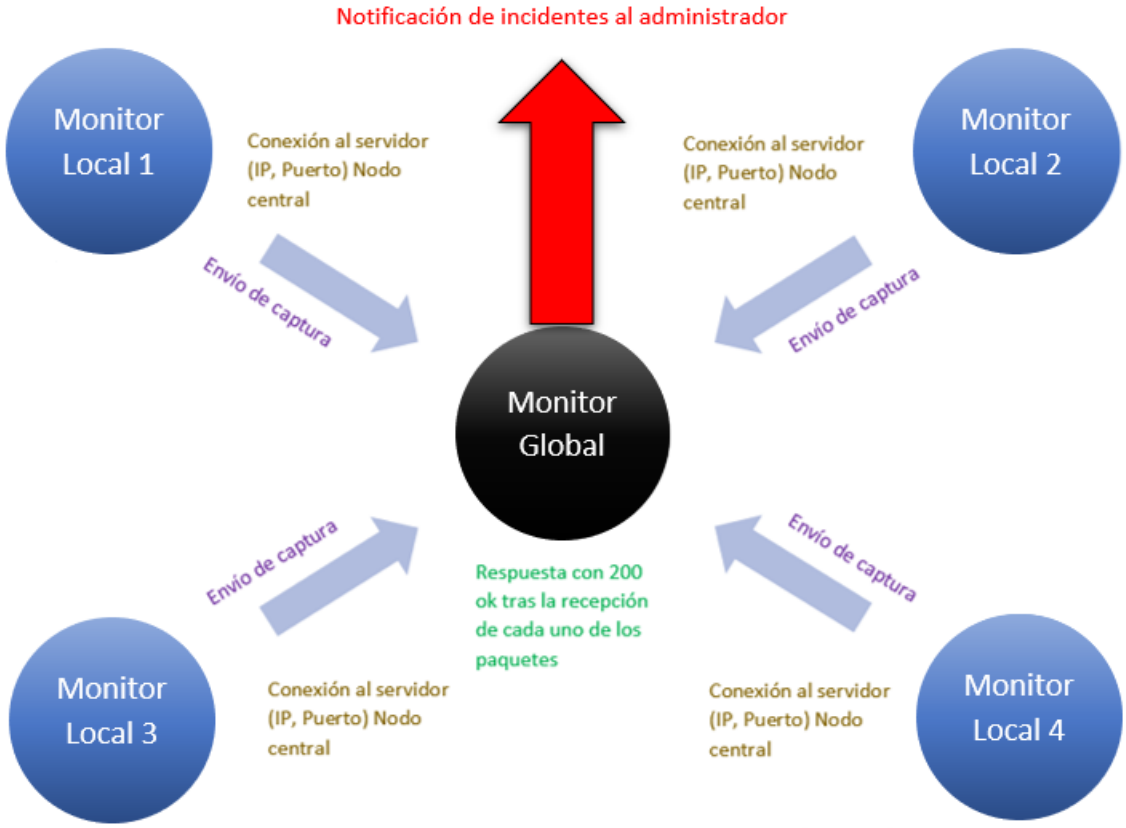

Ilustración 15 Posible caso de uso

El modelo de inteligencia artificial encargado del procesamiento, presente tanto en el ML como en el MG, requiere un poco más de atención.

Este modelo también se divide en varias partes. La primera de ellas se encargará de la preparación de los datos para el entrenamiento del modelo. En este primer punto deberemos preparar ficheros con las dos posibles estructuras para vectorizarlos y pasárselos al modelo utilizado.

Una vez preparados los datos debemos etiquetarlos para poder entrenar posteriormente al modelo, de manera que deberemos tener como atributos las clases con los nombres de los ataques en una lista.

El tercer bloque de este apartado será la parte correspondiente al entrenamiento de la red neuronal, donde incluiremos como atributo principal los elementos de entrenamiento y la cantidad de iteraciones que nuestro modelo deberá realizar para entrenarse.

El cuarto y último bloque consta de la parte de verificación del funcionamiento del modelo con una serie de gráficas en las que podremos observar si el comportamiento es el esperado.

En la siguiente ilustración encontraremos un resumen de este bloque de diseño.

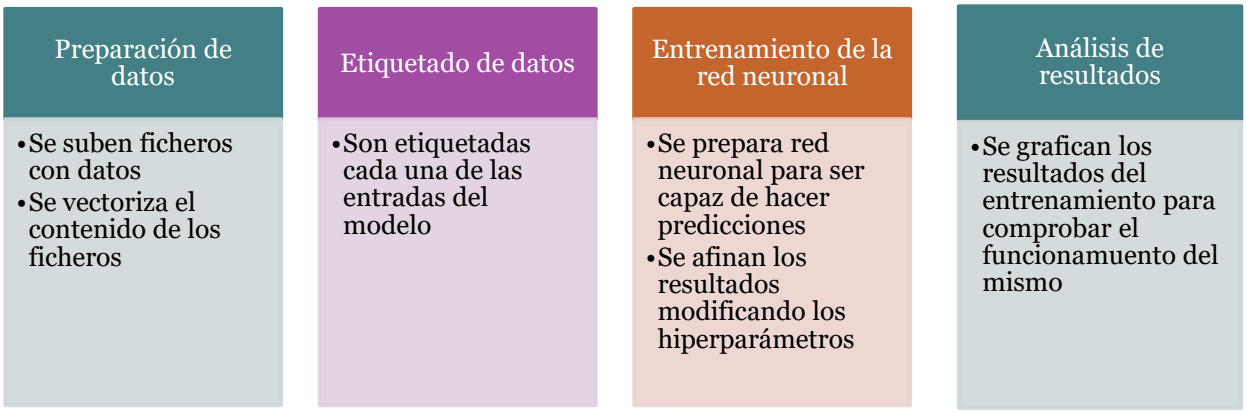

Ilustración 16 Resumen creación IA

5. Capítulo 5

5.2. Tecnología utilizada

Dada la densidad de este apartado es conveniente dedicarle un capítulo propio ya que, aunque finalmente haya sido utilizado un único lenguaje de programación se ha realizado un análisis exhaustivo de cuáles eran las posibilidades y finalmente hemos llegado a esa conclusión. Lo hemos repartido en tres bloques fundamentales para facilitar la comprensión del lector.

En el primer bloque veremos las tecnologías empleadas en la creación de los monitores, en el segundo la creación del modelo de Deep Learning y por último veremos la creación del entorno. Aunque el modelo vaya a ser empleado dentro de los monitores entendemos la tecnología a emplear es la misma.

5.2.1. Implementación de monitores

En cuanto al primer bloque, los monitores, no había ningún problema con los lenguajes de programación, simplemente necesitamos poner la tarjeta de red en modo monitor y ponerla a funcionar. Por tanto, primeramente, fueron programados en java, ya que es el lenguaje de programación en el que más experiencia tengo.

Por continuar en esa línea decidimos implementar el nodo central en java de la misma manera, con los sockets correspondientes para la recepción de la información y el envío de mensajes de confirmación. Todo ello mediante la librería de Pcap4J [25].

No obstante, cuando pasamos al análisis de los datos que se recibían y se enviaban, vimos que había muchos paquetes de los cifrados con TLS que no aparecían e hicimos pruebas con IPv6 y tampoco fueron satisfactorias, esto hizo que nos planteáramos el uso de librerías diferentes o la construcción de las mismas desde cero. Investigamos otras posibilidades y observamos que aún con esas limitaciones ésta era la librería más potente para trabajar con sockets y java. Y dado que el contenido fundamental del TFG no era únicamente la creación del IDS, sino también la investigación en las posibilidades del cambio de paradigma decidimos considerar el cambio en el lenguaje de programación.

El siguiente paso fue hacer una comparativa de diferentes tipos de lenguajes que había utilizado alguna vez para ver cuál iba a ser el más sencillo para trabajar a nivel general y poder ejecutarlo posteriormente en un computador monoplaca.

Python	R	Java
Desarrollo rápido	Gran potencia para análisis de datos	Mucha documentación
Código muy legible	Fácil configuración de paquetes	Fuente abierta
Escritura dinámica	Altamente eficiente	Hilos nativos
Tensor Flow y Keras	Gran curva de aprendizaje	Fácil de leer
Rendimiento poco eficiente	Desarrollo lento	IA muy compleja
Sin funciones anónimas	Altas necesidades tecnológicas	Pocas librerías para IA
Estructura del código	Sintaxis muy desordenada	Poca documentación para IA

Tabla 5 Comparación de lenguajes

Tras mucho analizar y viendo que con Java estaba teniendo problemas, finalmente dudé entre R y Python. Para la implementación del modelo hice un primer curso en R para ver cómo de complicado iba a resultarme su implementación y el desarrollo de la solución y vi que en el tiempo que disponía no iba a ser posible la creación de esta.

Por tanto, abrí la ventana del trabajo en Python, me sorprendió la cantidad de ejemplos que había y la gran comunidad que estaba trabajando con él. Viendo esto y habiendo hecho un curso de unas 15 horas que me introdujera lo suficiente a su entendimiento como para poder utilizarlo, finalmente me decanté en seleccionarlo como único lenguaje de programación para el proyecto.

El primer paso fue seleccionar la versión con la que quería trabajar y pese a que está aún muy dividido entre las versiones 2 y 3, finalmente me decanté por la 3 ya que está más actualizada y las principales librerías que pensaba utilizar daban menos problemas con esta.

Para evitar las limitaciones de hardware, decidimos enviar todos los paquetes en formato texto lo que supone un coste computacional mínimo para el sistema.

El siguiente aspecto que debía tener en cuenta es que durante la ejecución del programa debía controlar varios hilos de ejecución, por tanto, investigué que Python contaba con una librería que los controlaba de forma transparente para el programador y, por tanto, incluyendo la librería threads, mi aplicación contaba con un hilo que controlaba el monitor de datos y el envío de estos y otro hilo controla el estado de sobrecarga de la red e indica de qué manera deben enviarse los datos.

Comencé con el desarrollo de los monitores y aunque se podía implementar también de forma sencilla sin librerías, decidí emplear una llamada “scapy” [26]. Esta simplifica en gran medida todo el trabajo, además el modo de procesamiento de los paquetes es muy cómodo.

El siguiente elemento que debíamos crear era el monitor global. Las librerías de éste son básicamente las mismas, ya que, debemos procesar los paquetes y esto lo haremos con la misma librería “scapy”.

5.2.2. Implementación modelo IA

Otro elemento fundamental que hizo que abandonáramos el uso de Java es lo complejo que es crear y utilizar modelos de Deep Learning y más aún preprocesar la información para poder introducírsela a los modelos.

Como se ha mencionado en el apartado 2.1.2, la herramienta más adecuada para la implementación de la red neuronal será TensorFlow. Para ello, deberemos pasarle los datos preprocesados al modelo, en cuanto al preprocesamiento, será realizado por el paquete “textVectorization” [27] de la librería TensorFlow [28].

Además, para poder cargar el modelo preentrenado y utilizarlo para medir la fiabilidad de los paquetes, usaremos todo el pack de librerías de TensorFlow y de Keras [29].

El elemento final que incluimos en este trabajo es el obtenido del modelo de Deep Learning de las librerías de TensorFlow, ya que como hemos visto en la tabla 5 son las más fáciles y cómodas de programar. Dentro de la creación del modelo hemos necesitado también otras librerías como pueda ser “matplotlib.pyplot” [30] para graficar la efectividad del modelo o librerías como “os” que nos han permitido acceder al directorio de ficheros para entrenar al modelo.

5.2.3. Creación del entorno de simulación

Para el desarrollo del entorno de pruebas hemos empleado una serie de máquinas virtuales con Windows XP y Kali Linux que nos han permitido simular un sistema funcional que posteriormente explicaremos en detalle. Hemos empleado Kali Linux por ser una plataforma sobre la que resulta muy sencillo montar las pruebas de los ataques y Windows XP por lo ligero que es y la cantidad significativa de brechas de seguridad que tiene.

El último aspecto del que debemos hablar en cuanto a las tecnologías utilizadas son los entornos de desarrollo.

Por un lado, el proyecto lo inicié en mi ordenador personal, pero este cuenta con unas características limitantes para ciertos aspectos de Deep Learning, por tanto, todos los elementos que han sido desarrollados en mi equipo han sido realizados con el IDE PyCharm, lo que en gran medida me ha facilitado mucho su programación y son ejecutados desde la consola de la máquina virtual con Kali Linux.

Para toda la parte del modelo de Deep Learning tomamos la decisión de que iba a ser mucho más sencillo trabajar con computación en la nube y de forma gratuita para estudiantes los entornos que más cómodos me resultaron fueron Google Colab e IBM Watson.

Tras un poco de investigación sobre ambas herramientas encontré que había mucha más información y documentación para Colab. Además, el proyecto TensorFlow, que es la base de nuestro sistema, cuenta con el apoyo de Google, lo que hace que su sistema esté preparado para trabajar en este entorno con gran eficiencia y comodidad.

Además, es importante destacar que cuenta con una ejecución aún más rápida implementando aceleración por GPU o aceleración por TPU, cosa que para este tipo de proyectos es altamente recomendable. No obstante, dado lo reducida que es nuestra aproximación de cara al modelo de sistema que se pretende crear, con la aceleración GPU fue suficiente para completar el entrenamiento de nuestro modelo. El uso de estas tecnologías resultará necesario para conjuntos de datos más grandes.

6. Capítulo 6

6.1. Desarrollo de la solución

Partiendo del diseño final que podemos encontrar en la ilustración 10 y conociendo las tecnologías que vamos a utilizar y los aspectos del problema que pretendemos resolver con ellas podemos comenzar con su desarrollo.

En los siguientes apartados encontraremos el desarrollo completo de la solución desde los diferentes elementos con los que esta cuenta: el monitor local, el monitor global y el modelo de inteligencia artificial.

Finalmente, la etapa de integración permitirá que, aunque los vayamos a desarrollar de forma independiente, los tres elementos funcionen simultáneamente y trabajen como un único sistema.

6.1.1. Desarrollo del monitor local

Como hemos visto en capítulos anteriores y podemos revisar en la ilustración 4, este elemento está constituido por diversos elementos.

El recolector de información es la base sobre la que se construye este sistema. Para construirlo, mediante la función Sniff del paquete "Scapy", y dando al software derechos de administrador, obtenemos el resultado de la monitorización de la red. Estos datos serán posteriormente almacenados en una variable sobre la que trabajaremos para su procesamiento.

Sobre esta función es necesario comentar que tiene un parámetro que permite, proporcionar una función a invocar por cada paquete recibido. No obstante, como para nosotros es más interesante trabajar con un grupo de paquetes, emplearemos el parámetro *count*, que permite agrupar el tráfico por número de paquetes recibidos.

En la siguiente ilustración podemos observar cual será el resultado de esta orden:

```
>>> s = sniff(count = 3)
>>> s
<Sniffed: TCP:0 UDP:3 ICMP:0 Other:0>
>>>
```

Ilustración 17 Ejemplo de Sniff

Una vez almacenados los datos en una variable, la librería scapy crea, por cada paquete, varias listas con bloques de información, es decir, subdivide el paquete en bloques de información elementales como podemos ver en la siguiente ilustración.

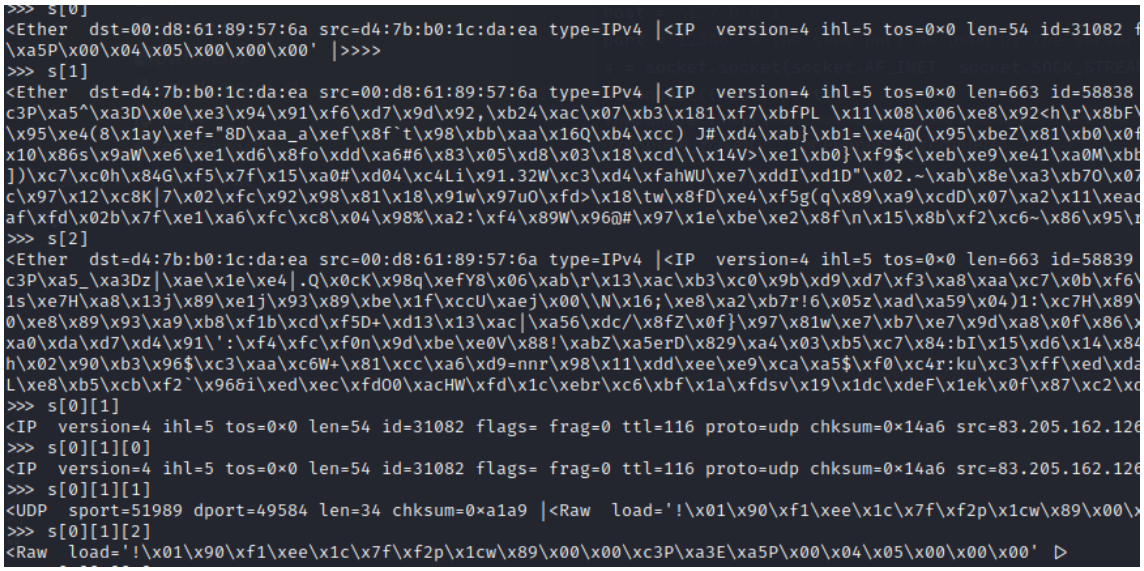

Ilustración 18 Contenido de la captura

Para acceder a toda la información únicamente deberemos recorrerla y transformarla, con el objetivo de usarla cómodamente, en una lista de textos, siendo cada texto un paquete diferente, de manera que alimentaremos posteriormente al modelo con todos los datos en forma de texto, lo que facilitará en gran medida su utilización.

Debemos tener en cuenta que nuestro sistema debe analizar diferentes segmentos de red, y la comunicación entre ML y MG se realiza sobre la misma red monitorizada. Por tanto, es posible que un ML tenga acceso a los paquetes que otro ML le está enviando al MG. Para no incluir estos paquetes en la traza, siempre que sean observados paquetes cuya dirección y puerto destino corresponda con el monitor global serán automáticamente omitidos.

Como ya hemos mencionado anteriormente, debemos contar con diversas líneas de ejecución. Para ello, al principio del programa lanzaremos varios hilos que cada uno realizará uno de los cometidos mencionados; Captura de paquetes, evaluación de carga, análisis de paquetes y finalmente comunicaciones. Para esto, emplearemos la librería de Python “threads” [31].

Para calcular el porcentaje de utilización de la red, en esta aproximación supondremos que nuestro adaptador tiene la capacidad de capturar el 100% de los paquetes. Teniendo esto en cuenta, de nuevo con la función Sniff, pero esta vez con el parámetro timeout, calculamos el tamaño total de los paquetes recibidos en un segundo y lo comparamos con la velocidad de transmisión de la red. En caso de que al dividirlos tengamos un valor mayor que 0.3 no deberemos enviar los paquetes completos, ya que, en el peor de los casos esto implicaría triplicar la carga de la red (al codificar los paquetes recibidos en texto plano, los paquetes enviados ocupan más que los originales) y con ello congestionaríamos la red, de manera que debemos preprocesarlos y enviar solamente la información fundamental.

En caso de querer afinar más el porcentaje de utilización de la red, podríamos realizarlo analizando el tiempo entre la transmisión de un paquete y su confirmación (Round Trip Time). No obstante, para el nivel al que se pretende llegar en este trabajo no es necesario.

Hecho esto, llegamos al detalle del tercer punto, la identificación de dispositivos y la creación de conexiones para el envío de paquetes. La identificación del monitor global por parte de cada monitor local se consigue mediante el siguiente mecanismo "Plug&Play": Cuando el monitor local se encienda por primera vez, o no obtenga respuesta de su monitor global, enviará una difusión IP sobre la red dirigida al puerto 12347, identificándose. El monitor global, que estará escuchando dicho puerto, le responderá con un datagrama IP dirigido indicándole que ha sido detectado. A partir de este instante, el monitor local conoce la IP del monitor global para posteriormente enviarle los paquetes capturados.

El otro punto a comentar es la conexión con el nodo central. Para ello se establece una conexión TCP entre ML y MG, y por ella le enviamos los paquetes con las capturas realizadas. Para esto, debemos tener en cuenta que primeramente debemos pasar el contenido de los paquetes (ya sea preprocesado o no) a bytes. Adicionalmente, se delimita el principio y el fin de cada paquete extraído de nuestra lista mediante el símbolo almohadilla ('#'), ya que no aparece nunca en los datos devueltos por la librería "scapy".

En la ilustración siguiente podemos ver un resumen del funcionamiento del monitor local. Podemos observar con claridad las tareas que se realizan simultáneamente y las que deben realizarse como comprobación inicial:

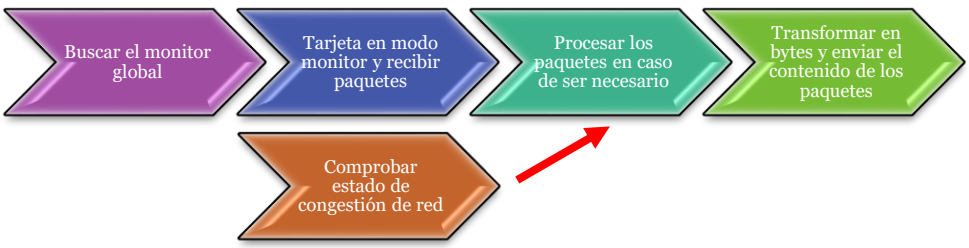

Ilustración 19 Resumen implementación de monitor local

6.1.2. Desarrollo del monitor global

El siguiente desarrollo a detallar es el correspondiente al monitor global. Éste debe recibir los paquetes de todos los monitores locales, de manera que necesitaremos implementar un servidor concurrente capaz de recibir las peticiones de conexión, recoger y almacenar la información de las trazas y contestar a las peticiones con el código “200 ok”, para que el monitor local pueda verificar que el contenido ha sido recibido correctamente.

Los datos recibidos desde el monitor local están expresados como bytes. Deben transformarse nuevamente en texto para almacenarlo en una lista, con un formato igual al que tenían inicialmente.

Simultáneamente a esto encontramos otro hilo de ejecución que, cuando tengamos los paquetes suficientes para realizar los análisis de los datos, los almacenará en una lista auxiliar y dejará la lista principal, donde guardábamos los paquetes nada más transformar los recibidos, completamente vacía.

Las listas que utilicemos deberán ser circulares, de forma que el hilo encargado de la recepción insertará paquetes y el hilo de análisis los extraerá, de esta manera evitamos la perdida de información.

Llegados a este punto, es interesante obtener una imagen de la comunicación entre los nodos para poder hacernos una idea de su funcionamiento. Esto lo podemos observar en la siguiente ilustración.

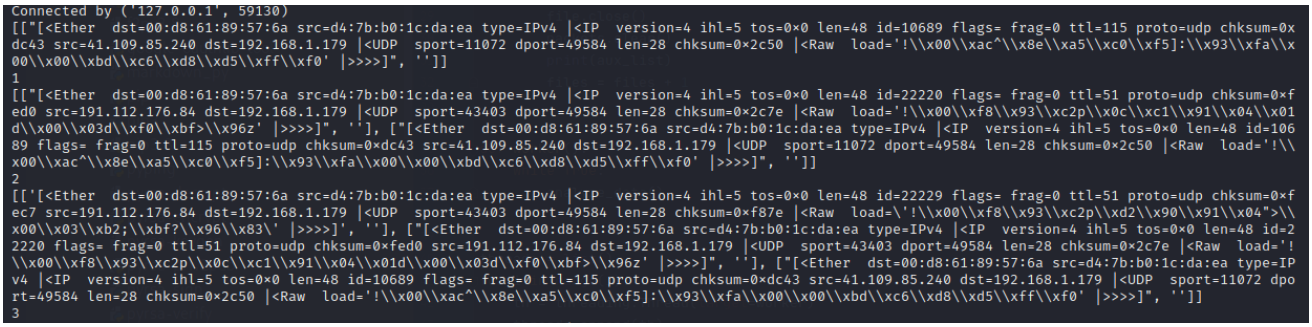

Ilustración 20 Ejemplo recepción nodo central.

En la ilustración podemos encontrar que, primeramente, el monitor local que se ha conectado a nuestro monitor global, y la lista de paquetes que éste ha monitorizado, con un contador que nos permite identificar cuántos han sido capturados. De esta manera, observamos el contenido de todos los paquetes que han sido entregados hasta el momento. Esto lo hacemos con el fin de que podamos verificar por consola, cuando lleguemos al número de paquetes definido para el análisis, en este caso 250, el contenido de lo que va a ser analizado.

Tal y como hemos visto anteriormente, identificamos los paquetes que nos entran por el carácter almohadilla que hemos añadido en el ML, lo que nos facilita el almacenamiento en la lista de la forma adecuada.

Como hemos mencionado anteriormente, el módulo de IA espera un formato de paquete basado en texto. Para ello deberemos transformarlo todo a texto antes de empezar a utilizar la red neuronal. Este módulo se estudia con más detalle posteriormente.

El siguiente paso, por tanto, será, cargar el modelo entrenado usando las librerías de TensorFlow.

Una vez cargado, deberemos pasarle los datos en el mismo formato en el que los hemos generado para entrenarlo. Para ello, la librería utilizada, Tensor Flow, requiere que transformemos este texto en vectores mediante la librería "TextVectorization", de manera que cada grupo de caracteres se pueda relacionar con un número y el estudio de los patrones se simplifique en gran medida.

La respuesta de la red neuronal nos indica, con un porcentaje de probabilidad, si entre las tramas suministradas se detecta algún tipo de ataque de entre aquellos para los cuales la hemos entrenado. Si el porcentaje es muy bajo, se asume que se trata de tráfico habitual no malicioso.

En el primer caso, se informará al administrador del sistema mostrando un mensaje de alerta por consola indicando el porcentaje de riesgo, de forma que éste pueda actuar para solucionar el problema antes de que sea más grave. En futuras versiones se implementarán otros métodos de alerta (correos electrónicos, mensajes a móviles, como SMS, Telegram, etc.) e incluso acciones defensivas como el bloqueo del tráfico sospechoso por un cortafuegos adicional.

En la siguiente ilustración encontraremos un resumen del funcionamiento de este elemento.

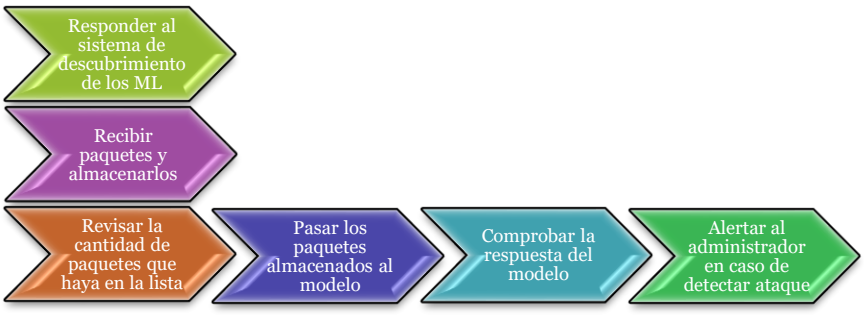

Ilustración 21 Resumen Monitor global

6.1.3. Desarrollo del Modelo de IA

El último bloque que debemos analizar de la solución que planteamos es el del desarrollo del modelo de inteligencia artificial. Durante la especificación del mismo explicaremos de qué manera obtenemos la información para entrenar al modelo y utilizarlo.

6.1.3.1. Capturas y procesado

Para la generación de los datos utilizaremos el conjunto de máquinas virtuales que se describe con detalle en el apartado de pruebas. Sobre el mismo, empleando herramientas de software abierto, realizaremos ataques del tipo Men in the Middle y del tipo Port Scan, y realizaremos capturas de los mismos. También se realizarán capturas de tráfico normal, sin ningún ataque, que posteriormente nos permitan verificar el modelo en un caso real. Estas capturas se emplearán para entrenar al modelo.

Para someter al modelo a situaciones reales, donde el tráfico correspondiente a un ataque se superpone al tráfico habitual, se realizarán también capturas en estas circunstancias. Estas capturas no se utilizan para entrenar al modelo, sino para su verificación posterior.

Las capturas correspondientes a cada uno de los ataques contemplados se almacena un directorio en ficheros .txt, montando de esta forma una estructura altamente escalable para generar más tipos de ataque.

Este directorio será comprimido de manera que no sea necesario acceder a los ficheros de forma individual y lo almacenaremos en una variable del tipo keras.file, en la cual descomprimiremos el directorio para acceder a los datos de forma individual .

Abrimos un fichero de prueba y así vemos que lo hemos guardado todo correctamente y que podemos acceder a los ficheros correctamente. El resultado debería ser algo similar a lo que podemos ver en la siguiente ilustración:

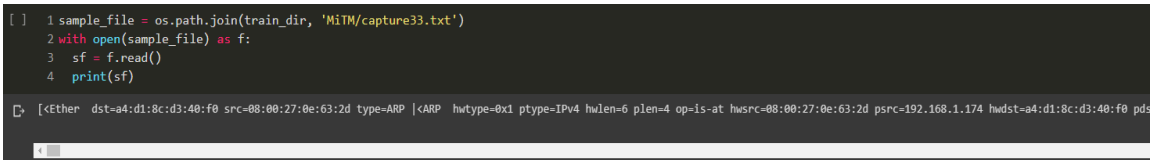

Ilustración 22 Fichero de ejemplo abierto

Como podemos ver, encontramos una captura de 250 paquetes abierta en formato texto en una única línea, esto será lo que posteriormente queramos analizar con el modelo y verificar si se detecta algún elemento peligroso.

En este punto explicaremos también el procesado de los datos necesario para que estos puedan ser entendidos por el modelo.

Para ello debemos definir la vectorización, ya que, como hemos mencionado antes, todos los elementos que le pasemos al modelo deberán ser pasados a vectores numéricos, transformando el texto en números. Este procedimiento analizará todo el fichero y transformará todos los conjuntos de caracteres en valores numéricos, de forma que cada uno tenga un valor asignado, de esta forma nuestro texto se convertirá en una matriz numérica con un diccionario de traducción asignado.

Para la vectorización definiremos un tamaño máximo para las matrices que nos indicará el tamaño máximo de palabras para cada fichero. Hemos seleccionado 10000, suponiendo que es un valor suficientemente grande para cualquier fichero. Es preferible tener ceros en las matrices donde no tengamos datos a dejar información fuera de la matriz. El siguiente paso será mapear los datos de entrenamiento eliminando las etiquetas, para posteriormente adaptarlo para vectorizarlo. Hecho esto, creamos una función a la que le pasaremos el texto y su etiqueta y nos la vectorizará para que podamos pasársela al modelo.

Comprobaremos que se ha vectorizado correctamente y que se nos ha generado un diccionario acorde a nuestras necesidades, tal y como podemos ver en la siguiente ilustración.

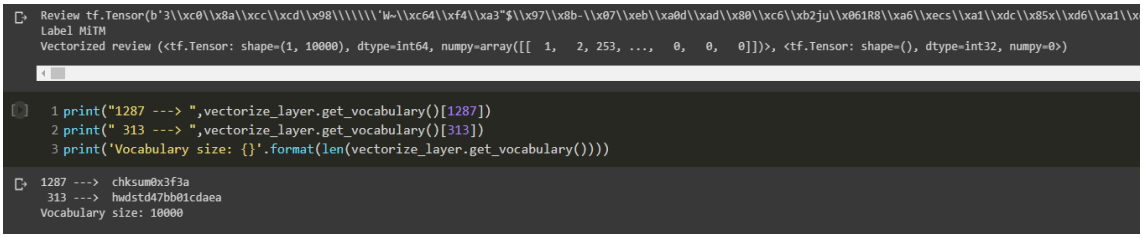

Ilustración 23 Ejemplo de vectorización y diccionario

Hecho esto, nuevamente volvemos a mapear los datos que hemos obtenido antes (los de entrenamiento, validación y testeo), con el texto vectorizado. De esta manera ya tendremos todos los datos preparados para la utilización de los mismos en la red neuronal.

En cuanto al contenido de los ficheros con los que entrenaremos y validaremos el modelo es importante remarcar que contienen tanto capturas donde únicamente tenemos los ataques que hemos preparado como capturas que incluyen estos ataques

mezclados con tráfico de la red, de manera que facilitamos el aprendizaje, pero contamos con ejemplos suficientes para verificar su funcionalidad.

6.1.3.2. Entrenamiento de la red

El siguiente paso será definir los datos de entrenamiento, para ello contamos con una función que nos permite extraer los ficheros de un directorio y seleccionarlos para ser preprocesados, en nuestro caso, para la parte correspondiente al entrenamiento emplearemos únicamente 24 ficheros de todos los que hemos creado.

Hecho esto, repetiremos el mismo proceso para los datos de validación, pero en este caso, elegiremos una mayor cantidad de ficheros para asegurarnos de que vayan a haber ficheros diferentes que en la de entrenamiento. Elegiremos 36.

No elegimos la totalidad de los ficheros en ambos casos para evitar que el modelo aparente funcionar pero que realmente no sea operativo. Esto podría ocurrir en el caso de que haya memorizado una secuencia o una serie de patrones de forma exacta y no es capaz de reconocer patrones más allá de los preentrenados. De esta forma, reservamos capturas para verificación distintas de las de entrenamiento.

En cuanto a la selección de los datos, generaremos una última variable, con todos los ficheros, pero esta vez los almacenaremos todos. Esta la utilizaremos posteriormente para testear nuestra red neuronal.

El siguiente paso es optimizar los datos para ser analizados con la función de TensorFlow “Autotune” y finalmente pasamos a definir el modelo, esta función define el número apropiado de procesos que están libres para funcionar.

Una vez definido, podemos compilarlo, y para este tipo de análisis de datos suele utilizarse de forma más específica el optimizador “Adam” [32].

Tras hacer varias pruebas con el modelo y con el objetivo de lograr una probabilidad de acierto cercana a la unidad, se ha encontrado que son necesarias unas 500 iteraciones, empleando para ello los valores que previamente hemos definido para análisis y validación.

6.1.3.3. Validación del modelo

Una vez finalizamos el entrenamiento, con los datos de testing podremos verificar su funcionamiento y, sorprendentemente, tras el entrenamiento, obtenemos una

probabilidad de acierto del 1 lo cual nos hace pensar que aparentemente nuestro modelo cumple sus expectativas.

Para comprobar esto, gracias a la librería "plt" graficamos los datos, estos los incluimos en las siguientes ilustraciones, y los analizamos para ver si efectivamente hemos obtenido el resultado esperado.

En la siguiente ilustración, en el eje x encontramos la cantidad de iteraciones que va a realizar el modelo durante el entrenamiento y en el eje y una tasa de error acumulativa, de esta manera podemos observar que es lo que ocurre en cada una de las iteraciones.

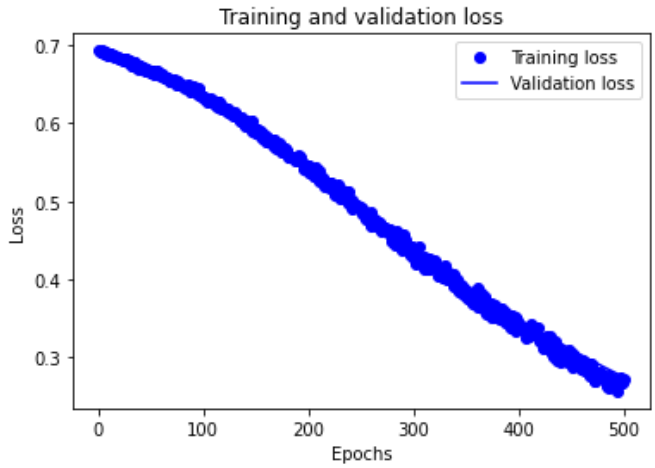

Ilustración 24 Gráfica de fallos del modelo

Tal y como esperamos por lo que hemos visto anteriormente, el número de errores se ve significativamente reducido al llegar a 500 iteraciones y, además, la línea de validación es aproximadamente la misma que genera la de entrenamiento. El hecho que tengamos una tasa diferente a 0 es que, al ser un valor acumulativo, aunque acierte en todos los casos reducimos significativamente la tasa de error.

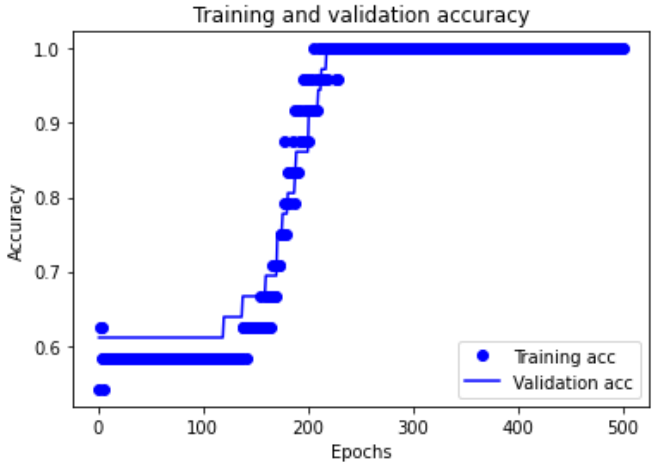

Ilustración 25 Gráfica de aciertos del modelo

En la gráfica anterior, contamos con una estructura similar a la ya explicada con la diferencia que analizamos la precisión. Podemos ver que el número de aciertos a partir

de 200 iteraciones es correcto, ya que es el acertamos en todas las ocasiones. Por tanto, si analizamos la información de las dos gráficas en conjunto podemos ver que nuestro modelo converge y que por tanto puede ser utilizado en nuestro sistema.

Además, un detalle importante a observar es que con los datos de entrenamiento y con los datos de validación obtenemos unos resultados muy similares y sabiendo que en los datos de validación hemos incluido ficheros que no teníamos en los datos de entrenamiento esto también nos da pistas de que el modelo es viable.

Para finalizar, en la siguiente ilustración podemos observar un resumen del funcionamiento del modelo de inteligencia artificial.

Ilustración 26 Resumen creación del modelo de IA

7. Capítulo 7

7.1. Implantación del producto

Nuestro sistema no ha sido implantado aún. No obstante, se han tenido en cuenta una serie de consideraciones que hacen que sea viable y pueda ser utilizado en un entorno real.

El primer factor ha sido la similitud de las máquinas virtuales con los dispositivos hardware que van a ser empleados en un entorno real. También se ha tenido en cuenta que las tecnologías empleadas funcionen en estos sistemas empotrados, de manera que estos dispositivos pueden emplearse de una manera funcional.

Además, como hemos visto en anteriores apartados se ha reducido la carga computacional de algunos procesos para evitar que este Hardware sea insuficiente.

Debemos observar también que el planteamiento contempla el funcionamiento real del IDS, como la posibilidad de que varios monitores compartan tráfico, y se ha estudiado su casuística y se han implementado las soluciones adecuadas.

El siguiente aspecto que se ha tenido en cuenta es el protocolo de automatización transparente al usuario de conexión de los monitores locales con el monitor global, de manera que no es necesaria la realización de ninguna modificación sobre el código fuente para que esto sea funcional.

En cuanto al proceso de la instalación del sistema en una red, es algo realmente sencillo, ya que únicamente es necesario a) identificar los segmentos de red que se desean monitorizar; b) instalar un monitor local conectándole el cable de red en el punto a observar; y c) acceder a la configuración del monitor global y definir el lugar donde deben enviarse las alertas. Tras estos pasos, el sistema estará operativo y listo para defender de las amenazas.

8. Capítulo 8

En este apartado vamos a explicar todas las pruebas que hemos realizado con todos los componentes, con el sistema, con el modelo y con el nodo central.

8.1. Entorno de pruebas

Lo primero que debemos comentar es la creación del entorno de pruebas propiamente dicho. Este entorno ha sido necesario para simular una red con las características necesarias y obtener capturas reales para tanto para el entrenamiento como para la verificación del IDS.

Para ello instalamos la aplicación VirtualBox y creamos una serie de máquinas virtuales, dos con Kali Linux y otras cuatro con Windows XP, como anteriormente se ha decidido en el apartado de tecnologías.

Hecho esto, modificamos la configuración del adaptador de donde vamos a instalar un *sniffer* para que éste pueda capturar tanto el tráfico de dentro de la red virtual como aquél fuera de la misma. De este modo, la captura obtenida contendrá tanto tráfico real como el correspondiente a los ataques que se generarán en la red de máquinas virtuales.

Para ello, en el apartado de red dentro del panel de administración de VirtualBox indicamos que queremos que la tarjeta de red funcione en modo puente y que además se permita todo el tráfico a través del modo promiscuo.

En la siguiente ilustración podemos ver como debe ser esta configuración:

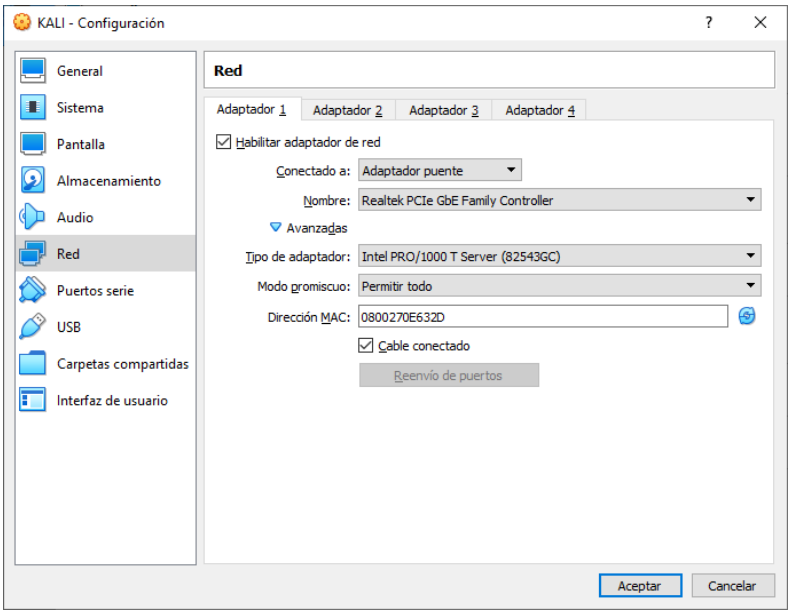

Ilustración 27 Configuración tarjeta de red Sniffer

Esta será la única máquina que tendrá esas. El resto mantendrán la configuración por defecto trabajando en modo NAT y con el modo promiscuo denegado ya que no necesitan escuchar el tráfico de la red externa a las máquinas virtuales. La configuración podemos verla en la siguiente ilustración.

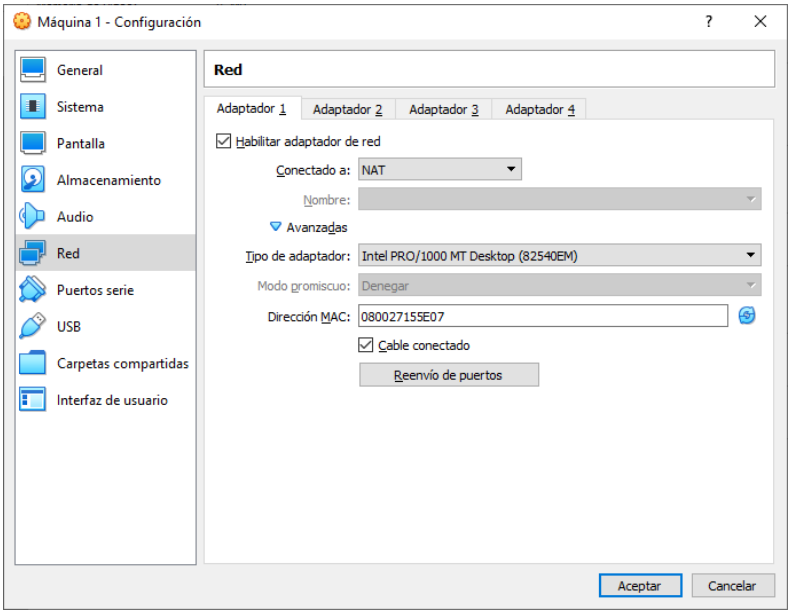

Ilustración 28 Configuración red máquinas restantes

Una vez construido el entorno de pruebas, se han realizado modificaciones sobre éste para capturar únicamente paquetes que correspondieran a los ataques a simular. Al tener el control absoluto sobre la red, podemos crear bloques de paquetes asociados a los ataques, de forma que, a la hora de entrenar la red neuronal, le sea mucho más sencillo caracterizar los mismos.

8.2. Pruebas del producto

Las siguientes pruebas a destacar se centran en el ML, que por comodidad empleará la librería “scapy”.

Una vez terminados el MG y el ML, imprimiendo los formatos de las capturas por consola y observando la carga de la red y los envíos de paquetes, el resultado que obtenemos se observa en las siguientes ilustraciones. La configuración considera bloques de 250 paquetes para su análisis y procesado.

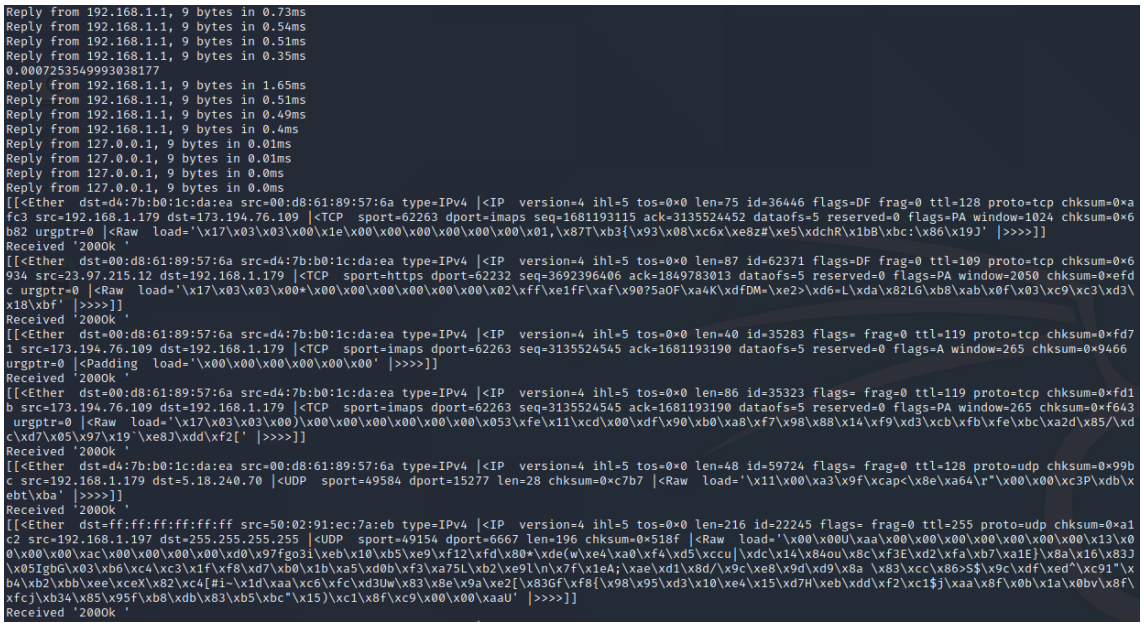

Ilustración 29 Entorno de pruebas sniffer

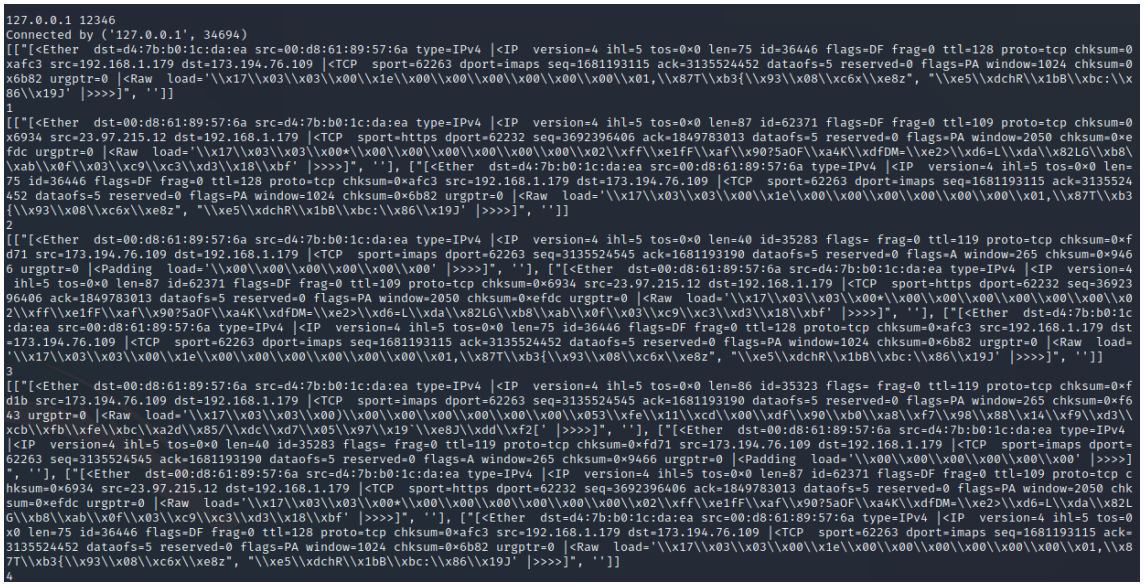

Ilustración 30 Entorno de pruebas nodo central

El siguiente paso es observar cómo enviamos cada uno de los paquetes por TCP al monitor global y son mostrados por consola.

Además, podemos comprobar también que cada vez que un paquete es enviado el nodo nos responde con el código “200 ok”, que indica que ha sido recibido e interpretado correctamente.

Por último, es conveniente considerar que el monitor global cuenta el número de paquetes recibidos, de forma que cada 250 paquetes se procesen adecuadamente. En paralelo, en el ML realizamos cálculos de la utilización de la red. En el momento de la ejecución del comando es de 0,07% de congestión, lo que nos indica que no existe prácticamente tráfico en la red.

8.3. Comparativa otros productos.

Teniendo desarrollado el sistema que se plantea en este TFG ya podemos realizar una comparativa con los sistemas que hemos investigado de entre todas las opciones que podemos encontrar en el mercado.

Comenzamos la comparativa con Snort. En la siguiente tabla podemos observar las ventajas que tiene el sistema que hemos desarrollado frente a Snort.

AHIDS	SNORT
Sistema híbrido	Sistema centralizado
Basado en Deep Learning	Basado en redes de reglas
Trabaja con % de fiabilidad	Trabaja con positivos o negativos
Falsos positivos muy reducidos	Falsos positivos abundantes
Altamente escalable	No es escalable
Multiplataforma (Requiere Python)	Plataformas limitadas

Tabla 6 Ventajas de AHIDS vs SNORT

El siguiente sistema con el que vamos a enfrentarlo es Suricata, en la siguiente tabla vemos los puntos en los que AHIDS se sitúa por delante del mismo.

AHIDS	Suricata
Sistema híbrido	Sistema centralizado
Basado en Deep Learning	Basado en redes de reglas
Trabaja con % de fiabilidad	Trabaja con positivos o negativos

Falsos positivos muy reducidos	Falsos positivos abundantes
Configuración sencilla	Difícil configuración

Tabla 7 Ventajas AHIDS vs Suricata

Por último, vamos a compararlo con el sistema completo OSSEC y en la siguiente tabla podemos ver en qué aspectos destaca nuestro sistema.

AHIDS	OSSEC
Basado en Deep Learning	Basado en redes de reglas
Trabaja con % de fiabilidad	Trabaja con positivos o negativos
Falsos positivos muy reducidos	Falsos positivos abundantes
Adaptable al tráfico de la red	Puede generar sobrecargas de red
Altamente escalable	No es escalable
No consume recursos de los equipos de la red	Consume recursos de todos los dispositivos de la red

Tabla 8 Ventajas AHIDS vs OSSEC

Como último aspecto de la comparativa, una vez hemos analizado en profundidad estos sistemas hemos realizado una calificación personal valorando las características de cada IDS.

Título	AHIDS	OSEC	Suricata	Snort
Escalabilidad	5	5	2	2
Complejidad	1	5	3	3
Eficiencia	3	1	5	3
Eficacia	5	4	3	3
Adaptabilidad	5	5	1	1

Tabla 9 Comparativa personal de sistemas

9. Conclusiones y trabajos futuros

9.1. Conclusiones

El primer elemento a estudiar en las conclusiones es el cumplimiento de los objetivos que han sido planteados. Para ello estudiaremos la solución en profundidad.

El objetivo principal ha sido logrado, ya que hemos implementado una solución hibrida, que emplea modelos de inteligencia artificial que hemos creado con TensorFlow en cada uno de los monitores.

En cuanto a los objetivos segundarios que se han planteado para la verificación del objetivo principal, tal y como podemos ver en todos los apartados de la memoria se ha realizado un análisis exhaustivo de la problemática y se ha indagado sobre las soluciones existentes e incluso las normativas legales que esta debe cumplir. Los ataques han sido caracterizados y ha sido explicada su forma de detección. Se ha realizado una implementación de los monitores basada en la estructura definida como necesaria, siguiendo el patrón de diseño que había sido planteado y se ha verificado el funcionamiento del sistema conjunto.

Si bien es cierto que para el desarrollo del trabajo no ha sido generada una gran base de datos con muchos casos de ataques, se considera suficiente para ejemplificar el funcionamiento del sistema complejo cuya línea de trabajo se pretende abrir.

En cuanto al modelo de inteligencia artificial, también es cierto que, en fases más avanzadas del mismo (por ejemplo, como producto final para comercializar), el conjunto de ataques y capturas de cada uno de ellos a emplear para el entrenamiento del modelo debe ser mucho mayor. De esta manera nos aseguraremos de que AHIDS realmente sea completamente funcional. Por tanto, un nuevo objetivo a plantear sería la creación de un mayor conjunto de capturas que permita tanto detectar más ataques como hacerlo de manera más fiable.

Otro aspecto que sería mejorable es la cantidad de pruebas que han sido realizadas para la verificación del sistema completo, ya que en caso de disponer de más tiempo se podría haber analizado más exhaustivamente el funcionamiento del sistema tal y como se ha desarrollado.

En cuanto al desarrollo de los monitores, no ha supuesto una tarea excesivamente complicada. Ha sido necesario aprender a utilizar este nuevo lenguaje de programación, pero ello no ha supuesto una carga excesiva. No obstante, la creación del modelo sí ha resultado costosa por la cantidad de conceptos y elementos que, aunque están bien documentados, resultan difíciles de comprender.

El principal error que se ha cometido en el proyecto ha sido tratar de realizar el desarrollo con las tecnologías que ya conocía. Esto produjo dificultades al inicio del

desarrollo ya que no se conseguía capturar todos los paquetes en la red con los monitores locales, puesto que las librerías disponibles no detectaban algunos paquetes (tipos ICMP o IPv6, por ejemplo. El estudio de la solución a este problema condujo a replantear en entorno de programación, cambiando a Python. Afortunadamente, este cambio fue decidido en las primeras fases del proyecto tras una investigación exhaustiva de cuál sería la mejor tecnología a utilizar.

Debo destacar que respecto al desarrollo de esta solución únicamente tenía conocimientos adecuados sobre las topologías de red y de la manera que quería diseñar las comunicaciones, pero no contaba con conocimientos sobre los análisis de paquetes, el lenguaje de programación utilizado ni sobre desarrollo de inteligencia artificial. Por tanto, la realización de este trabajo ha supuesto un importante aprendizaje en todos estos ámbitos.

Profesionalmente, el valor fundamental que he adquirido es la necesidad de una adecuada organización y planificación, ya que en algunas fases – afortunadamente tempranas – del proyecto se cometieron estimaciones erróneas en la planificación de tiempos que han conllevado retrasos apreciables.

9.2. Trabajos futuros

Primeramente, vamos a listar cuales son las ampliaciones posibles sobre el desarrollo del proyecto:

- Implantación de los módulos ML y MH sobre los dispositivos Hardware que se han planteado y realización de pruebas en entornos reales.
- Utilización de técnicas de compresión sobre los datos intercambiados entre ML y MG para minimizar la intrusión sobre la red observada. Ello requerirá contar con dispositivos que empleen un hardware más potente.
- Aumentar el conjunto de capturas en el entrenamiento del modelo de inteligencia artificial para detectar una mayor cantidad de ataques. Dotar al sistema de capacidad de reacción automática frente a amenazas (por ejemplo, actuando sobre un cortafuegos), convirtiéndolo por tanto en un IPS (Intrusion Prevention System)
- Envío de alertas al administrador de la red mediante sistemas más cómodos y en tiempo real (correo electrónico, SMS, mensajería móvil, notificaciones *push*, etc.).

Las líneas de desarrollo que se han abierto tras la realización de este proyecto son las siguientes:

- Creación de sistemas híbridos de detección de intrusiones.
- Implementación de monitores en dispositivos hardware de bajo coste.
- Estudio y categorización de ataques habituales sobre redes.
- Aplicación del aprendizaje profundo para detección de ataques.

10. Capítulo 10

10.1. Relación del trabajo con estudios

Al comentar la relación del proyecto con los estudios es importante destacar que todas las asignaturas cursadas han aportado una utilidad, por pequeña que sea al desarrollo del mismo. No obstante, en este apartado vamos a centrar las asignaturas que más relevantes han sido para su desarrollo.

Las asignaturas principales han sido todas las referentes a las redes de computadores (Redes, Sistemas de seguridad en red…), ya que es la que me ha abierto los ojos a la necesidad de la existencia de soluciones con estas características y sobre la que se asientan las bases del sistema desarrollado.

Por supuesto, todas las asignaturas que han implicado conocimientos de programación han sido fundamentales para permitirme el aprendizaje de un nuevo lenguaje de una manera rápida y eficaz.

En cuanto a la paralelización de muchos procesos y la creación del sistema híbrido no debo olvidar los conocimientos que me han aportado las asignaturas computación paralela y concurrencia y sistemas distribuidos, ya que sin ellas no habría sido capaz de diseñar una solución con estas características.

En cuanto a la planificación, gestión del tiempo, identificación de puntos importantes y herramientas necesarias para toda esta gestión, han sido muy importantes todas las asignaturas relacionadas con el sector empresarial. Éstas han hecho que fuera capaz de realizar una planificación adecuada del trabajo.

Para todo el tema de aspectos legales, además de las asignaturas de empresa, que también han aportado su granito de arena, la asignatura de deontología y profesionalismo me ha transmitido las bases para sustentar todos los apartados relacionados con el proyecto.

Finalmente, asignaturas como estadística y sistemas inteligentes son las que han hecho posible el entendimiento y la posibilidad de generar modelos de aprendizaje automático con TensorFlow, ya que me han aportado las bases de conocimiento necesarias para ello.

Esto no quita que el resto de las asignaturas no hayan sido de utilidad porque todas ellas me han aportado las base como ingeniero para desarrollar este sistema.

Respecto a las competencias transversales obtenidas gracias a la implicación de la UPV, destacamos por una parte la aplicación de pensamiento práctico, ya que han sido aplicados los conocimientos teóricos obtenidos en la carrera y han sido alcanzados unos objetivos determinados. Además, hemos resuelto un problema de una forma efectiva,

siendo capaces de identificar los elementos constituyentes. Hemos aplicado la creatividad para tratar de generar una idea de sistema de bajo coste que actualmente no se comercializa y además estamos emprendiendo ya que buscamos sacar un nuevo producto al mercado. Dadas las características del proyecto hemos aplicado la responsabilidad ética realizando análisis de toda la temática relacionada con la legislación. Hemos tratado de conseguir un aprendizaje permanente buscando emplear sistemas que no conocíamos y salir de la zona de confort y por último hemos empleado una instrumental específica que nos ha permitido resolver el problema.

11. Capítulo 11

11.1. Referencias

[1] ONU.(2020). Informe de los Objetivos de Desarrollo Sostenible. https://unstats.un.org/sdgs/report/2020/The-Sustainable-Development-Goals-Report-2020_Spanish.pdf . Último acceso en agosto de 2020.

[2] Alanazi, Hamdan & Md. Noor, Rafidah & Bahaa, Bilal & Zaidan, A. (2010). Intrusion Detection System: Overview.

[3] Hordri, Nur & Yuhaniz, Siti & Shamsuddin, Siti Mariyam. (2016). Deep Learning and Its Applications: A Review.

[4] https://www.snort.org/ . Último acceso en agosto de 2020.

[5] https://suricata-ids.org/ . Último acceso en agosto de 2020.

[6] https://www.ossec.net/ . Último acceso en agosto de 2020.

[7] Computer Recreations: Of Worms, Viruses and Core War" by A. K. Dewdney in scientific American, March 1989, pp 110.

[8] Prieto Meléndez, Rafael & Herrera, A & Pérez, J & Padrón-Godínez, Alejandro. (2000). EL MODELO NEURONAL DE McCULLOCH Y PITTS Interpretación Comparativa del Modelo.

[9] Thary, Hayder. (2019). Perceptron Learning. 10.13140/RG.2.2.12324.63366.

[10] Campbell, Murray & Hoane, A. Joseph & Hsu, Feng-hsiung. (2002). Deep blue. Artificial Intelligence. 134. 57-83. 10.1016/S0004-3702(01)00129-1.

[11] Betancourt, Gustavo. (2005). LAS MÁQUINAS DE SOPORTE VECTORIAL (SVMs). Scientia Et Technica.

[12] Serna M., Edgar & Serna, Alexei & Acevedo, Eder. (2017). Principios y características de las redes neuronales artificiales.

[13] Rodríguez-Mateos, Francisco & Bautista, Susana. (2006). Modelos ocultos de Markov para el análisis de patrones espaciales. Ecosistemas: Revista científica y técnica de ecología y medio ambiente, ISSN 1697-2473, Nº. 3, 2006. 15.

[14] Santana-Quintero, Luis & Coello, Carlos. (2006). Una introducción a la Computación Evolutiva y alguna de sus aplicaciones en Economía y Finanzas || An Introduction to Evolutionary Computation and some of its Applications in Economics and Finance. Revista de Métodos Cuantitativos para la Economía y la Empresa. 2.

[15] Cardenas, Marina & Medel, Ricardo & Castillo, Julio & Vazquez, Juan & Casco, Osvaldo. (2015). Modelos de aprendizaje supervisados: aplicaciones para la predicción de incendios forestales en la provincia de Córdoba.

[16] Quituisaca-Samaniego, Lilia & Álvarez, Harry. (2017). Aprendizaje no supervisado: agrupamiento o clustering. 10.13140/RG.2.2.19303.34724.

[17] Arango, Mauricio. (2019). Introducción al Aprendizaje por Refuerzo.

[18] Trabelsi, Zouheir. (2012). Switch's CAM table poisoning attack: hands-on lab exercises for network security education. 113-120.

[19] R. Droms (1997). Dynamic Host Configuration Protocol. rfc2131.

[20] David C. Plummer (1982). An Ethernet Address Resolution Protocol. rfc826.

[21] Google (2019). Panorama actual de la Ciberseguridad en España. https://www.ospi.es/ . Último acceso en agosto de 2020.

[22] Gartner (2018). Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019. Gartner.

[23] INCIBE. Resultados del Catálogo y Mapa de Conocimiento de la I+D+i en Ciberseguridad en España. INCIBE

[24] Consejo de Colegios de Ingeniería Informática. https://www.ccii.es/ejercicio-profesional-informatica/legislacion-informatica . Último acceso en agosto de 2020.

[25] https://www.pcap4j.org/ . Último acceso en agosto de 2020.

[26] https://scapy.net/ . Último acceso en agosto de 2020.

[27] https://www.tensorflow.org/api_docs/python/tf/keras/layers/experimental/preprocessing/TextVectorization . Último acceso en agosto de 2020.

[28] https://www.tensorflow.org/. Último acceso en agosto de 2020.

[29] https://keras.io/. Último acceso en agosto de 2020.

[30] https://matplotlib.org/api/pyplot_api.html . Último acceso en agosto de 2020.

[31] https://docs.python.org/es/3/library/threading.html . Último acceso en agosto de 2020.

[32] Diederik Kingma, Jimmy Ba (2014). Adam: A Method for Stochastic Optimization. arXiv/1412.6980.

[33] d'Anjou, Alicia & Graña, M. & Hernandez, C. & Torrealdea, Francisco. (2006). Computational experiments with Boltzmann Machines. 10.1007/BFb0035897.

11.2. Bibliografía adicional

1- Curso de aprendizaje automático de Google: https://developers.google.com/machine-learning/crash-course?hl=es-419
2- Documentación sobre Python: https://docs.python.org/es/3/ Último acceso en mayo de 2020.
3- Documentación VirtualBox: https://www.virtualbox.org/wiki/Documentation Último acceso en agosto de 2020.
4- Bases de datos de ataques catalogados: https://attack.mitre.org/ . Último acceso en agosto de 2020.
5- Bace, R., Mell, P.: "Intrusion Detection Systems". NIST Special Publication on Intrusion Detection System.
6- James Kurose & Keith Ross (2017). Computer Networking: A Top-Down Approach, 7th Edition.

11.3. Glosario

Aceleración GPU: técnica utilizada para acelerar las cargas de trabajo en procesos computacionalmente costosos mediante tarjetas gráficas.

Aceleración TPU: técnica utilizada para acelerar las cargas de trabajo en procesos computacionalmente costosos mediante circuitos integrados personalizados específicos de aplicaciones de Google.

ACK: En TCP, acuse de recibo de que ha sido correctamente recibido un segmento de red.

Algoritmo: Conjunto de operaciones que pretenden resolver un problema concreto.

ARP: *Address Resolution Protocol*. Protocolo de la capa de enlace de datos empleado para encontrar la dirección de hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

Beagle Board: Computadora monoplaca de bajo coste.

Botnet: Conjunto de máquinas comprometidas mediante robots informáticos o bots, que se ejecutan de manera autónoma y automática siendo controlados por el atacante que las haya infectado.

Concentrador (Hub): Dispositivo que permite centralizar el cableado de una red de computadoras.

Conmutador (Switch): Dispositivo digital lógico de interconexión a nivel de enlace de segmentos de red.

Creative Commons: Tipo de licencia donde el autor permite copiar, reproducir, distribuir, comunicar públicamente la obra y realizar obras derivadas (traducción, adaptación, etc.).

Deep Learning: Algoritmo automático que imita la percepción humana inspirada en nuestro cerebro y la conexión entre neuronas.

DHCP: *Dinamic Host Configuration Protocol*. Protocolo que permite la configuración dinámica de hosts, permitiéndole obtener tanto direcciones IP como opciones de configuración en una red.
Domotización: Conjunto de técnicas orientadas a automatizar una vivienda, que integran la tecnología en los sistemas de seguridad, gestión energética, bienestar o comunicaciones.

DoS: *Denial of Service*. Conjunto de técnicas utilizadas para inhabilitar un equipo o servicio.

Encaminador (Router): Dispositivo que permite la interconexión de redes locales con Internet

Formato RAW: Formato original en bruto, que contiene toda la información

FTP: *File Transfer Protocol*. Protocolo de aplicación para la transferencia de archivos entre sistemas conectados a una red TCP/IP.

HTTP: *Hyper Text Transfer Protocol*. Protocolo de aplicación para la transferencia de hipertextos entre sistemas conectados a una red TCP/IP.

ICMP: *Internet Message Control Protocol*. Protocolo utilizado para enviar mensajes de error e información operativa sobre IP.

IDS: Intrusion Detection System. Componente dentro del modelo de seguridad informática de una organización que pretende detectar actividades inapropiadas.

Inteligencia Artificial: Conjunto de algoritmos empleados para dotar de inteligencia a los sistemas informáticos.

IoT: *Internet of Things*. Ecosistema en el cual los objetos cotidianos interactúan entre sí mediante comunicaciones informáticas de forma transparente al usuario.

IPS: *Intrusion Prevention System*. Dispositivo que monitoriza el comportamiento de una red y puede realizar acciones para proteger a los sistemas computacionales de ataques y abusos.

IPv6: Versión 6 del protocolo IP.

MiTM: *Man in the Middle*. Conjunto de ataques empleados para comprometer los datos de la víctima haciendo que todos estos pasen a través del equipo del atacante.

Modelo: Conjunto de algoritmos empleados para lograr el aprendizaje de un sistema.

Monitor: Dispositivo encargado de controlar los flujos de información a los que tiene acceso.

MYSQL: Es un sistema de gestión de bases de datos relacional

ODS: Objetivos para el desarrollo sostenible definidos por las naciones unidas.

OpenSoruce: Modelo de desarrollo de software basado en la colaboración abierta.

PortScan: Conjunto de técnicas empleadas para realizar explorar los puertos de un sistema para comprobar cuáles son susceptibles de ser atacados.

Regla: Instrucción básica de evaluación de tráfico para considerar su peligrosidad.

RootKits: Conjunto de herramientas empleadas para lograr acceso de administrador a un sistema informático.

Rsyslog: Es un estándar de facto para el envío de mensajes de registro en una red informática IP.

Sniffer: Dispositivo utilizado para capturar todo el tráfico que circula a través de una red informática.

SSL: *Secure Sockets Layer*. Capa de encriptación que permite establecer conexiones seguras sobre una red TCP/IP.

TCP/IP: Pila de protocolos que permiten la comunicación entre los ordenadores en Internet.

TCP: *Transport Control Protocol.* Protocolo de comunicaciones fiables a nivel de transporte de la pila TCP/IP

TLS: Protocolos criptográficos que proporcionan comunicaciones seguras por una red.

Topología de red: Conjunto de elementos (cables, concentradores, conmutadores y encaminadores) que encontramos en el diseño de una red.