Universidad de Salamanca

TESIS DOCTORAL

Estrategias avanzadas de detección de la propagación del malware en redes IoT

Autor:
Marcos Severt Silva

Directores:
Prof. Dr. Ángel María Martín del Rey
Prof. Dr. Roberto Casado Vara

Salamanca, Julio, 2024

Declaración de Autoría

D. Marcos Severt Silva, presenta la tesis titulada ***Estrategias avanzadas de detección de la propagación del malware en redes IoT*** para optar al Grado de Doctor por la Universidad de Salamanca (programa de doctorado en Ingeniería Informática), y hace constar que ha sido realizado bajo la dirección del **Prof. Dr. Ángel María Martín del Rey**, Catedrático de Universidad de la Universidad de Salamanca, y del **Prof. Dr. Roberto Casado Vara**, Profesor Ayudante Doctor de la Universidad de Burgos.

Salamanca, 19 de Junio, 2024

Autor:

SEVERT SILVA MARCOS - 70960835Q

Firmado digitalmente por SEVERT SILVA MARCOS - 70960835Q
Fecha: 2024.06.24 13:12:43 +02'00'

Marcos Severt Silva

Directores:

MARTIN DEL REY ANGEL MARIA - 07953200F

Firmado digitalmente por MARTIN DEL REY ANGEL MARIA - 07953200F
Fecha: 2024.06.22 11:22:22 +02'00'

Prof. Dr. Ángel María Martín del Rey

CASADO VARA ROBERTO CARLOS - 45686237A

Firmado digitalmente por CASADO VARA ROBERTO CARLOS - 45686237A
Fecha: 2024.06.21 10:21:46 +02'00'

Prof. Dr. Roberto Casado Vara

Abstract

The exponential growth of Internet of Things (IoT) networks has prompted concerns about cybersecurity, particularly in the areas of malware detection and mitigation. Given the crucial importance of the role of IoT network security research in today's digital age, where the proliferation of interconnected devices encompasses virtually every aspect of daily life. These devices, which range from smart thermostats to connected health systems, offer a wide range of benefits in terms of efficiency, convenience and data access. However, this interconnectedness also introduces a number of significant challenges in terms of cybersecurity. The inherent vulnerability of IoT devices to cyber-attacks presents a significant threat not only to the privacy and data integrity of individuals and organizations but also to the physical security of people and critical infrastructure operations.

The objective of this PhD thesis is to develop new algorithms that can efficiently, effectively, and reliably protect dynamic Internet of Things (IoT) networks against a range of malware threats. To this end, many solutions to current problems, including network discovery, the classification of network packets into malicious packets, and navigation for malware discovery, are proposed.

The implementation of these solutions in real-world scenarios enables the overcoming of the challenges of malware detection and mitigation in dynamic and diverse IoT network environments. This improvement would not only safeguard against potential disruptions in industrial processes or exposure of sensitive data in home environments but also ensure continuous protection against emerging threats. Innovative approaches to address this issue include the use of algorithms based on reinforcement learning, which offers the ability to effectively adapt to dynamic changes in the network without requiring exhaustive knowledge at all times. This approach not only aims to enhance the efficacy of threat detection and mitigation but also to anticipate and proactively respond to evolving security challenges.

Consequently, this PhD thesis presents a number of intelligent algorithms designed to enhance security in IoT networks by overcoming the inherent limitations of these networks. This is achieved through the utilization of reinforcement learning and a range of classification algorithms.

Resumen

El crecimiento exponencial de las redes Internet de las Cosas (IoT) ha suscitado preocupaciones sobre la seguridad cibernética, especialmente en la detección y mitigación del malware. La investigación en seguridad de redes IoT tiene una importancia crucial en la era digital actual, donde la proliferación de dispositivos interconectados abarca prácticamente todos los aspectos de la vida cotidiana. Estos dispositivos, desde termostatos inteligentes hasta sistemas de salud conectados, ofrecen una amplia gama de beneficios en términos de eficiencia, comodidad y acceso a datos. Sin embargo, esta interconexión también introduce una serie de desafíos significativos en términos de ciberseguridad. La vulnerabilidad inherente de los dispositivos IoT a los ciberataques representa una amenaza no solo para la privacidad y la integridad de los datos, sino también para la seguridad física de las personas y las operaciones críticas de infraestructuras.

El objetivo de esta Tesis Doctoral es invetigar nuevos algoritmos que permitan de forma eficiente, efectiva y fiable la protección de redes IoT dinámicas frente a diversas amenazas malware. Para ello se proponen diferentes soluciones a los problemas actuales como son el descubrimiento de redes, la identificación de paquetes de red en benignos o malignos y la clasificación del tipo malware.

Gracias a la implementación de estas soluciones en escenarios reales conseguimos superar los desafíos en la detección y mitigación de malware en entornos de redes IoT caracterizados por su dinamismo y diversidad. Esta mejora no solo salvaguardaría contra posibles interrupciones en procesos industriales o la exposición de datos sensibles en entornos domésticos, sino que también garantizaría la protección continua ante las amenazas emergentes. Entre los enfoques innovadores para abordar esta problemática, destaca el empleo de algoritmos basados en aprendizaje por refuerzo, los cuales ofrecen la capacidad de adaptarse eficazmente a los cambios dinámicos en la red sin requerir un conocimiento exhaustivo en todo momento. Mediante esta aproximación, se busca no solo detectar y mitigar las amenazas de manera más efectiva, sino también anticipar y responder proactivamente a los desafíos de seguridad en constante evolución.

Como resultado, esta Tesis doctoral presenta varios algoritmos inteligentes que permiten mejorar la seguridad en el ámbito de las redes IoT superando las limitaciones físicas

de las mismas, gracias al uso del aprendizaje por refuerzo y diferentes algoritmos de clasificación.

Agradecimientos

Esta tesis no hubiera sido posible sin el apoyo de varias personas a las que quiero mostrar mi agradecimiento.

En primer lugar gracias a mis directores de tesis, Ángel Martín del Rey y Roberto Casado Vara, por todo el apoyo y confianza que me han dado durante este tiempo, así como por toda su ayuda. Sin su apoyo y motivación esta investigación no habría sido posible.

A mis padres por su constante apoyo, no solo durante el desarrollo de esta tesis, sino a lo largo de toda mi vida. Siempre han estado presentes cuando los he necesitado, y es gracias a ellos que he llegado a donde estoy hoy.

A mi pareja por su paciencia y apoyo durante el desarrollo de la tesis, pues convivir con un doctorando no es tarea sencilla.

A todos aquellos que durante este tiempo han ayudado a que esta tesis sea hoy una realidad.

Contenidos

Lista de Figuras

Lista de Tablas

Abreviaciones

AI	**A**rtificial **I**ntelligence
IoT	**I**nternet **o**f **T**hings
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
D2D	**D**evice-to-**D**evice
WSN	**W**ireless **S**ensor **N**etwork
DDoS	**D**istributed **D**enial **o**f **S**ervice
SI	**S**usceptible-**I**nfected
SIS	**S**usceptible-**I**nfected **S**usceptible
SIR	**S**usceptible-**I**nfected-**R**ecovered
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
LLDP	**L**ink **L**ayer **D**iscovery **P**rotocol
MDP	**M**arkov **D**ecision **P**rocess
GLIE	**G**reedy in the **L**imit with **I**nfinite **E**xploration
MC	**M**onte **C**arlo
TD	**T**emporal **D**ifference
CNNs	**C**onvolutional **N**eural **N**etworks
RL	**R**einforcement **L**earning
KNN	**K** - **N**earest **N**eighbors
SVC	**S**upport **V**ector **C**lassification
IDS	**I**ntrusion **D**etection **S**ystems
EDR	**E**ndpoint **D**etection and **R**esponse
DPI	**D**eep **P**acket **I**nspection
Wi-Fi	**W**ireless **Fi**delity
SIM	**S**ubscriber **I**dentity **M**odule
M2M	**M**achine to **M**achine

LPWAN	**L**ow **P**ower **W**ide **A**rea **N**etwork
LORAWAN	**Lo**ng **Ra**nge **W**ide **A**rea **N**etwork
ISM	**I**ndustrial, **S**cientific, and **M**edical
COAP	**C**onstrained **A**pplication **P**rotocol
MQTT	**M**essage **Q**ueue **T**elemetry **T**ransport
AMQP	**A**dvanced **M**essage **Q**ueuing **P**rotocol
DDS	**D**ata **D**istribution **S**ervice
LWM2M	**L**ightweight **M**2M
MDP	**M**arkov **D**ecision **P**rocess
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
MARL	**M**ulti-**A**gent **R**einforcement **L**earning
PCAP	**P**acket **C**apture **A**pplication **P**rogram

Ningún ingeniero ni químico ha pregonado tener la capacidad de producir un material que sea indistinguible de la piel humana. Es posible que se logre con el tiempo, pero, aun en el supuesto de que existiese este invento, sabríamos lo poco importante que resulta tratar de hacer más humana a una "máquina pensante" cubriéndola con esta carne artificial. (Alan Mathison Turing)

Capítulo 1

Introducción

Capítulo 1

Introducción

1.1 Introducción

Esta tesis doctoral comprende una investigación en el campo de la seguridad del malware en redes del Internet de las Cosas (IoT) aplicando técnicas de inteligencia artificial, que se ha llevado a cabo durante un periodo de tres años en la Universidad de Salamanca. Los conocimientos adquiridos durante este tiempo han permitido probar la hipótesis propuesta en esta tesis doctoral.

La hipotesis planteada en esta tesis es que es posible mejorar la seguridad de las redes del IoT con técnicas de aprendizaje supervisado y por refuerzo a los sistemas de monitorización, detección e identificación del malware actuales. Aquí proponemos una serie de algoritmos y metodologías novedosos para el descubrimiento de la topología, detección de nodos infectados y la identificación del malware atacante en redes IoT, que permiten adecuarse a las limitaciones hardware y software existentes en este tipo de dispositivos. Esta tesis supone un avanze en el campo de la seguridad en redes IoT al proponer una serie de algorimtos inteligentes y metodologías que son escalables, eficientes y semisupervisados de forma que la seguridad este garantizada en un entorno tan heterogéneo y cambiante como el de las redes IoT.

El resto del capítulo está organizado de la siguiente manera: la descripción del problema y la motivación se presentan en la sección 1.2. En la sección 1.3 se muestra la hipótesis de investigación y los objetivos de este trabajo, en la sección 1.4 se presenta la metodología y, por último, en la sección 1.5 se muestra la estructura de esta tesis.

1.2 Descripción del problema y motivación

En el ámbito de las redes IoT y la ciberseguridad, es fundamental centrarse en la detección y mitigación del malware, dada la vulnerabilidad de los dispositivos

interconectados [Aslan and Samet, 2020]. El interés por reforzar las medidas de seguridad se debe al creciente panorama de amenazas que plantean los agentes maliciosos que atacan las redes IoT [Gulatas et al., 2023]. Existe una necesidad de contar con protocolos de seguridad sólidos en los despliegues de IoT para protegerse de posibles infracciones y riesgos para los datos. Sin embargo, para poder aplicar todas estas medidas de seguridad, en primer lugar hay que detectar el malware en la red IoT así como el tipo de malware. En consecuencia, las partes interesadas, como los fabricantes de dispositivos IoT, los proveedores de redes y las empresas de ciberseguridad, están asumiendo un papel fundamental a la hora de reforzar la detección temprana del malware dentro de las redes IoT.

El campo de la seguridad de las redes IoT gira en torno a una red de dispositivos interconectados equipados con sensores y actuadores, diseñados para facilitar la comunicación y el intercambio de datos seguro y efectivo. Esta red de dispositivos abarca diversos ámbitos, como la automatización industrial, los hogares inteligentes, la sanidad y el transporte. Un elemento central del marco de seguridad de las redes IoT es la aplicación de protocolos y mecanismos estrictos para detectar, prevenir y mitigar las amenazas de malware en todo el heterogéneo ecosistema de dispositivos. Entre los componentes clave del marco de la detección del malware en redes IoT encontramos:

- **Soluciones de seguridad en endpoints**: El objetivo es detectar y eliminar el malware de forma efectiva en cada dispositivo. Esta parte de la estrategia combina software antivirus, sistemas de detección de intrusiones (IDS) y mecanismos de detección y respuesta de endpoints (EDR) [Kadrich, 2007, Shen and Shen, 2024].

- **Monitorización del tráfico de red**: La monitorización continua puede ayudar a identificar patrones anómalos que puedan indicar en el análisis del tráfico de las redes IoT la presencia del malware. Una de las formas de hacer esto, es examinando el contenido de los paquetes de datos más allá de los encabezados, lo que es esencial para detectar actividades maliciosas ocultas, a esta técnica se le llama inspeción profunda de paquetes (DPI) [Çelebi et al., 2023, Zang et al., 2023].

- **Análisis de comportamiento y detección de anomalías**: Esta estrategia se basa en el monitoreo continuo de los patrones de actividad de cada dispositivo, permitiendo identificar y mitigar rápidamente comportamientos anómalos que podrían indicar la presencia de malware. Para ello, se establecen perfiles de comportamiento de referencia para cada dispositivo, tomando como base su actividad habitual. Cualquier desviación significativa de estos patrones se convierte en una señal de alerta, activando mecanismos de investigación y respuesta inmediatos [Jeon et al., 2020, Pandit and Mondal, 2023].

El uso de las redes IoT ha revolucionado la forma en la que recogemos datos y interactuamos sobre nuestro entorno. Este nuevo paradigma habilita diferentes automatismos y análisis que facilitan nuestras vidas diarias. A pesar de su potencial para transformar diversos aspectos de nuestras vidas, los dispositivos IoT también presentan algunas limitaciones que es importante considerar de cara a las soluciones planteadas en esta tesis (ver [Balaji et al., 2019, Nižetić et al., 2020]):

- **Dependencia de la conectividad a Internet**: El funcionamiento de los dispositivos IoT depende en gran medida de una conexión a Internet estable, lo que los hace vulnerables a ataques de diferente tipo.
- **Actualizaciones de software**: La actualización del software de los dispositivos IoT puede ser un desafío, lo que los deja expuestos a vulnerabilidades conocidas.
- **Falta de protocolos de seguridad robustos**: Muchos dispositivos IoT no cuentan con protocolos de seguridad integrados sólidos, lo que los hace vulnerables a ataques simples como ataques de fuerza bruta o malware conocido.
- **Dificultad en la gestión de la seguridad**: La gestión de la seguridad de una gran cantidad de dispositivos IoT dispersos geográficamente puede ser una tarea compleja y costosa. La falta de estándares de seguridad unificados y la diversidad de plataformas y sistemas operativos dificultan la implementación de medidas de seguridad consistentes y efectivas.
- **Amenazas emergentes**: El panorama de las amenazas de seguridad en IoT está en constante evolución, con nuevas vulnerabilidades y técnicas de ataque que surgen constantemente.

A pesar de los avances en este campo, todavía hay margen de mejora en el ámbito de la detección del malware en redes IoT. Es necesario diseñar mecanismos que optimicen esos procesos. De este modo, las redes IoT serán más seguras evitando así fallos en procesos industriales por parte de actuadores o la filtración de datos sensibles de usuarios en ámbitos domésticos. Uno de los enfoques para la mejora de la seguridad en este ámbito en constante evolución es el uso de algoritmos y metodologías inteligentes que permitan adaptarse eficientemente a los cambios de la red IoT. De esta forma no es necesario tener un conocimiento absoluto de la red en todo momento para detectar e identificar amenazas.

La solución propuesta en esta tesis propone una serie de algoritmos y metodologías que realizan todas las tareas de descubrimiento, detección e identificación del malware de forma óptima y eficiente. Las siguientes preguntas motivaron nuestra investigación:

- ¿Cómo pueden los algoritmos actuales ser mejorados para identificar la topología de redes IoT de manera eficiente y precisa?
- ¿Qué algoritmos o metodologías autónomas y escalables existen o pueden desarrollarse para la detección de dispositivos infectados por malware en redes IoT?
- ¿Son efectivas las técnicas actuales de detección de malware en un entorno de red IoT cambiante y heterogéneo?
- ¿Hasta qué punto puede el análisis del tráfico de una red IoT ser utilizado para la detección precisa de malware en dispositivos IoT?
- ¿Qué mejoras en las técnicas de identificación de topología y detección de malware pueden contribuir a aumentar la robustez y fiabilidad de las redes IoT?

En esta tesis pretendemos encontrar respuestas a estas preguntas mejorando los propuestas existentes en la literatura en el ámbito de seguridad de redes IoT y creando los algoritmos y/o metodologías necesarias si no existen en el estado del arte, y en caso de que haya trabajos similares, propondremos las mejoras que consideremos oportunas para encontrar soluciones a los problemas encontrados.

1.3 Hipótesis y objetivos

Este trabajo de investigación desarrolla una solución motivada por la discusión anteriormente planteada, considerando las lagunas de investigación existentes en las técnicas de detección e identificación del malware en redes IoT:

La hipótesis de este trabajo de investigación es que es posible mejorar y optimizar las técnicas y tecnologías actuales para la identificación de diversos tipos de malware que se propagan a través de redes IoT.

Se han detectado varias lagunas en algunos de los procesos de seguridad de redes IoT, que podrían mejorarse para lograr una mayor eficiencia, efectividad y fiabilidad de las mismas. Modificamos los procesos y técnicas actuales de detección e identificación del malware en redes IoT con el objetivo de mejorarlos. Además, desarrollamos nuevas técnicas, tecnologías y algoritmos. La característica distintiva de este trabajo es el diseño de algoritmos que permitan adaptarse de forma dinámica a las variaciones de topología de la red IoT, mejorando así la seguridad en las redes IoT. Así, el resultado final de este trabajo es un nuevo sistema dinámico, inteligente y viable de seguridad para su implementación en redes IoT dinámicas.

El objetivo principal de este trabajo de investigación es diseñar algoritmos y metodologías que permitan descubrir la topología de las redes IoT, identificar los nodos infectados por malware e identificar el malware que se propaga en dichas redes.

Este trabajo de investigación abarca diversas áreas de investigación interrelacionadas en el ámbito de las redes IoT y la seguridad informática. Se investiga la arquitectura y los protocolos de comunicación de las redes IoT, así como los dispositivos y aplicaciones que las componen. Se analizan las amenazas de seguridad, especialmente el malware, y se diseñan algoritmos de aprendizaje automático y por refuerzo para la detección de nodos infectados y la gestión eficiente de la red. Además, se exploran técnicas de seguridad de red y análisis de datos para proteger y gestionar de manera eficaz las redes IoT frente a ataques cibernéticos y anomalías de seguridad. Para llevar a buen término esta investigación es necesario establecer unos objetivos más específicos que van a permitir alcanzar la meta principal. Estos objetivos se describen a continuación:

- Investigar y documentar las tecnologías, arquitecturas y protocolos actuales utilizados en redes IoT.
- Investigar y documentar las diversas formas de malware, así como sus métodos de propagación en redes IoT.
- Crear un algoritmo capaz de mapear la topología de redes IoT de manera eficiente, adaptándose a su naturaleza cambiante y a sus diversas dimensiones.
- Crear un algoritmo que, al recorrer por la red IoT, pueda identificar de manera autónoma los nodos infectados, adaptándose a diferentes tamaños de red sin comprometer su eficacia.
- Evaluar y elegir el algoritmo que ofrezca la mejor precisión y eficiencia en la clasificación de nodos IoT como infectados mediante el análisis de tráfico de red.
- Identificar los parámetros de los modelos de malware que se propagan en una red IoT básados en modelos de epidemiología matemática.

1.4 Metodología

Es necesario definir todas las actividades que se realizarán a lo largo de esta investigación, desde la etapa de investigación hasta la etapa de validación de resultados, para asegurar que se logren los resultados esperados en cada fase. El método elegido es el presentado en [Reason and Bradbury, 2001], conocido como *action-research*. Este enfoque está

orientado a la acción y al cambio, lo que le permite centrarse en problemas definidos para producir conocimiento a partir de investigaciones asociadas durante un período establecido de tiempo. Esta metodología popular permite la práctica de la investigación empírica. Comienza con la identificación de un problema real, para el cual se enumeran todas las posibles hipótesis. Luego, estas hipótesis se estudian y se selecciona una en función de la cual se desarrolla una propuesta. Posteriormente, el estudio se centra en verificar la hipótesis seleccionada y se llevan a cabo una serie de acciones que ayudan a determinar si es verdadera o falsa. Finalmente, se extraen conclusiones de la evaluación de los resultados de la investigación. Para formalizar este modelo de investigación, se han definido una serie de actividades que se describen brevemente a continuación. Estas actividades concuerdan con los objetivos de este trabajo de investigación:

1. Identificación y descripción de las características del problema de poder recorrer redes IoT, detectar, marcar y mitigar amenazas malware. Se definirán las características de las redes IoT dinámicas, y se propondrán diferentes hipótesis para la solución parcial o total del problema.

2. Estudio incremental y revisión del estado del arte. A lo largo de esta fase, se ha analizado el estado del arte de las áreas, tecnologías y desarrollos relacionados con la presente investigación, lo que ha permitido obtener el marco teórico y sus posibles desarrollos. Esta fase ha permitido enriquecer el conocimiento del monitoreo y control en redes IoT y ha mejorado la calidad científica del trabajo de investigación presentado.

3. Diseño iterativo y progresivo del modelo propuesto de acción. A partir de la información obtenida en las actividades anteriores, se ha diseñado un modelo integral que integra todos los componentes necesarios que permiten desarrollar técnicas y tecnologías innovadoras y originales para lograr una mayor eficiencia energética en el monitoreo y control de redes IoT en edificios inteligentes.

4. Desarrollo e integración del modelo a través de nuevos algoritmos. En esta fase se ha formalizado la funcionalidad, componentes, iteraciones, etc. de los nuevos algoritmos modulares y se han integrado en la arquitectura desarrollada. La implementación de la solución propuesta ha permitido realizar pruebas y obtener resultados que han servido para la evaluación y formulación de conclusiones.

5. Diseminación continua del conocimiento, resultados y experiencias a la comunidad científica. Esta actividad continua a lo largo del proceso de investigación ha permitido diversas publicaciones en revistas y presentaciones en conferencias y talleres que han validado y dado a conocer el progreso y los resultados parciales

de los diversos hitos de investigación en áreas de experticia con el consiguiente retroalimentación al trabajo.

1.5 Estructura de la tesis

Esta tesis doctoral se divide en siete capítulos y un apéndice. A continuación se describe la estructura de cada uno de los capítulos.

Capítulo 1: Constituye la introducción a esta investigación. Describe los problemas a tratar en la investigación y la motivación detrás de la mismos. El capítulo señala la importancia de diseñar nuevos enfoques para este problema. Se presentan los objetivos, hipótesis y motivación que han llevado al desarrollo de este trabajo de investigación. Por último, se detalla la metodología de investigación aplicada y se describe brevemente la estructura de la esta tesis doctoral.

Capítulo 2: Revisa el estado del arte en lo relativo a las principales ténicas y tecnologías utilizadas en esta investigación como las redes IoT, el malware y el aprendizaje por refuerzo. El capítulo comienza definiendo las redes IoT y revisando sus principales caracteristicas y elementos. Continua introduciendo el malware y sus diferentes tipos, así como, el uso de la epidemiología matemática para modelizar la propagación del malware, describiendo en detalles los tres modelos utilizados en los casos de estudio para la validación de las propuestas. El capítulo acaba haciendo una breve introducción a los conceptos principales del aprendizaje por refuerzo, ya que es el fundamento teórico detras de varios de los algoritmos diseñados en esta tesis.

Capítulo 3: Este capítulo pretender dar una solución novedosa a la primera parte del objetivo general que plantea diseñar un algoritmo para descubrir la topología de una red IoT sin información previa. El capítulo plantea una propuesta de algoritmo para el problema de descubrimiento de una topología de red sin poseer información previa de la misma. Para ello el capítulo analiza una nueva estrategia utilizando el Proceso de Decisión de Markov y redes complejas. En el capítulo se detalla el proceso de modelización matemática del problema y los fundamentos teóricos detras de la resolución del mismo.

Capítulo 4: En este capítulo aborda la segunda parte del objetigo general de esta investigación en relación a diseñar nuevas soluciones para identificar los nodos infectados de las redes IoT así como identificar el malware que los esta infectando. Para ello, el capítulo se divide en tres partes, cada una de ellas describiendo la solución propuesta. En primer lugar se propone un algoritmo para identificar los nodos infectados de la red IoT basado en aprendizaje por refuerzo. A continuación, se describe la metodología

para identificar el malware que esta infectando una red IoT si este es conocido. En caso de que sea desconocido, se procede a modelizar su comportamiento con la metodología propuesta en la tercera parte del capítulo, que permite resolver el problema inverso y calcular los parámetros de los modelos para así identificarlos.

Capítulo 5: En este capítulo se describen los casos prácticos realizados, se presentan y se analizan los resultados obtenidos. Para evaluar y demostrar la eficacia de las nuevas metodologías y los algoritmos propuestos, se han diseñado una cuatro casos de estudio prácticos. Los resultados presentados en este capítulo demuestran la eficacia de los algoritmos y metodologías diseñados para descubrir la topología de las redes IoT desconocidas, identiricar los nodos infectados e identificar el malware que las infecta.

Capítulo 6: Esboza las conclusiones extraídas del trabajo y describe las formas en que contribuye al estado de la técnica. También presenta las futuras líneas de investigación que deja abiertas este trabajo de investigación.

Finalmente, se presenta un listado con todas las referencias bibliográficas que se han utilizado en esta tesis doctoral y que han sido referenciadas a lo largo de la misma.

Capítulo 2

Técnicas y tecnologías para el estudio de la propagación y detección del malware en las redes IoT

Capítulo 2

Técnicas y tecnologías para el estudio de la propagación y detección del malware en las redes IoT

2.1 Introducción

En este capítulo se presentan las técnicas y tecnologías presentes en la literatura que se van a usar en esta tesis doctoral para el analisis y la detección del malware propagando en una red IoT. En primer lugar, se analizan las definiciones y caracteristicas básicas de las redes IoT para permitirnos entender lo necesario sobre las redes IoT así como algunos trabajos relacionados con estos elementos principales de las redes IoT. En segundo lugar, analizamos el malware y sus diferentes tipos, además, revisamos brevemente los diferentes tipos de propagación que tiene el malware. Se motiva el uso de la epidemiología matemática para simular la propagación del malware en las redes IoT y se detallan tres modelos epidemiológicos SI, SIS y SIR. Para concluir, se revisan los conceptos básicos del aprendizaje por refuerzo presentes en la literatura ya que dos de las cuatro nuevas propuestas de este trabajo de investigación se basan en el aprendizaje por refuerzo.

El resto del capítulo está organizado como sigue: la sección 2.2 nos presenta los conceptos básicos de las redes y dispositivos IoT, la sección 2.3 nos muestra los conceptos fundamentales del malware así como una descripción detallada de los modelos epidemiologicos que se usaran en este trabajo. La sección 2.4 nos muestra los conceptos básicos del aprendizaje por refuerzo así como la principal técnica de aprendizaje de este tipo de algoritmos. Para concluir, en la sección **??** se presentan las conclusiones de este capítulo.

2.2 Redes IoT

2.2.1 Introducción

El concepto de IoT, "Internet of Things" o "Internet de las cosas" en español, se refiere a un nuevo paradigma tecnológico donde todo, desde edificios hasta aplicaciones, se interconecta a través de redes para cumplir propósitos específicos [Laghari et al., 2021]. Los avances tecnológicos han iniciado una era de conectividad digital, caracterizada por la disponibilidad de dispositivos informáticos y redes inalámbricas [Khanna and Kaur, 2020]. Esta conectividad ha facilitado la aparición del IoT, un paradigma transformador que está cambiando fundamentalmente nuestro mundo [Hossein Motlagh et al., 2020]. El IoT no se refiere a una única tecnología, sino a un conjunto de sistemas como sensores, actuadores, elementos de red y dispositivos cotidianos, unificados en un proyecto tecnológico interconectado [Sobin, 2020].

El IoT permite emplear sistemas inteligentes y dispositivos interconectados que recogen y aprovechan datos para ejecutar operaciones específicas. Sensores, actuadores y objetos cotidianos recopilan y comparten datos mediante diferentes protocolos de red [Zikria et al., 2021]. Este paradigma posibilita conexiones privadas y públicas entre dispositivos de diversas redes a través de Internet, creando un mundo más conectado e integrado con la automatización de tareas. Al equipar objetos cotidianos con sensores, actuadores y conectividad de red, el IoT los transforma en participantes activos en el mundo digital, permitiendo un flujo fluido de información [Sovacool and Del Rio, 2020].

Este avance en conectividad sin precedentes está revolucionando múltiples sectores, optimizando procesos, automatizando tareas y mejorando la toma de decisiones [Ang et al., 2022]. En la agricultura, los sensores IoT monitorizan condiciones del suelo y la salud de los cultivos, aumentando la productividad y previniendo pérdidas [Farooq et al., 2020]. En la industria, las máquinas sensorizadas simplifican procesos de producción y automatizan decisiones, reduciendo fallos y tiempos de inactividad [Rahim et al., 2021]. En el hogar, los dispositivos IoT transforman casas en hogares inteligentes, ajustando temperatura, iluminación y control de acceso [Maswadi et al., 2020]. En el ámbito laboral, los sensores en el material de trabajo proporcionan información sobre hábitos y condiciones físicas, ayudando a prevenir accidentes laborales [Nappi and de Campos Ribeiro, 2020]. En el sector automovilístico, los coches conectados anticipan condiciones de la carretera y optimizan rutas, mejorando la seguridad y reduciendo la congestión del tráfico [Shrivastava et al., 2020]. Estos ejemplos ilustran cómo el IoT ha impulsado una gran revolución tecnológica en los últimos años [Ketu and Mishra, 2022].

El potencial del IoT va más allá de aplicaciones individuales, abarcando estructuras sociales más amplias (ver Figura 2.1). Al conectar ciudades, infraestructuras y servicios públicos, el IoT está creando entornos urbanos más inteligentes, sostenibles y habitables. Los sistemas de gestión del tráfico optimizan el flujo vehicular, reduciendo congestión y contaminación. Las redes IoT inteligentes equilibran la demanda y suministro de energía, asegurando una distribución eficiente y fiable. Los sistemas de gestión de residuos conectados optimizan las rutas de recogida, minimizando el impacto medioambiental y mejorando el saneamiento.

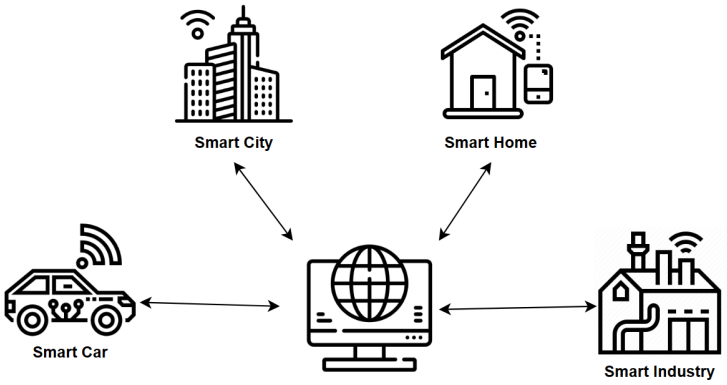


FIGURA. 2.1: Paradigma IoT.

El IoT no está exento de desafíos. Con el aumento de dispositivos conectados, también crece la necesidad de medidas robustas de ciberseguridad para proteger datos sensibles y prevenir ciberataques [Torres-Carrión et al., 2022]. Las preocupaciones sobre la privacidad deben abordarse para asegurar que los datos generados no se utilicen indebidamente. Además, es crucial considerar el impacto medioambiental del IoT, asegurando que su desarrollo sea sostenible [Khatua et al., 2020]. A pesar de estos desafíos, el IoT ofrece una oportunidad sin precedentes para transformar nuestro mundo, impulsando la innovación y mejorando nuestras vidas. Debemos ser conscientes de sus implicaciones, asegurando su uso para el beneficio de la sociedad y la preservación del planeta.

2.2.2 Definición y características de las redes IoT

2.2.2.1 Dispositivo IoT

Hoy en día el uso de dispositivos IoT está tan extendido que todo el mundo usa en su vida cotidiana varios de ellos, por ejemplo cerraduras inteligentes, control por voz de la vivienda, etc [Winarno and Affandi, 2022]. Podemos definir dispositivo IoT como una pieza de hardware que cumple con determinadas tareas muy concretas y que hace posible la comunicación con otros dispositivos a través de Internet [Imteaj et al., 2021]. No solo

se compone de una parte hardware sino que la existencia de un sistema de software es necesaria para su funcionamiento y cumplimiento de las tareas para las que ha sido creado y programado. Los dispositivos IoT pueden ser dispositivos móviles, dispositivos médicos, equipo industrial, etc. Lo que tienen todos en común es el poder comunicarse entre ellos indistintamente de su naturaleza [Souri et al., 2022].

Los dispositivos IoT ofrecen diferentes opciones de utilización de los mismos siendo la industria uno de sus principales usuarios, ayudándoles a generar diferentes oportunidades de negocio y crecimiento. Ademas de reducir costes y aumentar ventas, los dispositivos IoT permiten obtener beneficios de diferentes maneras como el acceso a datos industriales, acceso a datos de pacientes en la industria médica, recopilación de datos de transporte, etc [Chataut et al., 2023].

En cuanto al ciclo de vida de los dispositivos IoT, nos referimos al proceso que explica cómo funciona un dispositivo IoT, cómo se desarrolla y como es capaz de llegar a cumplir con sus tareas [Brandt et al., 2022]. Este ciclo de vida que da forma a lo que conocemos como dispositivo IoT lo podemos definir en cinco fases: despliegue, supervisión, servicio, gestión y actualización (ver Figura 2.2).

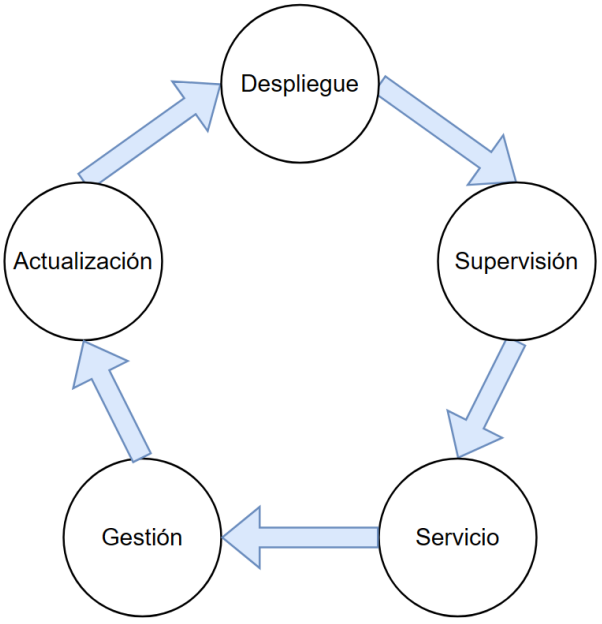


FIGURA. 2.2: Ciclo de vida de un dispositivo IoT.

Al principio, los desarrolladores tienen que desplegar los dispositivos IoT hardware, software y otros componentes necesarios. Tras el despliegue completo, la fase de supervisión es la encargada de asegurar que el dispositivo cumple con los estándares y capacidades esperadas. La fase de servicio hace referencia a el cumplimiento de las tareas para las cuales el dispositivo IoT ha sido diseñado. La fase de gestión hace referencia a como se gestionan las tareas, la información o cualquier otro elemento fruto

del desarrollo de las tareas programadas en el dispositivo. La última fase gestiona todo el dispositivo tras las actualizaciones periódicas que dan comienzo al ciclo siendo necesaria una actualización de software o hardware.

Los dispositivos IoT como ya se ha descrito nacen con el objetivo de poder conectar dispositivos normales a Internet, enviar información a través de la red y dotar de inteligencia a los mismos. El mercado IoT crece cada día estimando más de 20.000 millones el año que viene y por tanto no podemos hacer una categorización específica y rigurosa de los diferentes dispositivos, sin embargo, para el desarrollo de la tesis doctoral hemos establecido tres categorías en base al ámbito de uso de los mismos: diispositivos para consumidores, dispositivos para empresas y dispositivos de uso industrial.

2.2.2.2 Arquitectura de redes IoT

Este apartado se centra en las diferentes formas en las que los dispositivos IoT se agrupan para formar redes IoT complejas con diferentes arquitecturas [Núñez et al., 2022]. Para transferir datos entre sensores IoT y una aplicación web/móvil a través de una plataforma IoT, existen muchas opciones. Cada una tiene sus pros y sus contras. No existe una solución única para todos los casos [Jabraeil Jamali et al., 2020]. En el caso de las soluciones más simples tenemos a las redes cableadas o de corto alcance. Esta primera opción consiste en cablear todos los dispositivos entre si o usar redes inalámbricas de corto alcance mediante tecnologías como el WIFI, Bluetooth, ZigBee, Z-Wave [Tao et al., 2021].

La segunda de los opciones es emplear la forma tradicional para conectar un objeto a Internet a través de una tarjeta Subscriber Identity Module (SIM) [Nita and Mihailescu, 2022]. Este tipo de redes comúnmente conocidas como Machine to Machine (M2M) tiene una serie de desventajas:

- Equipos caros: Los dispositivos M2M a menudo son costosos debido a la complejidad de su diseño y fabricación. Además, suelen necesitar componentes especializados y tecnología avanzada para garantizar una comunicación fiable y segura entre máquinas.
- Consumo de energía muy elevado: Se requiere una fuente de alimentación permanente o una batería potente. Esto hace que el tamaño del sensor sea incompatible con muchos casos de uso.
- Gran huella energética: Los dispositivos M2M a menudo tienen un tamaño considerable debido a la necesidad de albergar componentes de comunicación,

sensores y otras tecnologías necesarias para su funcionamiento. Esta gran huella puede dificultar su integración en algunos entornos o dispositivos donde el espacio es limitado.

- Un coste de suscripción para la conectividad : Además del costo inicial de adquisición de los dispositivos M2M, también suele haber un costo recurrente asociado con la conectividad a la red. Esto puede incluir tarifas de suscripción mensuales o anuales para acceder a la infraestructura de comunicaciones necesaria para enviar y recibir datos entre los dispositivos.

Por eso es difícil conectar objetos pequeños con M2M bien al aceptar que no hay fuente de alimentación o bien por el hecho de que el coste del sensor sea superior al del propio objeto conectado. En este contexto de problemas derivados del bajo alzance o los altos costes, surgen las redes de tipo LPWAN (Low Power Wide Area Networks)(ver Figura 2.3) [Salama et al., 2023]. Estas redes estan caracterizadas por ser de bajo coste, tamaño muy reducido, poder operar con batería con mucha autonomía y con comunicaciones muy optimizadas para baja cantidad de datos [Iqbal et al., 2020].

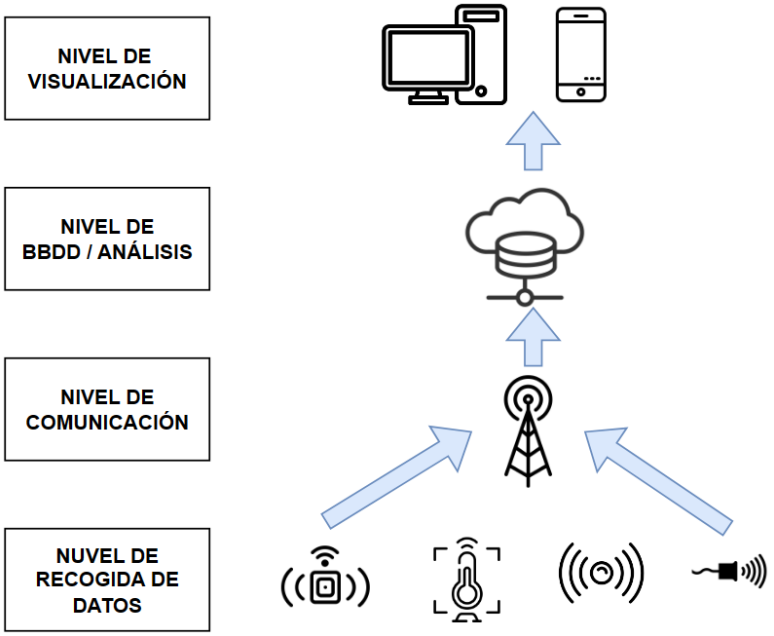


FIGURA. 2.3: Ejemplo de red IoT de tipo LPWAN.

Mientras que en el ámbito doméstico la arquitectura más común es la Low Power Wide Area Network (LPWAN), en el industrial encontramos SigFox [Mane, 2021] y Long Range Wide Area Network (LoRaWAN) [Jouhari et al., 2023] (ver Figura 2.4). Estos son dos protocolos de comunicación inalámbrica que funcionan en la banda de frecuencia Industrial, Scientific and Medical (ISM), que es una banda de frecuencia libre que no requiere licencia. Sin embargo, esta banda de frecuencia está regulada para evitar

interferencias con otros dispositivos. Por lo tanto, SigFox y LoRaWAN están diseñados para utilizar solo una pequeña cantidad de ancho de banda, lo que limita la cantidad de datos que se pueden transmitir [Jouhari et al., 2023]. El principio de funcionamiento de estos protocolos es el siguiente:

- A intervalos regulares, el sensor se despierta para transmitir los datos medidos.
- Los datos se envían en paquetes de 12 bytes que son recogidos por todas las pasarelas visibles.
- Las pasarelas transmiten los datos por Internet a una plataforma de IoT.
- La plataforma de IoT procesa los datos.

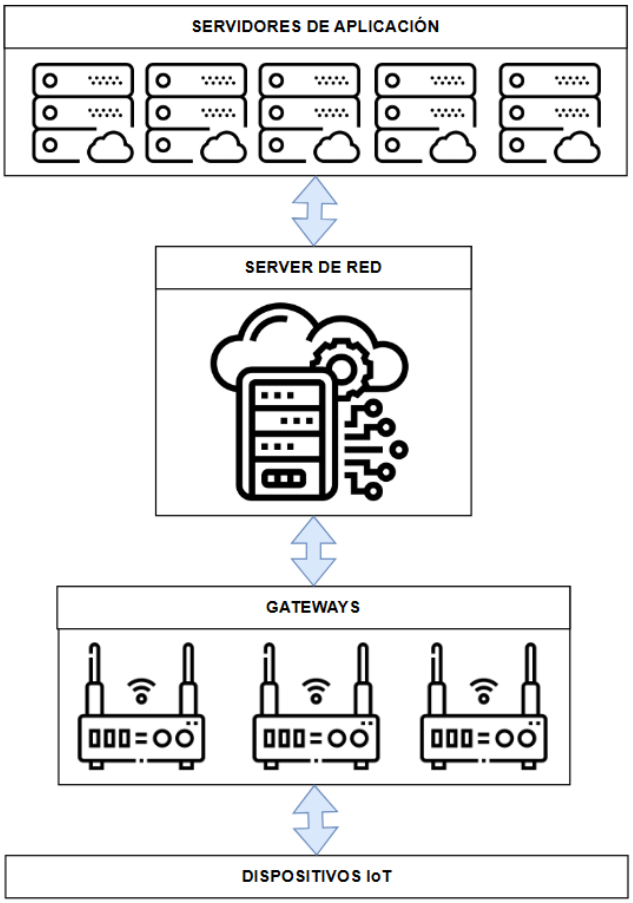


FIGURA. 2.4: Diagrama de red LoRaWAN.

2.2.2.3 Protocolos IoT

Este apartado del capítulo tratará sobre la comunicación entre dispositivos IoT y los protocolos que se siguen para ello. Estos protocolos y estándares IoT fijan una serie

de directrices para todo el ciclo de vida de los dispositivos IoT expuesto en el apartado anterior.

En cuanto a los protocolos, en el mundo de la informática estos definen reglas que hay que seguir para transmitir datos entre dispositivos no necesariamente homogéneos y que estos puedan entenderse y entablar comunicación. Estos protocolos garantizan una transmisión de datos siguiendo una estructura homogénea independientemente del dispositivo que manda o recibe la información. Por todo ello diferentes protocolos son desarrollados y estandarizados para que cualquier fabricante de dispositivos IoT pueda emplearlos [Tightiz and Yang, 2020].

La gran diversidad de protocolos IoT exigen a los usuarios finales la utilización adecuada en función del contexto. Cada protocolo permite la comunicación entre dispositivos, centros de datos, la pasarela al centro de datos o el dispositivo a la pasarela. Los protocolos desempeñan un papel fundamental y diferenciado en cada capa de comunicación del modelo Open Systems Interconnection (OSI) y en este apartado se exponen los principales protocolos de comunicación que serán relevantes en el desarrollo de este trabajo a la hora de hablar de posibles anomalías en la comunicación entre dispositivos y su categorización como posibles ataques de malware:

- **Constrained Application Protocol (CoAP)**: CoAP es un protocolo ligero diseñado para dispositivos con recursos limitados. Se basa en HTTP, pero utiliza un formato de mensaje más pequeño y menos funciones para reducir el consumo de recursos. CoAP es una buena opción para aplicaciones que necesitan enviar y recibir pequeñas cantidades de datos, como lecturas de sensores o comandos de actuadores [Alhaidari and Alqahtani, 2020].

- **Message Queue Telemetry Transport (MQTT)**: MQTT es un protocolo de mensajería diseñado para redes de baja ancho de banda e inseguras. Es un protocolo de publicación-suscripción, lo que significa que los editores envían mensajes a temas, y los suscriptores reciben mensajes de temas. MQTT es una buena opción para aplicaciones que necesitan intercambiar datos de forma poco frecuente, como la monitorización remota o el seguimiento de activos [Bhowmik and Riaz, 2023].

- **Advanced Message Queuing Protocol (AMQP)**: AMQP es un protocolo de mensajería más complejo que MQTT. Admite una gama más amplia de tipos de mensajes y funciones, y es más escalable. AMQP es una buena opción para aplicaciones que necesitan manejar un gran volumen de mensajes o que requieren más flexibilidad [Yakupov, 2022].

- **Data Distribution Service (DDS)**: DDS es un protocolo de mensajería centrado en los datos diseñado para aplicaciones en tiempo real. Admite una variedad de tipos de datos, incluidos datos estructurados y multimedia. DDS es una buena opción para aplicaciones que necesitan distribuir datos a múltiples suscriptores con baja latencia [Bodkhe and Tanwar, 2020].
- **Lightweight M2M (LwM2M)**: LwM2M es un protocolo diseñado para la gestión de dispositivos en aplicaciones M2M (máquina a máquina). Es un protocolo ligero que se basa en CoAP. LwM2M es una buena opción para aplicaciones que necesitan gestionar un gran número de dispositivos con recursos limitados [Doyu et al., 2021].

Los estándares son fundamentales en la industria de la tecnología de la información y en especial en el IoT [Karie et al., 2021]. Estos definen las reglas, regulaciones y directrices que deben seguirse para desarrollar cualquier dispositivo o proyecto de IoT. Las directrices son específicas para el hardware, el software u otros componentes de cualquier proyecto o sistema. Diversas instituciones de la industria y comunidades académicas trabajan en el desarrollo de estándares que deben seguirse durante todo el ciclo de vida de un dispositivo IoT.

2.2.2.4 Software IoT

En este apartado se tratará el software que permite a los dispositivos el cumplimiento con las tareas especificadas y que es de especial importancia a la hora de comprender en futuros capítulos como el malware es una amenaza en las redes IoT. Al igual que el software estándar, el software IoT habilita el funcionamiento del hardware IoT y ejecuta instrucciones. El sistema operativo controla todo el dispositivo al igual que en un ordenador personal. Podemos considerar software IoT cualquier aplicación que necesite realizar algunas operaciones comunes como la recopilación de datos, análisis de datos, integración de dispositivos y aplicación. Se han desarrollado múltiples sistemas de IoT para realizar tareas específicas. El software de IoT aborda las principales tareas relacionadas con las redes, las plataformas, el middleware y los sistemas integrados. La robótica y los sistemas de pedidos son ejemplos de sistemas empresariales que se integran con muchas aplicaciones. El software integrado admite estas aplicaciones para la ejecución de esas tareas [Al-Turjman et al., 2020, Razzaq, 2020]. Entre los diferentes tipos de software existentes hemos escogido elaborar una clasificación en base a las tareas que cumplen los diferentes dispositivos IoT:

- **Software de recolección de datos**: La recolección de datos es una de las funciones principales de los dispositivos IoT. El resultado de muchas tareas a

ejecutar de forma cíclica por una red IoT depende en muchas ocasiones de la correcta recolección de datos por parte de algunos de los dispositivos. Actualmente la tendencia de estos dispositivos es la de enviar los datos recogidos a un servidor central de la red IoT [Ali et al., 2020b].

- **Software de integración de dispositivos**: Las redes IoT están formadas por numerosos dispositivos heterogéneos siendo necesario un correcto funcionamiento del software que permite la integración entre todos ellos. En otras palabras, el software permite la integración y vinculación entre dispositivos que automáticamente asegura una cooperación eficaz [Khalid and Ameen, 2021].

- **Software de análisis en tiempo real**: Muchos de los sistemas IoT tienen como tarea establecida la toma de decisiones en base a datos recogidos, por ello, se necesita de un correcto funcionamiento del software de análisis para analizar en tiempo real los datos recogidos usando diferentes lógicas permitiendo así la toma de decisiones de forma efectiva [Krishnamurthi et al., 2020].

- **Software de extensión de aplicaciones**: Con el aumento del uso del paradigma IoT, los usuarios buscan diversas formas para controlar cualquier dispositivo. Este tipo de software es el encargado de ampliar la integración de los dispositivos con diferentes aplicaciones móviles o de escritorio [Kassim, 2020].

2.2.3 Elementos de las redes IoT

2.2.3.1 Sensores y actuadores

Definimos sensor como aquel dispositivo que es capaz de detectar una acción externa, presión, temperatura, etc., y transmitirla de forma adecuada. Por otra parte definimos actuador como aquel dispositivo de funcionamiento eléctrico, neumático o hidráulico, que actúa como un motor para cambiar la posición de aparatos móviles, como válvulas o compuertas [Gulati et al., 2022]. Estados dos definiciones hacen referencia a las dos categorias de dispositivos Iot más grandes que se dividen en base a si son pasivos o activos [Fizza et al., 2021]. El sensor activo es un tipo de sensor que genera señales utilizando un soporte externo cambiando sus propiedades en función de la tarea que debe cumplir. El sensor pasivo es justo lo contrario del sensor activo, no necesitando ningún tipo de excitación externa para generar señales de salida.

En muchas ocasiones los actuadores se han considerado el complemento tecnológico del sensor porque genera acciones y movimientos convirtiendo la energía eléctrica en energía no eléctrica. En la Tabla. 2.1 podemos ver los sensores y actuadores más comunes hoy

en día. Actualmente y dada la amplia gama de tareas que deben cumplir tanto sensores como actuadores es conveniente sepáralos y hablar de ambos como las dos categorías de dispositivos IoT [Kalsoom et al., 2020, Nguyen and Kim, 2021].

Nombre del Sensor	**Descripción utilidad del sensor**
Sensor de Visión	Determina la presencia de objetos o los colores de objetos y los convierte en salida visual. La cámara, los controladores y la luz son los principales componentes de este sensor.
Sensor de Presión	Este sensor es un tipo de dispositivo electromecánico que mide la presión y genera señales. Utiliza un diafragma para detectar y medir fuerzas.
Sensor de Temperatura	Mide la temperatura y sus cambios. Después de medición, genera una señal de salida o muestra directamente la lectura.
Sensor de Radiación	Detecta diversas partículas alfa, beta y gamma para proporcionar señales para visualizar en el dispositivo. Las energías máximas y mínimas detectables son características del sensor.
Sensor de Posición	Detecta las posiciones y los movimientos de los objetos para suministrar señales. Este sensor tiene especificaciones incluyendo las características y rango de medición.
Actuador de Válvula	Se encarga de abrir y cerrar válvulas permitiendo o no el paso de fluidos cuando recibe una señal. Estos dispositivos son esenciales en el ámbito de la industria 4.0.
Actuador de Puerta	Se encarga de abrir y cerrar puertas cuando recibe una señal. Estos dispositivos son esenciales en el ámbito de las smart cities y smart homes.
Actuador de Contacto	Se encarga de permitir o cortar el flujo eléctrico por una línea de baja, media o alta tensión. Estos dispositivos son esenciales en el ámbito de la industria 4.0 y smart cities.

Tabla. 2.1: Principales sensores y actuadores

2.2.3.2 Redes Inalámbricas

En este apartado se expone el papel de las redes inalámbricas en el paradigma IoT. El desarrollo de la tecnología IoT avanza junto con el crecimiento de los dispositivos móviles, creando entornos inteligentes como ciudades y hogares inteligentes [Lai et al., 2020]. La conectividad entre dispositivos cotidianos demanda alta conectividad, velocidades de datos extremadamente altas y baja latencia, por lo que las redes inalámbricas 5G son un habilitador clave de la tecnología IoT [Gulati et al., 2022].

Las redes inalámbricas están evolucionando rápidamente, como se observa en el despliegue de diversas redes de diferentes tamaños: redes de área personal inalámbrica (WPAN), redes de área local inalámbrica (WLAN), redes de área metropolitana inalámbrica (WMAN) y redes de área amplia (WWAN) [Singh et al., 2020]. Estas

redes incluyen redes celulares, ad hoc, malla, redes de comunicación vehicular y redes de sensores. Sin embargo, carecen de seguridad física, ya que la comunicación se realiza mediante radiación electromagnética en espacio abierto [Ndjuluwa et al., 2023]. Este es un desafío significativo en la seguridad de las redes IoT, investigado a fondo en esta tesis doctoral.

Mejorar la seguridad de las redes inalámbricas IoT presenta desafíos técnicos como la compatibilidad con redes existentes y la complejidad de implementación. Estudios indican que 5G es esencial para el IoT debido a sus tecnologías habilitadoras [Ahmad et al., 2021, Chanal and Kakkasageri, 2020, Mohanta et al., 2020]. La conectividad confiable para dispositivos IoT requiere la integración de diversas tecnologías 5G [Shafique et al., 2020].

Una tecnología importante es la comunicación dispositivo a dispositivo (D2D), que permite la comunicación directa entre dispositivos IoT sin una base centralizada, mejorando la tasa de comunicación de datos [Salim et al., 2023, Yashoda and Shivashetty, 2022]. Aunque D2D ofrece muchas ventajas, también presenta problemas de seguridad significativos [Chakraborty and Rodrigues, 2020]. Otra tecnología clave es Millimeter Wave Technology (MWT), que utiliza frecuencias de 30 a 300 GHz para transmitir grandes tasas de datos entre sensores simultáneamente [Farooq and Lokam, 2023, Loktongbam et al., 2023]. El uso de relays en redes IoT mejora la escalabilidad y cobertura. Los dispositivos IoT pueden conectarse a estaciones repetidoras (RS), proporcionando mejor conectividad (ver Figura 2.5) [Ashraf et al., 2021].

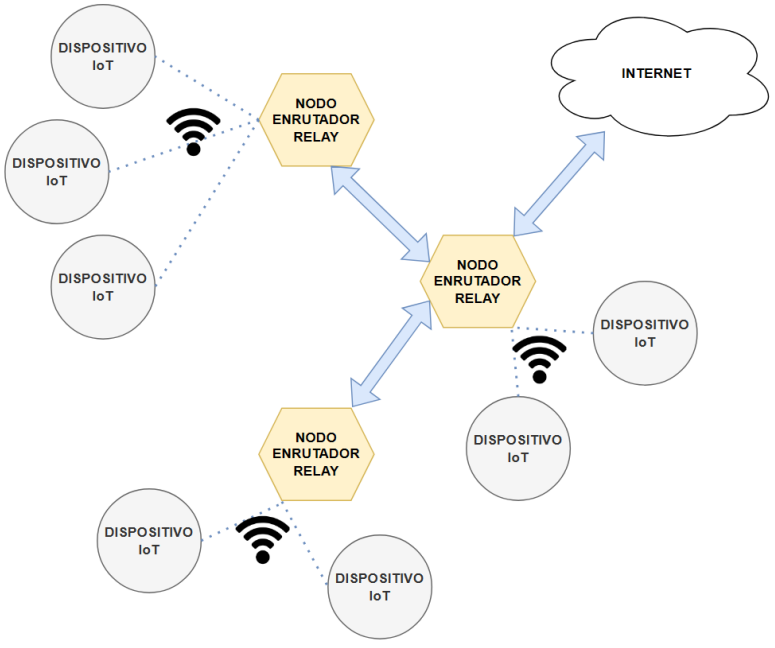


FIGURA. 2.5: Diagrama de red IoT wireless 5G con relays.

Las redes de sensores inalámbricas (WSN, por sus siglas en inglés) son populares por su facilidad de implementación y capacidad de autoorganización [Sharma et al., 2020]. A diferencia de las redes tradicionales, las WSN pueden desplegarse ad hoc, sin necesidad de infraestructura física preestablecida, lo que las hace flexibles y versátiles para entornos remotos o con limitaciones de infraestructura [Gulati et al., 2022]. Los nodos en WSN establecen conexiones y cooperan para transmitir datos eficientemente, adaptándose a cambios en el entorno sin intervención manual [Amodu et al., 2023]. Su despliegue ad hoc y autoorganización las convierten en herramientas valiosas para la automatización, monitoreo y toma de decisiones en diversos ámbitos [Shakeri et al., 2020].

2.3 Malware en Redes IoT

2.3.1 Introducción

En el panorama actual de la seguridad informática, el malware se ha convertido en una amenaza cada vez más preocupante, evolucionando a un ritmo alarmante y adoptando diversas técnicas de ofuscación para ocultarse en los sistemas. La proliferación de este software malicioso pone en grave riesgo la integridad de los sistemas informáticos y la infraestructura de Internet. Por ello, resulta imperativo desarrollar métodos efectivos para detectar y neutralizar el malware antes de que pueda causar daños a gran escala [Caviglione et al., 2020].

Los desarrolladores de malware usan técnicas más avanzadas para crear malware difícil de detectar y eliminar. En primer lugar tenemos la ofuscación de código [Ebad et al., 2021]. La ofuscación de código es el proceso de modificar un fichero fuente para que deje de ser útil a un programador a la hora de leerlo. Sin embargo, sigue siendo totalmente funcional. El proceso no altera la salida del programa, aunque puede alterar las instrucciones del mismo para lograr el mismo resultado [Zhang et al., 2021].Las técnicas de ofuscación más usadas son:

- **Rename Obfuscation**: Esta técnica consiste en renombrar métodos y variables. Hace que el código fuente descompilado sea más difícil de entender para un humano, pero no altera la ejecución del programa.
- **Control Flow Obfuscation**: La ofuscación del flujo de control es la síntesis de condiciones, ramas e iteraciones que producen una lógica ejecutable válida, pero producen resultados semánticos no deterministas. En pocas palabras, hace que el código tenga una lógica muy difícil de entender para un programador. El rendimiento en tiempo de ejecución del fichero fuente puede verse alterado.

- **Instruction Pattern Transformation**: Convierte instrucciones comunes creadas por el compilador en otras construcciones menos obvias.

Otra de las técnicas de propagación que emplea el malware hoy en día es el denominado "comportamiento sigiloso". El malware puede diseñarse para ocultar su actividad maliciosa mediante la ejecución de tareas maliciosas en segundo plano o la eliminación de sus propias huellas digitales. Esta situación complica mucho la detección de aquellos dispositivos infectados usando técnicas tradicionales o incluso recurriendo a la supervisión manual de los equipos por parte de profresionales de la ciberseguridad [Kara, 2023].

Una de las grandes amenazas a las que nos enfrentamos cuando hablos del auge del malware en el mundo digital, es la adaptabilidad del mismo. El malware puede adaptarse a los cambios en los sistemas informáticos o en los métodos de detección mediante la actualización periódica del código o la adopción de nuevas técnicas de ofuscación. Es por ello que nuevas técnicas de detección, eliminación y aislamiento de malware son necesarias, especialmente cuando hablamos de redes IoT donde el manejo de gran cantidad de información y un gran número de conexiones están al orden del día [Debas et al., 2023].

Si entramos a valorar el impacto actual del malware dentro de las redes IoT tenemos una amplia gama de daños [Madhvan and Zolkipli, 2023]. Teniendo en cuenta la baja capacidad de los dispositivos IoT y la sensibilidad de información que estos manejan, nos enfrentamos a graves problemas como son:

- **Pérdida de datos:** La pérdida de datos en sensores IoT es un problema creciente que puede tener un impacto significativo en las organizaciones que dependen de estos dispositivos para recopilar y recopilar datos.
- **Daños a los archivos:** Los daños a los archivos pueden interrumpir los procesos productivos que dependen de los datos almacenados en los sensores. Esto puede causar pérdidas financieras y de productividad.
- **Interrupción del funcionamiento del sistema:** La interrupción del funcionamiento del sistema en sensores IoT puede provocar una negativa de servicio en los sistemas o aplicaciones que dependen de estos sensores.
- **Toma de control del sistema:** En algunos casos, la toma de control del sistema en sensores IoT puede utilizarse para causar daños físicos.
- **Robo de información confidencial:** El robo de información confidencial en redes IoT es un problema creciente que puede tener un impacto significativo en las organizaciones que dependen de estos dispositivos para recopilar y recopilar datos.

La investigación continua en el campo de la detección de malware es esencial para desarrollar métodos más efectivos. Actualmente los investigadores están trabajando en nuevas técnicas que puedan detectar malware conocido y desconocido, así como malware que utiliza técnicas de ofuscación. Una de las técnicas más empleadas a día de hoyt es el uso de aprendizaje automático para detectar malware. Los algoritmos de aprendizaje automático pueden ser muy efectivos para detectar malware, pero pueden requerir una gran cantidad de datos de entrenamiento. Otra de estas técnicas es el uso de métodos de detección multifacéticos o métodos en tiempo real. Los enfoques de detección en tiempo real pueden detectar malware antes de que cause daños [Tayyab et al., 2022].

La detección de malware es una tarea compleja y en constante evolución debido a la naturaleza adaptable y evasiva del malware. Si bien los enfoques y métodos actuales han logrado avances significativos, aún queda un largo camino por recorrer para lograr una detección completa y efectiva de todo el malware existente [AliAhmad et al., 2023]. La investigación continua en esta área es crucial para mantener la seguridad de los sistemas informáticos y la infraestructura de Internet.

2.3.2 Tipos de malware en las redes IoT

2.3.2.1 Definición de malware

El malware, o software malicioso, es un término general que describe cualquier programa o código diseñado para dañar sistemas informáticos. Es hostil, intrusivo y deliberadamente perjudicial, y puede invadir, dañar o desactivar computadoras, sistemas informáticos, redes y dispositivos móviles [Bansal et al., 2021].

Los motivos detrás del malware son variados. Puede utilizarse para obtener dinero, sabotear la capacidad de una persona para trabajar, hacer una declaración política o simplemente para presumir. Aunque el malware no puede dañar el hardware físico de los sistemas o equipos de red, puede robar, cifrar o eliminar datos, alterar o secuestrar funciones informáticas centrales y espiar la actividad informática de una persona sin su conocimiento o permiso [Abusitta et al., 2021].

Las aplicaciones maliciosas pueden esconderse en aplicaciones aparentemente legítimas, especialmente cuando se descargan de sitios web o enlaces directos (en un correo electrónico, mensaje de texto o mensaje de chat). Aquí es importante observar los mensajes de advertencia de los dispositivos cuando un software de terceros nos solicita permisos de acceso en nuestro dispositivo [Caviglione et al., 2020].

En cuanto a como un ciberdelincuente elige su objetivo para infectarlo con malware, los dispositivos móviles, como teléfonos inteligentes y tabletas junto a las redes IoT son

objetivos atractivos. Esto representa una amplia gama de objetivos de ataque para adware y spyware, registradores de teclas y malvertising, además de ser un método atractivo para que los delincuentes pues crean y distribuyen malware a la mayor cantidad de objetivos posible, con relativamente poco esfuerzo [Carvajal and Inlago, 2021].

2.3.2.2 Tipos de malware

Tal y como venimos describiendo en el capítulo el mundo del malware evoluciona constantemente con nuevas amenazas. El término malware puede referirse a una amplia gama de software malicioso, cada uno con su propio conjunto único de características y objetivos, sin embargo, para reconocer las amenazas potenciales y aplicar una protección eficaz, es importante establecer una clasificación (En base a su nombre en español o inglés más conocido) de los tipos de malware:

- **Virus:** Es uno de los tipos de malware más comunes. Se caracteriza por ser capaz de replicarse a si mismos, de ser ejecutados por un usuario y extenderse a otros programas o ficheros de los diferentes dispositivos IoT [Talukder and Talukder, 2020].
- **Worms:** Son bastante similares a los virus en el aspecto de ser capaces de replicarse a si mismos, sin embargo, se diferencian en su capacidad para propagarse de forma independiente sin depender de las acciones de los usuarios. Este tipo de malware aprovecha las vulnerabilidades de las redes y los sistemas para propagarse, a menudo causando interrupciones generalizadas y consumiendo recursos de la red [Singh and Singh, 2021].
- **Troyanos:** Este tipo de malware se caracteriza por suplantar programas o archivos inofensivos y buscan que los usuarios los ejecuten. Una vez se activan, estos buscan obtener acceso total al sistema permitiendo así el robo de información confidencial, instalar malware adicional o interrumpir el correcto funcionamiento del sistema [Nelson et al., 2023].
- **Ransomware:** El ransomware es un malware basado en la encriptación de los ficheros de las víctimas. Esta encriptación hace que los ficheros sean innacesibles para el usuario. Todo ello suele ir acompañado de unos requisitos monetarios a cambio de permitir a la víctima recuperar sus ficheros [Beaman et al., 2021].
- **Spyware:** El spyware esta diseñado para recopilar información personal y hábitos de navegación sin conocimiento de la victima. Estos malware pueden rastrear actividades en línea, robar contraseñas y credenciales, y vigilar el comportamiento

del usuario, a menudo con fines de publicidad dirigida o monetización de datos [Naser et al., 2023].

- **Rootkits:** Los rootkits son tipos de malware especialmente sigilosos que consiguen un control profundo de las funciones básicas de un sistema, a menudo aprovechando privilegios administrativos. Pueden ocultar su presencia, modificar el comportamiento del sistema y crear puertas traseras para que los atacantes mantengan el acceso remoto [Nadim et al., 2023].

- **Botnets:** Este malware se caracteriza por robar el control del dispositivo para hacerle miembro de una red de dispositivos infectados con el objetivo de realizar diferentes tipos de ataques. Estos dispositivos infectados, conocidos como bots, pueden utilizarse para enviar mensajes de spam, lanzar ataques distribuidos de denegación de servicio (DDoS) o robar información confidencial [Mahboubi et al., 2020].

- **Phishing Attacks:** Los ataques de phishing consisten en engañar a los usuarios para que revelen información confidencial, como contraseñas o credenciales financieras, a través de correos electrónicos, sitios web o mensajes de redes sociales fraudulentos [Tandale and Pawar, 2020].

Teniendo en cuenta la velocidad de evolución del malware la lista de malware se debe ir actualizando de forma periódica para estar al día de las nuevas vulnerabilidades. La tarea de todo experto en ciberseguridad pasa por mantenerse informado sobre las amenazas más recientes y aplicar medidas de seguridad sólidas para proteger a las personas, las organizaciones y las infraestructuras críticas de los peligros siempre presentes del malware.

2.3.3 Métodos de propagación del malware en las redes IoT

A pesar de las ventajas que ofrecen, los dispositivos IoT son muy vulnerables al malware. Su pequeño tamaño y su limitada potencia de cálculo los convierten en un blanco fácil para los atacantes. Proteger estos dispositivos de los ataques es un reto diario para los equipos de seguridad de las empresas. Cualquier dispositivo informático no estándar y de bajas prestaciones se clasifica como dispositivo IoT [Mahboubi et al., 2020]. Puede ser de consumo, como televisores inteligentes y wearables, o industrial, como sistemas de control, cámaras de vigilancia, rastreadores de activos o dispositivos médicos. Independientemente de su enfoque, los dispositivos IoT han cambiado la forma en que el mundo funciona y vive. Hay miles de tipos diferentes de dispositivos IoT. Lo que todos tienen en común es la capacidad de conectarse a una red. Esta conectividad permite

controlar estos dispositivos a distancia y acceder a sus datos y recopilarlos [De Fazio et al., 2021].

A pesar de sus muchas ventajas, los dispositivos IoT son extremadamente atractivos para los hackers malintencionados debido a los datos que generan, recopilan y comparten, y a las operaciones que realizan. El hecho de que estén conectados a una red los deja abiertos a ataques remotos, y sus factores de diseño hacen que no tengan la seguridad integrada necesaria para protegerse de las amenazas y la explotación [Alenezi et al., 2020]. Podemos listar una serie de fallas de seguridad que los hacen vulnerables al malware [Sadhu et al., 2022]:

- **Contraseñas débiles o codificadas:** Muchos dispositivos IoT vienen con contraseñas predeterminadas débiles o codificadas que son fáciles de adivinar o descifrar. Esto permite a los atacantes tomar el control de los dispositivos sin mucha dificultad.
- **Falta de cifrado:** Los datos almacenados o transmitidos en texto sin formato son vulnerables a la intercepción, la corrupción y el secuestro. Esto podría permitir a los atacantes robar información confidencial o interrumpir el funcionamiento de los dispositivos.
- **Componentes vulnerables:** Los dispositivos IoT suelen utilizar componentes de hardware comunes que son susceptibles a las vulnerabilidades. Esto significa que los atacantes pueden explotar estas vulnerabilidades para tomar el control de los dispositivos.
- **Diversidad de dispositivos:** Los dispositivos IoT vienen en una variedad de formas y tamaños, lo que dificulta el desarrollo de medidas de seguridad que se adapten a todos los dispositivos.
- **Falta de auditoría:** Los dispositivos IoT suelen ser difíciles de auditar, lo que dificulta a los administradores de sistemas identificar y remediar las vulnerabilidades.
- **Mecanismos de actualización deficientes:** Muchos dispositivos IoT carecen de mecanismos de actualización seguros, lo que permite a los atacantes explotar las vulnerabilidades conocidas.
- **Falta de conciencia de seguridad:** Los usuarios a menudo no son conscientes de las vulnerabilidades de seguridad de los dispositivos IoT, lo que los convierte en objetivos fáciles para los atacantes.

2.3.4 Impacto del malware en las redes IoT

Las redes IoT, que se están desplegando en la actualidad a una gran escala, representan un objetivo de ataque y plantean riesgos de seguridad significativos. Un único dispositivo infectado con malware puede propagarse rápidamente por toda la red, hasta el punto de inutilizarl [Ngo et al., 2020]. Por ello, la detección y cuarentena del malware no siempre es suficiente para prevenir su propagación. Por otro lado, el uso de la teoría de control tradicional para la contención del malware no es eficaz, ya que la mayoría de los métodos existentes no consideran estrategias de control en tiempo real que puedan adaptarse a la información sobre la infección de los nodos de la red [Uchenna et al., 2021].

Al tratarse de dispositivos de bajas capacidades como se ha enunciado previamente, estas redes puden sufrir diferentes consecuencias fruto de los ataques malware contra las mismas. Una de estas consecuencias es la de causar daños físicos o financieros. El malware puede causar daños físicos a los dispositivos IoT, como incendios o explosiones. También puede causar daños financieros, como pérdidas de datos o interrupción de los servicios [Gaurav et al., 2023].

Otra de las grandes consecuencias del ataque a una red IoT es la violación de la privacidad. Los dispositivos IoT están diseñados para recopilar y transmitir datos, lo que los convierte en un objetivo atractivo para los ciberdelincuentes [Alrubayyi et al., 2021]. Los ciberdelincuentes pueden utilizar los datos recopilados por los dispositivos IoT para realizar una serie de actividades ilegales, como las siguientes:

- **Robo de identidad:** Los ciberdelincuentes pueden utilizar los datos personales para crear identidades falsas o cometer fraude.
- **Extorsión:** Los ciberdelincuentes pueden amenazar con publicar datos personales o sensibles si la víctima no paga un rescate.
- **Ataques dirigidos:** Los ciberdelincuentes pueden utilizar los datos recopilados para realizar ataques dirigidos a personas o empresas específicas.

Sin embargo, todas estas nefastas consecuencias de un ataque malware contra una red IoT pueden tener una serie de contramedidas básicas como:

- Mantener los dispositivos IoT actualizados con las últimas actualizaciones de seguridad [Kim et al., 2017].
- Utilizar contraseñas seguras y únicas para todos los dispositivos IoT [Shah and Venkatesan, 2018].

- Deshabilitar las funciones de recopilación de datos que no sean necesarias [Shaukat et al., 2021].

Estas medidas básicas pero eficientes no son válidas para frenar el malware una vez este ha infectado a la red. Es por ello que esta tesis se centra en diferentes algoritmos para navegar, detectar, clasificar y poner en cuarentena a los diferentes dispositivos que forman el grafo de la red [Aslan et al., 2021].

2.3.5 Epidemiología matemática aplicada a la propagación del malware

En este apartado se recogen los modelos epidemiológicos que han sido empleados durante el desarrollo de la tesis doctoral [del Rey, 2015, Guillén and Del Rey, 2018, Martín del Rey, 2023a]. Todos los modelos empleados durante la tesis se tratan de modelos compartimentales. Este tipo de modelos se caracterizan por asignar compartimentos con etiquetas como S, I o R (Susceptibles, Infecciosos y Recuperados). Las personas, por diferentes dinámicas que se modelizan, pueden pasar de un compartimento a otro. El orden en el que se sitúan las etiquetas suele mostrar el patrón de flujo de individuos entre los diferentes compartimentos [Ben-Shlomo et al., 2023]. Estos modelos compartimentales intentan predecir aspectos como la propagación de una enfermedad, el número total de personas infectadas o la duración de una epidemia. También pueden estimar diversos parámetros epidemiológicos, como el número de casos reproductivos. Estos modelos pueden mostrar cómo las distintas intervenciones de salud pública pueden afectar a los resultados de la epidemia, por ejemplo, cuál es la forma más eficiente de distribuir un número limitado de vacunas a una población [Tang et al., 2020].

Durante todo el apartado del capítulo hablaremos de dispositivos y no de individuos, pues es el ámbito en el que se han aplicado estos modelos durante todo el desarrollo de la tesis doctoral. En el contexto de la propagación del malware, los dispositivos susceptibles son aquellos que pueden ser infectados, los dispositivos infectados son aquellos que ya han sido comprometidos y pueden propagar el malware a otros dispositivos, y, en algunos modelos, los dispositivos recuperados son aquellos que han sido limpiados del malware y han adquirido cierta inmunidad o protección contra futuras infecciones.

Para modelizar la propagación del malware, se han empleado los modelos SI, SIS y SIR, cada uno con sus características y aplicaciones específicas:

- **Modelo SI (Susceptible-Infectado):** En este modelo, los dispositivos pasan del estado susceptible (S) al estado infectado (I). No se considera la recuperación, lo que es adecuado para ciertos tipos de malware que, una vez instalados, permanecen en el dispositivo a menos que se tomen medidas específicas para eliminarlos.

- **Modelo SIS (Susceptible-Infectado-Susceptible):** Este modelo permite que los dispositivos infectados se recuperen y vuelvan al estado susceptible. Es útil para modelar malware que puede ser eliminado temporalmente, pero donde los dispositivos pueden ser reinfectados si no se actualizan las medidas de seguridad.
- **Modelo SIR (Susceptible-Infectado-Recuperado):** Este modelo incluye un estado de recuperación, donde los dispositivos infectados pueden ser limpiados y no pueden ser reinfectados, al menos durante un periodo de tiempo. Es aplicable a escenarios donde las actualizaciones de seguridad o los parches pueden proteger efectivamente contra futuras infecciones del mismo malware.

Estos modelos proporcionan una comprensión suficiente de cómo el malware se propaga en una red de dispositivos IoT y permiten evaluar la eficacia de diversas estrategias de detección desde un punto de vista teórico. Estos modelos de epidemiologia matemática nos han permitido diseñar los algoritmos y metodologias, así como diseñar casos de prueba en los que validar el rendimiento de las propuestas. En el resto del capítulo, se van a detallar más estos tres modelos de epidemiologia matemática para poder entender su funcionamiento matemático.

2.3.5.1 Modelo SI

El sistema SI es uno de los modelos más básicos y fundamentales para entender la propagación del malware en una red IoT. En este modelo, la población se divide en dos compartimentos:

- (S)usceptibles: Dispositivos que pueden infectarse con el malware.
- (I)nfectados: Dispositivos que han contraído el malware y pueden transmitirla a los susceptibles.

quedando el diagrama del modelo SI como se ve en la Figura 2.6.

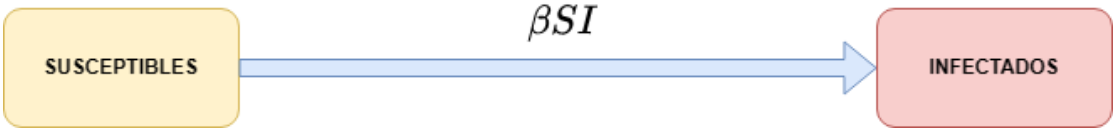


FIGURA. 2.6: Diagrama de flujo del modelo SI.

La particularidad de este modelo es que la población está formada solo por susceptibles e infectados, y si se contagia un dispositivo, el malware es permanente, es decir,

no hay recuperación. La dinámica del modelo se describe generalmente mediante ecuaciones diferenciales que representan el cambio en el número de dispositivos en cada compartimento a lo largo del tiempo. Las ecuaciones del modelo SI son:

$$\begin{cases} \dfrac{dS}{dt} = -\beta S(t)I(t), \quad S(0) = S_0 > 0 \\ \dfrac{dI}{dt} = \beta S(t)I(t), \quad I(0) = I_0 > 0 \end{cases} \tag{2.1}$$

donde:

- $S(t)$ es el número de dispositivos susceptibles en el tiempo t.
- $I(t)$ es el número de dispositivos infectados en el tiempo t.
- β es la tasa de transmisión o contagios.

Por tanto, si $\beta > 0$ es la tasa de contagios, entonces $\beta S(t)I(t)$ es la cantidad de susceptibles que se convierten en infectados con el paso del tiempo. Es decir, se puede interpretar el modelo SI tal que, la ecuación $\frac{dI}{dt} = -\beta S(t)I(t)$ indica que los susceptibles disminuyen a medida que entran en contacto con los infectados y se contagian. Por otro lado, la ecuación $\frac{dS}{dt} = \beta S(t)I(t)$ muestra que los infectados aumentan a medida que los susceptibles se contagian.

Teniendo en cuenta que $N = S(t) + I(t)$ el sistema del modelo SI se puede simplicar en

$$\frac{dI}{dt} = \beta(N - I(t))I(t) \tag{2.2}$$

Esta ecuación es resoluble por el método de variables separadas, integrando se tiene que la solución para $I(0) = I_0$ es:

$$I(t) = \frac{I_0 N}{(N - I_0)e^{(-\beta N t)} + I_0} \tag{2.3}$$

Es facil ver, que cuando $I(t) \longrightarrow N$ cuando $t \longrightarrow +\infty$, todos los dispositivos acabaran infectados.

Hay que destacar, que el modelo SI asume que la población es constante, que no hay recuperaciones de dispositivos infectados y que todos los dispositivos tienen la misma probabilidadd de encontrarse en contactos con otros dispositivos y por tanto infectarse. A pesar de estas limitaciones, el modelo SI proporciona una base muy útil para comprender la propagación del malware en una red IoT y para desarrollar modelos más complejos que incluyan otros factores relevantes para la dinámica de la propagación del malware.

2.3.5.2 Modelo SIS

El sistema SIS es un modelo matemático utilizado en epidemiología para estudiar la dinámica de la propagación del malware en una población susceptible e infectada. Las siglas reflejan la dinámica de los dispositivos que pueden ser susceptibles a la infección, infectados y luego recuperados pero nuevamente susceptibles. Por tanto, el diagrama del modelo SIS como vemos en la Figura 2.7.

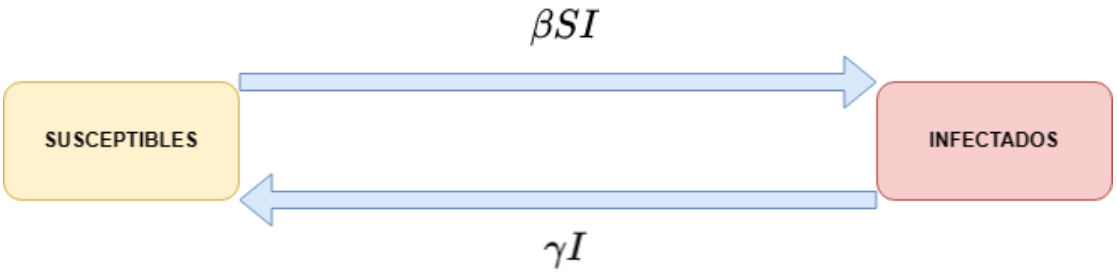


FIGURA. 2.7: Diagrama de flujo del modelo SIS.

Las ecuaciones que describen el sistema SIS son:

$$\begin{cases} \frac{dS}{dt} = -\beta S(t)I(t) + \gamma I(t) \\ \frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) \end{cases} \tag{2.4}$$

donde:

- $S(t)$ número de dispositivos susceptibles en el tiempo t.
- $I(t)$ número de dispositivos infectados en el tiempo t.
- β es la tasa de transmisión o contagio del malware.
- γ es la tasa de recuperación del malware.

La dinámica del sistema SIS depende de la interacción entre la tasa de transmisión y la tasa de recuperación. Cuando la tasa de transmisión es alta y/o la tasa de recuperación es baja, el malware puede persistir en la población. Por otro lado, si la tasa de recuperación es alta y/o la tasa de transmisión es baja, el malware tiende a desaparecer.

La solución del sistema SIS puede obtenerse analíticamente o mediante métodos numéricos, y permite predecir la evolución de la enfermedad en la población a lo largo del tiempo. Estas ecuaciones diferen de las del modelo SI en el término $\gamma I(t)$, que describe el comportamiento de los dispositivos que se recuperan del malware y se vuelven

a convertir en susceptibles. Similarmente al modelo SI, la población de tamaño fijo es N, y analogamente, podemos reducir el sistema SIS a una dimensión sustituyendo $S(t) = N - I(t)$, de donde se tiene:

$$\frac{dI}{dt} = \beta I(t) \left(\left(N - \frac{\gamma}{\beta} \right) - I(t) \right) \tag{2.5}$$

Similarmente al apartado anterior, esta ecuación tiene solución analítica por el método de separación de variables, que con $I(0) = I_0$ es:

$$I(t) = \frac{\beta N - \gamma}{\beta + \left(\frac{(\beta N - \gamma)}{I_0} - \beta \right) e^{-(\beta N - \gamma)t}} \tag{2.6}$$

Si analizamos la función $I(t)$ podemos predecir el comportamiento a largo plazo del malware, es decir, si habra o no dispositivos infectados. Vamos a analizar que ocurre en el largo plazo, para ello vamos a estudiar el siguiente limite:

$$\lim_{t \to \infty} I(t) = \lim_{t \to \infty} \frac{\beta N - \gamma}{\beta + \left(\frac{(\beta N - \gamma)}{I_0} - \beta \right) e^{-(\beta N - \gamma)t}} \tag{2.7}$$

este limite va a tener dos soluciones, cuando $\beta N - \gamma > 0$, el resultado del limite será $\frac{\beta N - \gamma}{\beta} > 0$ ya que $e^{-\infty} = 0$. Por otro lado, si $\beta N - \gamma < 0$, el resultado del limite será 0 ya que $e^{+\infty} = \infty$. A partir de este resultado, se puede definir el número esperado de contactos infecciosos que realiza un dispositivo infectado, como el número reproductivo básico del modelo SIS, como $R_0 = \frac{\beta N}{\gamma}$, donde βN es la tasa con la que un dispositivo contagia a los demas dispositivos y $\frac{1}{\gamma}$ es el tiempo que un dispositivo permanezca infectado. Por tanto, se puede interpretar la solución del limite, como que si $R_0 > 1$ se contagiará toda la red (esto se conoce como epidemia), mientras que en caso contrario, si $R_0 < 1$ no se contagiara toda la red IoT (Teorema del Umbral para el modelo SIS [Montesinos-López and Hernández-Suárez, 2007]).

Para concluir este análisis, se puede decir que en el modelo SIS comparado con el modelo SI, siempre hay una vuelta a ser susceptibles. Pero habra epidimia dependiendo de lo rápido que sea esta recuperación. Por tanto, saber el signo de $\beta N - \gamma$ es imprescindible o lo que es lo mismo, saber si R_0 es mayor o menor que uno.

2.3.5.3 Modelo SIR

El modelo SIR es un modelo fundamental en epidemiología matemática que describe la propagación del malware en una red IoT propuesto por Kermack y McKendric [Kermack and McKendrick, 1927]. El modelo divide a la población en tres compartimentos:

- $S(t)$: Número de dispositivos susceptibles en el tiempo t.
- $I(t)$: Número de dispositivos infectados en el tiempo t.
- $R(t)$: Número de dispositivos recuperados en el tiempo t.

el diagrama con notación sería el que vemos en la Figura 2.8.

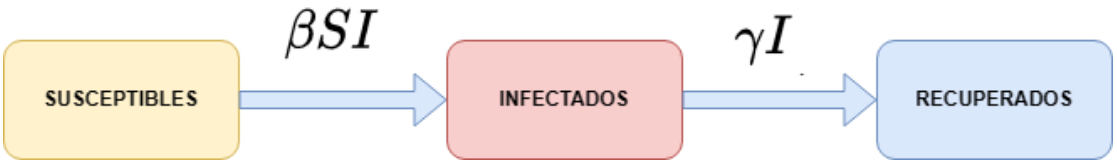


FIGURA. 2.8: Diagrama de flujo del modelo SIR.

Las dinámicas del modelo SIR se describen mediante el siguiente sistema de ecuaciones diferenciales ordinarias:

$$\begin{cases} \frac{dS}{dt} = -\beta S(t)I(t), \quad S(0) = S_0 = N - I_0 \\ \frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t), \quad I(0) = I_0 > 0 \\ \frac{dR}{dt} = \gamma I(t), \quad R(0) = 0 \end{cases} \tag{2.8}$$

donde β es la tasa de transmisión o contagio y γ es la tasa de recuperación, $S(t), I(t), R(t)$ corresponden al número de dispositivos en los compartimentos susceptible, infectados y recuperados respectivamente, en el tiempo t, con $S(t) + I(t) + R(t) = N$. La dinámica del modelo incluye una fase inicial de crecimiento exponencial de la infección, un pico y una fase de decaimiento hasta alcanzar un equilibrio. Para entender mejor este modelo epidemiologico, vamos a realizar un breve analisis cualitativo.

En primer lugar, es fácil probar que $\frac{dS}{dt} \leq 0$ y $\frac{dR}{dt} \geq 0$, con lo que se puede afirmar que $S(t)$ es decreciente, luego $N \geq S(0) \geq S(t) \geq 0$. Además, $R(t)$ es creciente, y por tanto, $0 \leq R(0) \leq R(t) \leq N$. Si ahora analizamos que $\frac{dI}{dt} = I(t)(\beta S(t) - \gamma)$ puede ser positiva, negativa o cero, ya que $(\beta S(t) - \gamma)$ puede tomar cualquiera de esos valores. Por tanto, se tienen dos posibles casos que van a determinar el Teorma del Umbral para el modelo SIR [Martín del Rey et al., 2024, Montesinos-López and Hernández-Suárez, 2007]):

- Si $S(0) = S_0 < \frac{\gamma}{\beta}$ entonces $S(t) < \frac{\gamma}{\beta}$ ya que $S(t)$ es decreciente. Por tanto, se tiene que $\frac{dI}{dt} < 0$, lo que implica que $\lim_{t\to\infty} I(t) = 0$
- Si $S(0) = S_0 > \frac{\gamma}{\beta}$ se pueden encontrar algunos valores de t tales que $S(t) > \frac{\gamma}{\beta}$ por lo que $I(t)$ crecerá pudiendose formar una epidemia. Sin embargo, como sabemos

que $S(t)$ es decreciente, se puede encontrar un tiempo t^*, en el que $S(t) < \frac{\gamma}{\beta}$ si $t > t^*$. Esto significa que $I(t)$ tiene un máximo en t^* y luego ya comienza a decrecer.

Para encontrar los puntos críticos del modelo SIR [Zaman et al., 2008], analizamos el sistema de ecuaciones diferenciales:

$$\begin{cases} \frac{dS}{dt} = -\beta S(t)I(t) = 0 \\ \frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) = 0 \\ \frac{dR}{dt} = \gamma I(t) = 0 \end{cases} \tag{2.9}$$

De la ecuación $\frac{dR}{dt} = 0$, obtenemos que $I = 0$. Con $I = 0$, la ecuación $\frac{dI}{dt} = 0$ se satisface sin importar el valor de S. Por lo tanto, tenemos los siguientes puntos críticos:

- Punto crítico trivial (sin infección): $(S, I, R) = (S_0, 0, R_0)$ donde S_0 y R_0 son los valores iniciales de susceptibles y recuperados, respectivamente.
- Punto crítico endémico (con infección): $(S, I, R) = \left(\frac{\gamma}{\beta}, 0, R_0\right)$. Este punto crítico es significativo solo si $S_0 > \frac{\gamma}{\beta}$, lo que implica que inicialmente hay suficientes susceptibles para mantener una infección endémica.

El punto crítico trivial representa el estado en el que la enfermedad ha sido erradicada o no se ha introducido en la población. El punto crítico endémico representa un equilibrio en el que el malware puede mantenerse en la población si $R_0 > 1$, donde $R_0 = \frac{\beta}{\gamma}$ es el número reproductivo básico. Además, se puede definir ademas un número de reproducción efectiva, $R_e = \frac{\beta S_0}{\gamma}$. Este nuevo número de reproducción efectiva supone el umbral que determina si el malware causara epidemia o se extinguira ya que:

- Si $R_e \geq 1$, entonces $I(t)$ decrece a cero cuanto $t \longrightarrow \infty$.
- Si $R_e > 1$ entonces $I(t)$ empieza creciendo, alcanza su máximo y decrece a 0 cuando $t \longrightarrow \infty$.

2.4 Aprendizaje por refuerzo

2.4.1 Introducción

El Aprendizaje por Refuerzo (RL por sus siglas en inglés) emerge como un paradigma innovador dentro del Aprendizaje Automático, diferenciándose sustancialmente de

los enfoques tradicionales. En contraste con la dependencia de conjuntos de datos etiquetados y comparaciones directas entre resultados esperados y obtenidos, el RL se fundamenta en la interacción dinámica entre un agente y su entorno [Wang et al., 2020]. Este enfoque innovador permite al agente aprender de manera autónoma a través de la experimentación, tomando acciones y recibiendo recompensas o penalizaciones en función del resultado obtenido. De este modo, el RL simula el proceso de aprendizaje natural, similar a la forma en que los humanos y los animales adquieren habilidades y conocimientos [Shakya et al., 2023].

A diferencia del Aprendizaje Supervisado, que se centra en modelar relaciones entre entradas y salidas predefinidas, el RL persigue un objetivo más ambicioso: maximizar la recompensa a largo plazo. Esta característica fundamental permite al agente explorar diversas estrategias, incluso si algunas implican recompensas inmediatas más bajas, con el objetivo de alcanzar un beneficio mayor en el futuro [Le et al., 2022]. Un ejemplo ilustrativo es el de un vehículo autónomo que necesita sortear un obstáculo: alejarse del destino a corto plazo puede parecer contraproducente, pero es necesario para alcanzarlo finalmente.

El Aprendizaje por Refuerzo se basa en un proceso de aprendizaje similar al que experimentamos los humanos al realizar actividades. Al principio, una persona que intenta hacer una actividad comete errores. Sin embargo, a través de la práctica y el ensayo y error, aprende a coordinar sus movimientos y a realizarla con mayor precisión. Este proceso de aprendizaje se basa en recompensas por los éxitos y castigos por los fallos [Ladosz et al., 2022].

2.4.2 Elementos del Aprendizaje por Refuerzo

En este apartado se va a especificar cuales son los elementos que componen un algoritmo de aprendizaje por refuerzo.

En la imagen Figura 2.9 podemos ver el proceso de aprendizaje de un agente [Oroojlooy and Hajinezhad, 2023]:

- El Agente observa el estado.
- El Agente, mediante una función de toma de decisión, determina la acción a realizar
- Tras la acción realizada, el entorno determina la recompensa que se le debe aportar al agente.
- Se almacena la recompensa obtenida sobre la experiencia realizada de estado-acción

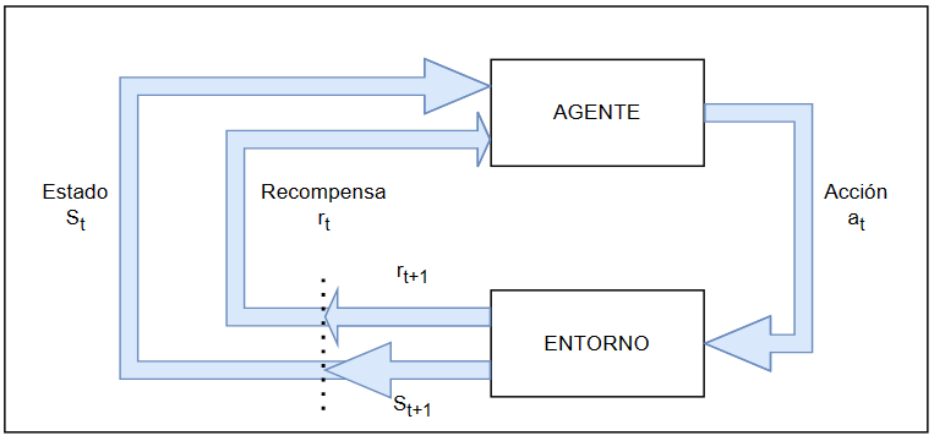


FIGURA. 2.9: Modelo de Aprendizaje por Refuerzo

2.4.2.1 Agente

Es el componente responsable de adquirir conocimientos para resolver el problema planteado. El agente es una entidad con capacidad de memoria y razonamiento. Su principal función es interactuar con el entorno en el que opera, tomando las decisiones necesarias para alcanzar el objetivo definido, y aprender de las recompensas y castigos que obtiene del entorno, con el fin de maximizar las recompensas recibidas [Feriani and Hossain, 2021]. El proceso mediante el cual se determina una acción se denomina política. El objetivo del agente es descubrir la política que optimice sus recompensas. Esta política se ajusta durante el proceso de entrenamiento, utilizando para ello las acciones previas, las recompensas obtenidas y las observaciones realizadas [Shah, 2020].

En el contexto del aprendizaje por refuerzo, es dentro del agente donde se ubica la red neuronal, siendo el agente responsable de probar múltiples configuraciones para encontrar la política óptima. Las entradas de la red incluyen las observaciones del entorno, las recompensas y las acciones realizadas por el propio agente; la salida de la red proporciona una actualización en la función (Ver Figura 2.10) [Nguyen et al., 2020]. Como ocurre con cualquier red profunda, esta puede diseñarse de diversas maneras y con un número variable de capas.

2.4.2.2 Entorno

En el ámbito del Aprendizaje por Refuerzo, el entorno se define como el espacio físico o virtual en el cual el agente opera y con el que puede interactuar. Este entorno no solo proporciona el contexto en el que se desarrolla el aprendizaje, sino que también juega un papel crucial en el proceso de recompensa [Padakandla, 2021].

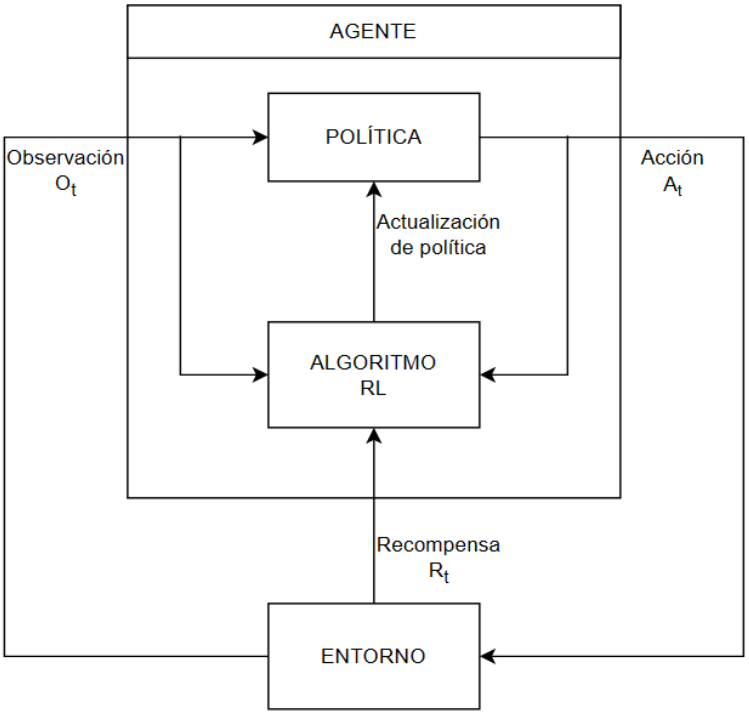


FIGURA. 2.10: Modelo interno de un agente

El entorno ofrece al agente datos sensoriales o información contextual que le permite comprender su situación actual y tomar decisiones. Las acciones del agente tienen efectos en el entorno, modificando su estado y generando nuevas situaciones a las que el agente debe adaptarse. El entorno establece la función de recompensa, determinando el valor que se asigna a cada estado o acción del agente [Luo et al., 2024]. Esta recompensa es fundamental para guiar el aprendizaje del agente y motivarlo a tomar acciones que lo lleven a alcanzar sus objetivos.

2.4.2.3 Estado

En el contexto del Aprendizaje por Refuerzo, el estado actual representa la situación específica en la que se encuentra el agente dentro del entorno en un momento determinado. Este estado encapsula la información relevante sobre el entorno y la posición del agente, proporcionando una descripción precisa del contexto en el que se debe tomar la siguiente acción[Eppe et al., 2022]. El estado actual debe incluir toda la información relevante para que el agente pueda tomar decisiones informadas. El estado actual cambia constantemente a medida que el agente interactúa con el entorno y las acciones del agente modifican su situación. El agente debe ser capaz de percibir el estado actual del entorno a través de sensores o información disponible [Shah, 2020].

El estado actual juega un papel crucial en el aprendizaje por refuerzo, ya que proporciona al agente la base sobre la cual tomar decisiones. Al comprender su estado actual, el agente puede [Vouros, 2022]:

- Evaluar sus opciones: Considerar las diferentes acciones disponibles y sus posibles consecuencias en el entorno.
- Seleccionar la mejor acción: Elegir la acción que, según su política de comportamiento, tiene la mayor probabilidad de conducir a un resultado favorable (maximizar la recompensa).
- Actualizar su política: Aprender de las experiencias pasadas y ajustar su política de comportamiento para tomar mejores decisiones en el futuro.

2.4.2.4 Acción

En el ámbito del Aprendizaje por Refuerzo, la acción representa la decisión que el agente toma en un estado específico (S_t) para interactuar con el entorno. El objetivo del agente es elegir la acción óptima para cada estado, es decir, la acción que le permita maximizar la recompensa a largo plazo [Zhu et al., 2021].

Características clave de la acción [Wang et al., 2020]:

- Representación formal: La acción se denota como at, donde t representa el instante actual de tiempo.
- Tipos de acciones: Las acciones pueden ser de dos tipos:
  - Discretas: Cuando existe un número fijo de posibilidades entre las que el agente puede elegir. Un ejemplo es un robot clasificador que solo puede agrupar los objetos en dos categorías (A o B).
  - Continuas: Cuando existe un número infinito de valores dentro de un rango específico. En el caso de un vehículo, la dirección o la velocidad pueden tomar cualquier valor dentro del rango permitido por las características del vehículo.
- Impacto en el entorno: La acción seleccionada por el agente modifica el estado del entorno, generando nuevas situaciones y consecuencias.
- Selección de la acción: La elección de la acción se realiza en base a la política de comportamiento del agente, la cual define la probabilidad de seleccionar cada acción disponible en un estado determinado.

La acción juega un papel fundamental en el aprendizaje por refuerzo, ya que es el medio a través del cual el agente explora el entorno y experimenta las consecuencias de sus decisiones [Shakya et al., 2023]. A medida que el agente toma acciones y observa los resultados, aprende a asociar las acciones con las recompensas y a ajustar su política de comportamiento para seleccionar acciones que le lleven a obtener mayores beneficios en el futuro [Ladosz et al., 2022].

2.4.2.5 Recompensa

En el ámbito del aprendizaje por refuerzo, el concepto de recompensa juega un papel central. Esta constituye el único parámetro que la red recibe como retroalimentación sobre la calidad de la respuesta del agente ante los estímulos del entorno [Eschmann, 2021]. Para evaluar el desempeño del agente en el entorno, se requiere diseñar una función de recompensa que lo cuantifique. La recompensa asignada debe ser un valor escalar, donde valores más altos indican un mejor resultado y viceversa [Rmus et al., 2021].

La función de la recompensa tiene como parámetros el estado del agente en el entorno y la acción que este efectúa para este estado [Eschmann, 2021].

$$\text{Recompensa } = \text{f(estado,accion)} \tag{2.10}$$

La función de recompensa se describe matemáticamente como el sumatorio de las recompensas en cada instante de t, y donde k es la duración del episodio

$$G_t = \sum_{k=0}^{T} T_{t+k+1} \tag{2.11}$$

2.4.3 Técnicas de aprendizaje por refuerzo

2.4.3.1 Cadenas de Márkov

Las cadenas de Markov son procesos que ocurren en tiempo discreto, en los cuales una variable aleatoria X_n cambia con el tiempo. Estas cadenas llevan el nombre del matemático Andréi Márkov. En este contexto, la probabilidad de que ocurra un evento $(X_n = j)$ depende únicamente del evento inmediatamente anterior (X_{n-1}). Si en una cadena de Markov las probabilidades no dependen del tiempo n en el que se observan,

se le denomina cadena homogénea, lo que indica que las probabilidades son constantes en cada paso [Bellman, 1957].

$$P\left(X_n \mid X_1 \ldots X_{n-1}\right) = P\left(X_n \mid X_{n-1}\right) \tag{2.12}$$

Si en una cadena Homogénea finita con m posibles estados, y existe la posibilidad de ir de un estado a otro, se denominan probabilidades de transición.

$$p_{ij} = P\left(X_n = j \mid X_{n-1} = i\right) \tag{2.13}$$

Esta ecuación describe las probabilidades de transición de un estado i a un estado j en una cadena de Markov homogénea y finita con m posibles estados $(E_1, E_2, \ldots, E_m)$, donde $\mathbf{i}, \mathbf{j} = 1, 2, \ldots, \mathrm{m}$. Si $p_{ij} > 0$, significa que el estado E_i puede alcanzar al estado E_j. Además, si $p_{ji} > 0$, se dice que la comunicación es mutua.

La forma más común de representar una cadena de Markov es mediante un grafo dirigido, en el que cada nodo representa un estado diferente y los enlaces entre los nodos indican las probabilidades de transición. La suma de todas las probabilidades de transición que salen de un nodo debe ser igual a 1 [Kearns et al., 2002].

2.4.3.2 Proceso de decisión de Markov

Los procesos de decisión de Markov se utilizan para describir el entorno en el que se desarrolla el aprendizaje por refuerzo. Estos procesos son una extensión de las cadenas de Markov que también incluyen las decisiones tomadas por el agente [Wei et al., 2020a].

Existen tres enfoques para determinar la política óptima en un problema de decisión secuencial: política miope, política de horizonte finito y política de horizonte infinito [Adler and Subramanian, 2024].

- Las políticas miopes consideran que el agente solo maximiza la recompensa del siguiente estado.
- Las políticas de horizonte finito consideran un marco temporal específico sobre el cual maximizar la política.
- La política de horizonte infinito toma en cuenta todos los tiempos futuros y es la más utilizada en el aprendizaje por refuerzo.

2.4.4 Q Learning

El algoritmo Q-Learning se ha convertido en una herramienta fundamental en el ámbito del aprendizaje por refuerzo, gracias a su capacidad para encontrar políticas óptimas en diversos entornos [Clifton and Laber, 2020]. Su objetivo principal es maximizar la recompensa acumulada a lo largo de la interacción con el entorno, lo que lo convierte en una herramienta ideal para resolver problemas de control y optimización [Vidyasagar, 2020].

Sin embargo, cuando se aplica a sistemas infinitos, el algoritmo Q-Learning enfrenta tres desafíos principales [Kabanda et al., 2023]:

- Recompensas Infinitas: En sistemas infinitos, la recompensa total esperada puede ser infinita, lo que dificulta la convergencia del algoritmo a una solución óptima. Esto se debe a que la suma de infinitas recompensas futuras también es infinita, lo que puede llevar a una sobreestimación del valor de las acciones.
- Descuento de Recompensas: Para abordar el problema de las recompensas infinitas, se introduce un factor de descuento que asigna un menor valor a las recompensas futuras en comparación con las recompensas inmediatas. Sin embargo, la elección del factor de descuento adecuado puede ser un desafío en sí mismo, ya que un valor demasiado alto puede conducir a una subestimación del valor real de las acciones, mientras que un valor demasiado bajo puede generar inestabilidad en el algoritmo.
- Priorización de Recompensas Cercanas: En entornos con recompensas infinitas, el algoritmo Q-Learning no tiene en cuenta la proximidad temporal de las recompensas. Esto significa que todas las recompensas futuras se consideran con el mismo peso, sin importar lo lejanas que estén en el tiempo. Esto puede llevar a que el algoritmo priorice acciones que generen recompensas pequeñas a corto plazo, en lugar de acciones que generen recompensas más grandes a largo plazo.

Para enfrentar estos desafíos, se han propuesto diversas estrategias [Vidyasagar, 2020]:

- Funciones de Recompensa Normalizadas: Se pueden utilizar funciones de recompensa normalizadas que limiten la magnitud de las recompensas, evitando así el problema de las recompensas infinitas.
- Algoritmos con Descuento Adaptativo: Se pueden emplear algoritmos que ajusten el factor de descuento de forma dinámica en función del estado actual del sistema.

- Priorización Temporal: Se pueden incorporar mecanismos de priorización temporal que asignen un mayor peso a las recompensas más cercanas en el tiempo.

2.4.4.1 On-policy y off-policy

En el ámbito del aprendizaje por refuerzo, la elección del algoritmo adecuado es crucial para el éxito del proceso de aprendizaje. Entre los diferentes tipos de algoritmos disponibles, dos categorías importantes son los algoritmos on-policy y off-policy [Uehara et al., 2022]. Los algoritmos on-policy se caracterizan por aprender y mejorar la política actual a medida que interactúan con el entorno. En otras palabras, la política que guía la toma de decisiones del agente se actualiza utilizando las experiencias obtenidas siguiendo la misma política [Prudencio et al., 2023]. Características principales:

- Actualización continua: La política se actualiza constantemente en función de las recompensas recibidas.
- Exploración-explotación: El equilibrio entre la exploración de nuevas acciones y la explotación de las acciones conocidas es crucial para un aprendizaje eficiente.
- Convergencia: La convergencia a una política óptima depende de la tasa de aprendizaje y la estrategia de exploración.

A diferencia de los algoritmos on-policy, los algoritmos off-policy aprenden a partir de experiencias generadas por una política diferente a la que se está utilizando. Esto permite al algoritmo evaluar diferentes estrategias y aprender de los errores cometidos en el pasado [Uehara et al., 2022]. Características principales:

- Aprendizaje diferido: La política se aprende a partir de datos históricos, no de la experiencia actual.
- Estimación de la función de valor: Se utiliza una función de valor para estimar la recompensa esperada de cada estado-acción.
- Mejora de la política: La política se mejora utilizando la información aprendida de la función de valor.

Capítulo 3

Descubrimiento inteligente de las topologías de redes IoT parcial o totalmente desconocidas

Capítulo 3

Descubrimiento inteligente de las topologías de redes IoT parcial o totalmente desconocidas

3.1 Introducción

En este capítulo donde se da comienzo la aportación original de esta tesis se describe el primero de los algoritmos que se han diseñado para la identificación del malware en la redes IoT. Para que este algoritmo pueda ser implementado y cumpla con su función, se asume que se cuenta con un punto de entrada, es decir, se conoce un nodo inicial de la red IoT a partir de la cual el algoritmo irá iterando y descubriendo la red. Como ya se ha comentado en la introducción, se quiere descubrir la topología de las redes IoT a partir de información parcial o nula para poder identificar los nodos que están infectados y, a su vez, identificar el malware si este es conocido. Por tanto, se puede decir que el objetivo específico de este algoritmo es descubrir la topología de la red IoT a partir de información parcial o nula y un punto de entrada. Las principales contribuciones de este capítulo son:

- Este capítulo muestra cómo se ha modelizado el problema de descubrir la topología de la red IoT mediante un problema de aprendizaje por refuerzo. Este tipo de aprendizaje es muy interesante para este tipo de problemas ya que va aprendiendo según el problema avanza. Además, no necesita conocer el entorno (la red IoT) para poder encontrar una solución al problema.
- El algoritmo basado en aprendizaje por refuerzo está definido como un problema de decisión de Markov (MDP por sus siglas en inglés). Este tipo de problemas necesita a su vez definir ciertos elementos que los componen. Este capítulo detalla todos los componentes del problema de decisión de Markov como los estados, las acciones, la función de transferencia y la función de recompensa.

- El algoritmo propuesto en este capítulo utiliza una política conocida como "greedy" ya que es la más óptima en los problemas en los que el modelo o el entorno es desconocido o incierto, que es la hipótesis de partida que tenemos sobre la información que se tiene sobre la topología de la red IoT. Además, se muestra cómo encontrar la política óptima mediante la técnica de evaluación Monte Carlo y su convergencia a la función de valor de acción óptima (es decir, la solución en la que el algoritmo ya ha descubierto la topología de la red IoT).

El resto del capítulo está organizado como sigue: en la primera parte del capítulo se presenta el nuevo algoritmo que se ha desarrollado en esta investigación, se describe el problema de decisión de Markov y se modeliza el problema del descubrimiento de la topología de la red IoT usando el problema de decisión de Markov, así como todos los elementos que lo componen. En la segunda parte del capítulo se explica cómo encontrar una solución al problema presentado en la primera parte y el fundamento teórico que permite justificar que el problema de decisión de Markov está resuelto cuando se consigue encontrar la función de valor de acción óptima a partir de una política "greedy".

3.2 Algoritmo para descubrir información en redes IoT

Tras analizar el problema de descubrir la topología de una red IoT parcial o totalmente desconocida, se decidió que la mejor forma de solucionar este problema era diseñando un algoritmo basado en aprendizaje por refuerzo. Este tipo de aprendizaje tiene la gran ventaja de que el algoritmo puede aprender de la experiencia, de la exploración del entorno o de una mezcla balanceada de ambas [Sutton et al., 1998]. Como se puede ver, en nuestro caso, según se vaya descubriendo la red IoT, el algoritmo podrá ir utilizando cada vez más la experiencia, mientras que al principio tendrá que utilizar preferentemente la exploración. Esto es así, ya que a priori no se conoce el entorno, es decir, desconocemos parcial o completamente la red IoT. En este tipo de casos, se suele utilizar principalmente estrategias de actualización de la política que no usen el modelo y, de esta forma, el algoritmo se va a entrenar (es decir, se va a buscar la solución al MDP) utilizando principalmente su experiencia [Wang and Su, 2020]. Una vez que hemos discutido el tipo de actualización de la política que vamos a usar, vamos a pasar a detallar el diseño del algoritmo. El algoritmo diseñado para identificar la topología de las redes IoT se interpreta como un agente que va recorriendo el entorno, que en nuestro caso es la red IoT, tomando una serie de acciones (A) según los estados (S) en los que se encuentra actualmente y la recompensa (R) que le indica al agente cómo de bien está tomando las decisiones tomando en cuenta el objetivo final para el que está diseñado. Esta es una arquitectura clásica en los problemas de aprendizaje por refuerzo [Sutton et al., 1998].

3.2.1 Proceso de decisión de Markov

Desde el punto de vista de las matemáticas, un MDP es un proceso de control estocástico en tiempo discreto [Bellman, 1957]. Proporciona un marco matemático para modelar la toma de decisiones en situaciones en las que los resultados son en parte aleatorios y en parte están bajo el control de quien toma la decisión [Bellman, 1957]. Los MDP son útiles para estudiar problemas de optimización resueltos mediante programación dinámica. Un MDP se define por una tupla (S, A, R, P, s_0) donde S representa el espacio de estados, A denota las acciones permitidas, R recopila las recompensas, P representa una matriz de transición de estados, y s_0 es el estado inicial. Dado un estado $s_t = s \in S$ y una acción $a_t = a \in A$ en el tiempo t, la probabilidad de alcanzar el estado $s^{'}$ en el tiempo $t + 1$ está definida por la probabilidad de transición $P(s, a, s^{'})$, que se denota como,

$$P(s, a, s^{'}) = P(S_{t+1} = s^{'} | S_t = s, a_t = a). \tag{3.1}$$

Dado un estado inicial s_0, el modelo se ejecuta continuamente de acuerdo con la dinámica del entorno definida por la ecuación (3.1) hasta el punto en que alcanza un estado objetivo. Un MDP cumple con la propiedad de Markov, que básicamente establece que el proceso futuro es independiente del pasado dado el presente. En MDPs existen dos tipos de funciones de valor, que son el valor del estado $V(s)$ y el valor del estado-acción $Q(s, a)$. La política π de un agente es un mapeo desde un estado s y una acción a a la probabilidad $\pi(a|s)$ de tomar la acción a en el estado s. Por lo tanto, la función de valor de un estado s siguiendo la política π, denotada como $V_\pi(s)$, puede considerarse como las recompensas futuras esperadas, es decir,

$$V_\pi(s) = E_\pi \left[\sum_{k=0}^{k} \gamma^k R_{t+k+1} | s_t = s \right], \tag{3.2}$$

donde γ es un factor de descuento. El valor del estado-acción de elegir una acción a en el estado s siguiendo la política π es

$$Q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{k} \gamma^k R_{t+k+1} | s_t = s, a_t = a \right]. \tag{3.3}$$

Las estrategias para resolver MDPs con un número finito de estados y acciones tienen diferentes enfoques, como la programación dinámica. Los algoritmos que se analizan en este capítulo de la tesis se centran en MDPs con un número limitado de estados y acciones, donde se conocen explícitamente las probabilidades de transición y las funciones de recompensa. Sin embargo, estos conceptos básicos se pueden adaptar para abordar otros tipos de problemas mediante la aproximación de funciones. Se puede considerar que un MDP está resuelto si se ha alcanzado una política optima [Wei et al., 2020a]. Revisando la literatura, se encuentra que para calcular políticas óptimas en MDPs con estados y acciones finitos, se utilizan algoritmos estándar que requieren el almacenamiento de dos conjuntos de datos indexados por estado: el valor V, que contiene valores reales, y la política π, que contiene acciones. Al final del proceso, π contendrá la solución ideal, mientras que $V(s)$ contendrá la suma descontada de recompensas esperadas al seguir esa solución desde el estado s [Wei et al., 2020b].

Estos algoritmos constan de dos fases: una actualización del valor y una actualización de la política, que se aplican repetidamente en todos los estados hasta que ya no se produzcan cambios. Ambos pasos actualizan recursivamente las estimaciones de la política óptima y del valor del estado utilizando valores estimados previos.

$$
\begin{aligned}
V(s) &= \sum_{s'} P_{\pi(s)}\left(s, s'\right)\left(R_{\pi(s)}\left(s, s'\right) + \gamma V\left(s'\right)\right), \\
\pi(s) &= \operatorname{argmax}_a \left\{ \sum_{s'} P_a\left(s, s'\right)\left(R_a\left(s, s'\right) + \gamma V\left(s'\right)\right) \right\}.
\end{aligned}
\tag{3.4}
$$

Su orden depende de la variante del algoritmo; también se pueden hacer para todos los estados a la vez o estado por estado, y más a menudo a unos estados que a otros.

Una de las variantes más utilizadas es la iteración de valores presentada por Bellman [Bellman, 1957]. En este proceso, también conocido como inducción hacia atrás, la función π no se utiliza; en cambio, el valor de $\pi(s)$ se calcula dentro de $V(s)$ cada vez que es necesario. Sustituir el cálculo de $\pi(s)$ en el cálculo de $V(s)$ da como resultado el paso combinado:

$$
V_{i+1}(s) = \max_a \left\{ \sum_{s'} P_a\left(s, s'\right)\left(R_a\left(s, s'\right) + \gamma V_i\left(s'\right)\right) \right\}, \tag{3.5}
$$

donde i es el número de iteraciones. La iteración de valor comienza en $i = 0$ y utiliza V_0 como una suposición inicial de la función de valor. Luego itera, calculando repetidamente V_{i+1} para todos los estados s, hasta que V converge con el lado izquierdo igual al lado derecho (esta es la "ecuación de Bellman" para este problema).

Existen algoritmos capaces de hallar políticas óptimas en MDPs finitos con una complejidad computacional que sigue un comportamiento polinómico en relación al tamaño de la representación del problema [Abbasi et al., 2021, Agarwal et al., 2021]. Esto coloca los problemas de decisión basados en MDPs dentro de la clase de complejidad computacional P. No obstante, el problema de la dimensionalidad afecta considerablemente, ya que el tamaño de la representación del problema tiende a crecer de manera exponencial conforme aumenta el número de variables de estado y acción. Esta situación limita las técnicas de solución exacta a problemas con una representación más compacta.

En la práctica, técnicas de planificación en línea como la búsqueda basada en el árbol de Monte Carlo pueden ser efectivas para encontrar soluciones útiles en problemas más extensos. Además, en teoría, es viable desarrollar algoritmos de planificación en línea capaces de encontrar políticas casi óptimas sin depender de la complejidad computacional asociada al tamaño del espacio de estados [Kearns et al., 2002].

3.2.2 Modelización del problema usando MDP

Partiendo de la hipótesis de que no tenemos ningún conocimiento de la red IoT que se quiere explorar. Y del hecho de que necesitamos uno o varios puntos de entrada en la red IoT para poder desplegar nuestro algoritmo. Se ha diseñado un algoritmo basado en MDP que sea capaz de recorrer la red IoT desconocida, de forma que haga un mapa de la misma y, si es posible, identifique cada uno de los nodos de la red IoT y las conexiones existentes. El diseño de algoritmos basados en RL consiste en modelizar el problema identificando los estados, acciones y la función de recompensa. Dentro del campo del aprendizaje por refuerzo, este tipo de algoritmos basados en MDP necesitan tener identificada y modelizada claramente la tupla que define al MDP. Hay que destacar que el algoritmo, desde un punto de vista formal, está compuesto por el entorno, por el agente que se mueve por el entorno, por las acciones que puede realizar el agente, por los estados que tiene cada uno de los lugares del entorno que recorre el agente y todo ello guiado por la recompensa que recibirá el agente por cada acción que haga. A continuación se definen los elementos necesarios para definir el MPD.

3.2.2.1 Estados

El primer elemento que se necesita modelizar para poder definir un MDP son los estados que tiene el problema que se quiere definir. En nuestro caso, como se quiere identificar la topología de una red IoT desconocida, se han identificado los siguientes componentes que se deben tener en cuenta en el diseño del estado. En primer lugar, dado que se quiere identificar una red IoT que se ha modelizado mediante un grafo, se va a necesitar conocer la posición del agente que está explorando el grafo. Esta posición se refiere al nodo del grafo en el que está el agente. Por otra parte, dado que se asume que la red está protegida por niveles de acceso, otra componente del estado debe ser el nivel de permiso que tiene el nodo. Esto es importante ya que si se detecta un nodo del que se puede extraer información libremente, el algoritmo lo va a explotar, mientras que si se detecta un nodo que necesita un nivel de permiso muy elevado para extraer información, el algoritmo lo va a obviar y continuará su camino habiendo obtenido información parcial de ese nodo. En resumen, la propuesta de estado para nuestro algoritmo, $s = (p_t, l_t, t)$, consiste en tres componentes, llamadas posición del grafo $p_t \in P = \{\text{conjunto finito de posiciones}\}$, nivel de permiso $l_t \in L = \{\text{niveles de permiso}\}$, y $t \in T$ el tiempo actual. Para determinar las posibilidades una vez que el malware exploratorio llega a un nodo específico, creamos un conjunto finito de niveles de permiso L dentro del propio nodo, independientemente del tipo de equipo que sea. En este algoritmo, hemos diseñado los diferentes niveles de permiso sobre la red que tiene un usuario de una computadora conectada a la red.

El conjunto de cuatro niveles de permisos se define de la siguiente manera: En el primer nivel, al que llamaremos nivel base, asumimos que simplemente alcanzamos el equipo y no tenemos ningún tipo de permiso más allá de poder interactuar con él. El segundo nivel, que llamaremos nivel invitado, asume que tenemos acceso a nivel de invitado al equipo específico. Esto nos permite obtener información muy limitada sobre él, pero es un avance en términos de permisos sobre el nivel anterior; El tercer nivel, que llamaremos nivel de permisos básicos, asumimos que tenemos cierta capacidad operativa dentro del equipo. Esto nos permite acceder a cierta información, como las interfaces de red del equipo, direcciones IP y máscaras de red. Esto nos permite, con el uso de protocolos de nivel de red como ICMP, saber qué otros equipos existen dentro de la topología. El último y cuarto nivel, que llamaremos nivel de root, nos permite tener acceso global a toda la computadora. Esto significa tener acceso a cualquier tipo de comando o al uso de analizadores de red que nos permitirían conocer el tráfico global existente en este equipo y, en consecuencia, aquellos equipos que tienen comunicación directa con él. Por ejemplo, $s = (3, 2, 5)$ es un estado en el que el malware está en el tercer nodo, con nivel de permisos de invitado (nivel 2) cuando han pasado 5 "pasos de tiempo", es decir, $t = 5$.

Como se puede ver en la definición del estado, las componentes, posición y permisos dependen del tiempo. Esto puede parecer contradictorio debido a que el nivel de permisos tendría que mantenerse constante durante toda la simulación. Pero, se considera que el algoritmo, al descubrir la red IoT y explorar la información de cada uno de los nodos, puede llegar a conseguir niveles de prioridad más altos en los permisos en la red y, por tanto, al volver a pasar por alguno de los nodos ya conocidos o incluso los nuevos, puede llegar a tener más prioridad, lo que significará que podrá obtener más información de aquellos nodos que tengan un nivel de permiso menor que el que tenga el algoritmo. Sin embargo, la manera de obtener esta información (es decir, pentesting) queda fuera del alcance de esta investigación y vamos a suponer que es un trabajo automático.

3.2.2.2 Acciones

El conjunto de acciones permitidas en un MDP se representa como A. Estas acciones varían según el número de nodos vecinos disponibles desde el nodo actual. En este escenario, el agente que está explorando la red IoT debe seleccionar el nodo al que desee desplazarse entre sus vecinos. En este paso hemos asumido que, aun teniendo el nivel mínimo de permiso en la red, el agente tiene acceso a conocer todas las conexiones que tiene el nodo actual de la red IoT en el que se encuentra. Es importante señalar que algunos movimientos pueden resultar inalcanzables porque por la propia estructura del grafo no sean posibles. Por ello, se introduce una penalización significativa en la

recompensa para desincentivar al agente de tomar acciones que lo lleven a una red vecina inalcanzable. Los movimientos permitidos se realizan entre dos nodos conectados por una arista. La premisa que hace posible esto, es una de las limitaciones de diseño de este algoritmo, y en trabajo futuro queremos diseñarla e implementarla. Además, es importante indicar que el conjunto de acciones A es finito, ya que si no fuera finito, la resolución del MDP podría necesitar un poder de cómputo demasiado alto [Adler and Subramanian, 2024].

Cuando se efectúa un desplazamiento hacia un nuevo nodo en la red IoT, se lleva a cabo el pentesting para determinar el estado de permisos de dicho nodo. El pentesting consiste en una serie de pruebas destinadas a identificar vulnerabilidades de seguridad dentro del sistema, lo que potencialmente podría proporcionar un mayor nivel de privilegios dentro del sistema. Sin embargo, es importante notar que al igual que el pentesting puede resultar en la obtención de mayores privilegios, también puede ser detectado, lo que resultaría en la expulsión del sistema o en la reducción forzada de dichos privilegios [Bella et al., 2023]. Como ya se ha comentado en el apartado anterior, consideramos que el proceso de pentesting es automático, aunque, vamos a tener en cuenta que el agente va a realizar repetidamente este proceso en la red, y dado que, cuantas más veces se haga, más probabilidades hay de que sea detectado, vamos a penalizar al algoritmo que pase mucho tiempo en la red IoT. Esto lo haremos en la definición de la función de recompensa en la sección 3.2.2.4.

3.2.2.3 Transición de estados

Ya tenemos definidos los estados del entorno y las acciones que puede realizar el agente. Por tanto, lo que tenemos que definir a continuación son las reglas que van a permitir al agente realizar acciones y por tanto, moverse por el entorno. En el aprendizaje por refuerzo esto se conoce como proceso de transición. En este apartado se detalla el proceso de transición de estados que describe la evolución de los estados en nuestro MDP. En la sección 3.2.2.1 se establece que los estados en nuestro MDP se representan como una tupla $s = (p_t, l_t, t)$, donde p_t refiere a la posición en el grafo, l_t indica el nivel de permiso y t representa el tiempo actual. A pesar de que t cambia siguiendo sus propias reglas, tanto p_t como l_t están sujetos a algún proceso estocástico para determinar el siguiente estado.

Aunque podría asumirse que la transición de p_t a p_{t+1} es simple, en realidad, no se tiene información sobre los nodos siguientes en el grafo hasta que se descubre un nuevo nodo. Al llegar a un nodo nuevo, se descubren todos los nodos vinculados a él. Bajo las limitaciones actuales que hemos identificado en nuestra investigación, la transición

del estado de p_t al estado p_{t+1} se ve influenciada por las acciones del agente (a su vez influenciadas por la función de recompensa). Por otro lado, los estados alcanzados durante las pruebas de pentesting, donde la probabilidad de cada estado depende únicamente del estado previo en la prueba de pentesting anterior. Durante estas pruebas, la probabilidad de avanzar de nivel disminuye a medida que los privilegios aumentan, y también existe la posibilidad de permanecer en el mismo nivel o ser expulsado del dispositivo, regresando así al nivel base. Como ya hemos establecido a lo largo de este capítulo, el proceso de pentesting en sí mismo debería ser una acción que hiciera el agente, pero dado que hemos asumido que es una acción automática, no la tenemos en cuenta para el diseño de la función de transición entre estados.

Dado que el objetivo principal de esta parte de la investigación desarrollada en este estudio es descubrir la topología de la red IoT, se ha considerado que siempre hay una posibilidad de avanzar desde el nivel base hasta el nivel de máxima autorización a medida que vaya avanzando la simulación en el tiempo. En caso contrario, en el momento de que se encontrase un nodo con un nivel de permiso superior al que tiene el agente, este se quedaría bloqueado sin poder avanzar en dicho nodo.

3.2.2.4 Función de recompensa

En las estrategias basadas en aprendizaje por refuerzo, la falta de exploración en entornos no estructurados representa un desafío significativo. Esta limitación puede hacer que los problemas de planificación de rutas sean menos eficientes y carezcan de robustez [Xie et al., 2019]. Para abordar este desafío, desarrollaremos una función de recompensa que considere todos los posibles efectos que puedan influir de manera significativa en la tarea de aprendizaje del algoritmo. En un entorno no estructurado, el agente tiene la tarea de descubrir todos los nodos de una red IoT desconocida. Por lo tanto, nuestra función de recompensa se basará en el número de nodos aún desconocidos, si un nodo ha sido visitado más de una vez y el tiempo transcurrido desde el inicio del algoritmo.

Inspirados en el concepto del delta de Kronecker, diseñamos una función que representa el número de nodos desconocidos restantes en la red en el tiempo t. U_n denota el conjunto de nodos desconocidos aún por descubrir en el tiempo t, V_n indica los nodos visitados más de una vez, y n_t representa el nuevo nodo en el tiempo t. La función se modela como se indica en la siguiente ecuación:

$$f_{unknown}(n_t, U_n, V_n) = \begin{cases} 100 & \text{si} \quad n_t \in U_n \\ \\ -10\sqrt{x} & \text{si} \quad n_t \notin U_n \end{cases} \tag{3.6}$$

donde $x \in V_n$ representa el número de veces que se ha visitado un nodo.

Al combinar nodos desconocidos y nodos visitados repetidamente, definimos nuestra función de recompensa como:

$$R(n_t, U_n, V_n, t) = f_{unknown}(n_t, U_n, V_n) - t \tag{3.7}$$

Hemos incorporado el tiempo como una variable, ya que buscamos modelar el riesgo de ser detectados por las defensas cibernéticas de la red en función del tiempo que el malware permanece en la red mientras descubre todos sus nodos.

3.2.3 Solución del MDP

Se parte de la hipótesis de que la red es desconocida o al menos incierta, es decir, se puede partir del conocimiento parcial de la topología de la red IoT. Entonces, para descubrir la red IoT con el algoritmo que hemos diseñado, es necesario encontrar una solución al MDP. Aunque el modelo o entorno es desconocido o incierto, lo que se puede hacer en estos casos es muestrear la experiencia del agente. En este caso, se suelen elegir los enfoques de control sin modelo para el diseño final del algoritmo de aprendizaje por refuerzo. Este tipo de método es el más óptimo cuando el MDP es desconocido o incierto. Sea V la función de valor de acción y sea π la política, actualizaremos la evaluación de la política con la evaluación de políticas de Monte Carlo, donde $V = v_\pi$. Por lo tanto, debemos estimar v_π.

Dado que $Q(s, a)$ es libre de modelo, nos enfrentamos a la estimación de v_π con la mejora de políticas codiciosas.

$$\pi'(s) = \text{argmax } Q(s, a) \quad \text{con } a \in A, s \in S. \tag{3.8}$$

Este problema es similar a resolver la evaluación de políticas con Monte Carlo (es decir, $Q = q_\pi$). De manera equivalente, partiendo de Q, π, con $Q = q_\pi$ y $\pi = \epsilon-$greedy, alcanzaremos q_*, π_* el q y π óptimos. En cada paso, utilizamos la política "Greedy" al hacer el límite con Exploración Infinita (GLIE), donde todas las acciones-estados son exploradas muchas veces:

$$\lim_{k \to \infty} N_k(s, a) = \infty \tag{3.9}$$

luego la política converge como

$$\lim_{k \to \infty} \pi_k(a|s) = 1 \tag{3.10}$$

con $a = \text{argmax}\ Q_k(s, a')$.

Muestreamos el k-ésimo episodio usando $\pi : \{S_1, A_1, R_1, \dots\} \sim \pi$. Para cada estado S_t y acción A_t en cada episodio:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1 \tag{3.11}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t)) \tag{3.12}$$

donde G_t es la recompensa acumulada, $\epsilon \leftarrow \frac{1}{k}$, $\pi \leftarrow \epsilon-\text{greedy}(Q)$.

Teorema 3.1. *El control de Monte Carlo GLIE converge a la función de valor de acción óptima*

$$\lim_{t\to\infty} Q(s, a) \to q_*(s, a) \tag{3.13}$$

Proof. Ver [Sutton et al., 1998]. □

Capítulo 4

Identificación y analisis de nodos de la red IoT infectados por malware

Capítulo 4

Identificación y analisis de nodos de la red IoT infectados por malware

4.1 Introducción

En este capítulo nos centramos en diseñar una solución para la segunda parte del objetivo general de esta tesis, es decir, identificar los nodos de la red IoT que están infectados, y en último lugar identificar el malware que está infectando dichos nodos (en caso de que sea conocido). Por tanto, hemos dividido el capítulo en dos partes en las que abordaremos cada uno de estos subobjetivos. En la primera parte vamos a analizar el problema de la detección de los nodos infectados en las redes IoT. Para ello, volviendo a hacer uso del aprendizaje por refuerzo, se analiza el problema de identificar los nodos de la red IoT infectados. Una vez hemos reflexionado sobre cómo aplicar el aprendizaje por refuerzo para resolver esta problemática, se modeliza el problema usando un MDP y se amplía el problema sin pérdida de generalidad a un sistema multi-agente coordinado que permite al algoritmo ser más eficiente en relación al tiempo que tarda en explorar toda la red IoT. Mientras que en la segunda parte se diseñan dos metodologías para la identificación del malware presente en una red IoT, una basada en datos reales y machine learning y otra basada en datos sintéticos y técnicas de estimación de parámetros. La primera metodología está basada en aprendizaje supervisado para tratar de identificar el malware que se ha encontrado en dicho nodo IoT. Para este fin, se ha revisado la literatura para encontrar las mejores técnicas de análisis de datos y preprocesado de datos. Además, basándonos en trabajos relacionados en este mismo campo, se crea un conjunto de algoritmos de machine learning que se usa para realizar una comparativa del rendimiento de todos ellos en el caso de la detección de un ataque malware en concreto. Esta metodología está automatizada para que, proporcionándole un nuevo dataset con un formato similar, pueda entrenarse automáticamente y encontrar el algoritmo que

mejor rendimiento tiene para cada uno de los diferentes malwares. Como se puede ver, se ha decidido solo hacer el caso de estudio para un solo tipo de ataque malware, aunque una vez hecha la metodología y automatizada, resulta trivial cambiar el dataset con nuevos ataques malware para su uso. Por otro lado, la segunda metodología está basada en modelos epidemiológicos para generar datos sintéticos. Sobre estos datos se resuelve el problema inverso y se estiman los parámetros de los modelos epidemiológicos y de esta forma se pueden identificar los tipos de especímenes de malware. Esta segunda metodología, basada en datos sintéticos, tiene la limitación de que para poder diseñar los algoritmos y evaluar su efectividad se han tenido que usar modelos epidemiológicos que posteriormente se les realizaba el problema inverso para identificarlos. Además, tiene la misma limitación que la otra metodología diseñada para este mismo fin. Si no se conoce el modelo, al realizar la estimación de parámetros, no se va a poder identificar el modelo epidemiológico con total precisión. En resuemen, las principales contribuciones de este capítulo son:

- Se diseña una solución al problema de la identificación de los nodos infectados en una red IoT utilizando un sistema multiagente basado en aprendizaje por refuerzo. Este enfoque permite coordinar a varios agentes para explorar la red (asumiendo que es conocida) y detectar los nodos infectados de manera más eficiente. Esta selección se justifica debido a la necesidad de reducir el tiempo de detección de los nodos infectados en comparación con enfoques de un solo agente.

- Se ha formalizado el problema de detección de nodos infectados en una red IoT como un MDP y se ha propuesto un método de solución basado en N-step bootstrapping. Esto implica modelar el problema de detección de nodos infectados como un MDP, donde los agentes toman acciones en función de estados y reciben recompensas. La técnica de N-step bootstrapping se utiliza para estimar el valor óptimo de la función de valor de acción q_*, lo que permite entrenar al algoritmo para maximizar la detección de nodos infectados en la red IoT.

- Se ha diseñado una metodología integral detallada para la adquisición de datos que reflejen los escenarios de tráfico de red en contexto IoT en sistemas virtualizados. Además, esta metodología hace uso de recursos ya existentes como el dataset IoT-23 que ha servido para inspirar el diseño de la adquisición y captura de datos. Se propone una serie de técnicas incluidas en la metodología para la gestión de los datos, tratamiento de datasets desbalanceados y eliminar la multicolinearidad, entre otras cosas. La metodología se ha automatizado para que de forma semisupervisada pueda gestionar cualquier tipo de datos similares de otros ataques malware.

- Se han seleccionado de forma exhaustiva diferentes algoritmos de clasificación binaria basados en machine learning para la identificación del malware en las redes IoT. Se propone un conjunto de clasificadores basados en diversas técnicas como lineales, probabilísticas o de ensemble. Además, se han seleccionado las técnicas más habituales de la literatura para evaluar el rendimiento de este conjunto de algoritmos de clasificación y determinar de esta manera cuál es el que tiene mejor desempeño para cada uno de los malwares que se quiere usar.

- Se presenta una metodología que utiliza modelos epidemiológicos compartimentales globales, como el modelo SIR, para generar datos sintéticos de la propagación del malware en redes IoT. Luego, se propone resolver el problema inverso de estimación de parámetros para identificar el malware que se está propagando. Esta metodología combina técnicas de Monte Carlo y PINNs para estimar los parámetros del modelo epidemiológico que mejor se ajusten a los datos observados.

- En este capítulo se describe cómo se pueden utilizar las PINNs para estimar los parámetros de modelos epidemiológicos, como el modelo SIR, a partir de datos incompletos, es decir, solo teniendo información de los infectados de la red IoT. Esta técnica permite aprender la dinámica de transmisión del malware representada por los valores de los parámetros SIR, β y γ (en el caso del modelo SIR) utilizando conjuntos de datos incompletos. La optimización (es decir, el proceso de identificación del malware) se realiza minimizando una función de pérdida que incluye tanto los datos observados como las ecuaciones diferenciales que describen el modelo.

El resto del capítulo está organizado como sigue: en primer lugar, se encuentra la sección 4.2 en la que se aborda el diseño del algoritmo para descubrir los nodos infectados de la red IoT. En esta sección cabe destacar la formalización y modelización del problema usando MDP, y el trasfondo teórico aplicado para la solución del MDP. En segundo lugar, se tiene la sección 4.3 en la que se detalla la metodología para la identificación del malware en las redes IoT. La metodología se compone de una subsección de análisis y adquisición de los datos representativos, junto con una descripción del dataset construido. Técnicas de preprocesado de datos y una selección de los algoritmos que mejor desempeño tienen en la literatura en problemas de clasificación binaria. Además, se propone una selección de métricas para evaluar el rendimiento del conjunto de algoritmos seleccionado. En tercer lugar, se tiene la sección 4.4 en la que se detalla la metodología para la identificación del malware a partir de datos sintéticos. La metodología se compone de la generación de datos sintéticos de entrenamiento usando modelos epidemiológicos, la resolución del problema inverso para la estimación de parámetros.

4.2 Algoritmo de aprendizaje por refuerzo multiagente para el descubrimiento de malware en redes IoT

Una vez se cuenta con la información total o parcial de la red IoT con la que se está trabajando gracias al algoritmo descrito en el capítulo anterior, y motivados por el objetivo de identificar los nodos IoT que están infectados, necesitamos diseñar una solución que sea capaz de recorrer toda la red IoT en el menor tiempo posible, de forma que se puedan identificar estos nodos afectados en el menor tiempo posible y de esta forma, los encargados de la ciberseguridad de la red IoT puedan aplicar contramedidas o medidas de mitigación. En el planteamiento de este desafío, e inspirados por el diseño del problema anterior de descubrir la topología de la red IoT, pensamos que se podría diseñar un algoritmo basado en aprendizaje por refuerzo de forma que solucionara esta situación. Pero una vez estaba planteado el desafío y diseñada su solución, en la primera ronda de simulaciones se observó una complicación que surgía de la propia naturaleza del escenario y de la urgencia de identificar estos nodos afectados lo más rápidamente posible. El tiempo de simulación era bastante grande, en concreto, necesitaba más pasos de ejecución que como mínimo el número de nodos que tenía la red IoT. Rapidamente, uno se da cuenta de que si hubiera más de un agente recorriendo la red IoT e identificando estos nodos infectados, se reducirían mucho los pasos de tiempo que necesita el algoritmo para completar su tarea. Sin embargo, surge la necesidad de que estos agentes estén coordinados para que todos colaboren en la consecución del objetivo final. En el campo del aprendizaje por refuerzo, esto se conoce como aprendizaje por refuerzo con múltiples agentes o MARL por sus siglas en inglés [Oroojlooy and Hajinezhad, 2023]. Nos hemos decidido por esta técnica ya que desde un punto de vista tecnológico da continuidad al algoritmo diseñado en el capítulo anterior y por tanto, en trabajos futuros, se desearía integrar ambos algoritmos en un MARL que fuera capaz de descubrir la topología de una red desconocida mientras que a su vez, es capaz de identificar los nodos de la red IoT que están infectados con malware. Sin embargo, hay otras tecnologías en la literatura ampliamente utilizadas en el campo de la colaboración automatica, como por ejemplo los algoritmos basados en colonias de hormigas [Miao et al., 2021] o los algoritmos basados en optimización de enjambres de avejas [Gad, 2022]. Estos últimos están muy de actualidad hoy en día ya que son los que se utilizan para controlar enjambres de drones.

Por tanto, una vez expuestos los motivos por los que se ha decidido utilizar los MARL para modelizar y resolver este problema, el siguiente paso es realizar la formalización matemática del problema. En las siguientes subsecciones vamos a proceder a detallar toda la información necesaria sobre el nuevo algoritmo que hemos diseñado. En concreto,

desde un punto de vista matemático, vamos a describir el problema de la identificación de los nodos IoT infectados, a continuación vamos a revisar los dos métodos principales para estimar q_*, es decir, encontrar el valor óptimo de la función de valor o lo que es lo mismo, encontrar la solución al problema que estamos proponiendo. Para la estimación del q_* vamos a proponer la técnica de N-steps bootstrapping que es una mezcla de las técnicas Monte Carlo usadas en el capítulo anterior y las técnicas de diferencia temporal de un paso usadas habitualmente en los algoritmos en los que sí se conoce el modelo o entorno. Desde un punto de vista del diseño del algoritmo, en primer lugar definimos el MDP para un solo agente, y en la segunda parte de esta sección generalizamos al sistema MARL.

4.2.1 Formalización del problema

Consideremos una red de IoT con una amplia variedad de dispositivos. Para abordar el problema planteado en esta investigación de identificar los nodos de la red IoT infectados, consideramos que la red de IoT está representada por el grafo $G = (V, E)$, donde V es el conjunto de vértices y E es el conjunto de aristas. En este contexto, los vértices representan los sensores y las aristas representan las conexiones de comunicación entre estos sensores. En nuestro análisis, hemos empleado grafos no dirigidos y hemos supuesto que las comunicaciones permanecen estables a lo largo del estudio, es decir, no hay alteraciones en la conexión. La propagación del malware eventualmente convertirá la red de IoT en una botnet, lo que permitirá al malware atacar otras redes de IoT conectadas, así como redes informáticas y diversas fuentes de información [Ali et al., 2020a]. Dado que el enfoque de esta investigación es la detección de los nodos infectados por el malware en la red de IoT a medida que se propaga, si se ha conseguido detectar a tiempo, en caso contrario, el algoritmo se ejecutaría en cuanto se tenga constancia del malware. Cuanto antes se detecten todos los nodos infectados, antes se podrán aplicar técnicas antimalware para mitigar o detener su propagación. Además, hemos asumido que tenemos un conocimiento completo de la arquitectura de la red de IoT, es decir, conocemos el grafo, y suponemos que el malware se propaga siguiendo un modelo determinista individual. Tenemos que asumir esto, porque aunque en el capítulo anterior se ha diseñado un algoritmo capaz de descubrir la topología de una red IoT, la integración de ambos algoritmos no es un objetivo actual de esta investigación. Por otra parte, mantenemos la limitación de que el proceso de pentesting o de detección de malware en cada nodo se considere automático [ElSawy et al., 2020, Zhaikhan et al., 2020].

Para continuar, asumiremos que la red IoT considerada opera en tiempo discreto, donde el eje del tiempo se divide en enteros de tiempo iguales y no superpuestos (pasos). Sea

t un índice de paso de tiempo entero. En particular, el agente mantiene la detección de malware durante un intervalo de tiempo fijo de $T > 1$ pasos, que se denomina período de detección. También consideramos el período de decisión. Consideremos la siguiente estrategia para los agentes: cualquier agente asignado a identificar nodos de IoT con malware en el paso de tiempo t comienza a analizar el malware y, una vez que ha terminado esta acción, decide cuál es el nodo más cercano al que moverse. Por lo tanto, el agente debe terminar sus operaciones de detección y decisión para el final de su período de detección, es decir, en el paso de tiempo $t + T$. Finalmente, hemos identificado en los modelos teóricos que las tasas de propagación de malware dentro de la red de IoT se consideran extremadamente rápidas para que un solo algoritmo pueda detectarlo en tiempo real. Por lo tanto, esto nos ha motivado a diseñar un algoritmo de aprendizaje por refuerzo de múltiples agentes para optimizar el rendimiento de detección del algoritmo a través del uso de n agentes que pueden abarcar la red y detectar el malware antes de que la red alcance un punto crítico donde no se puedan utilizar contramedidas para eliminar el malware de la red de IoT y esta se transforme en una Botnet.

4.2.2 Modelización del problema usando MDP

Nótese que el objetivo del agente es detectar los nodos de IoT infectados por el malware antes de que este se propague por toda la red y la transforme en una botnet. Por lo tanto, considerando el hecho de que la función de recompensa define el objetivo del problema de aprendizaje, se puede concluir que la recompensa nos indica cuáles son las acciones buenas o malas que el agente puede realizar [Sutton and Barto, 2018]. Para una detección confiable de nodos infectados, es necesario incluir en el diseño del algoritmo de aprendizaje la restricción de que el número de nodos explorados sea mayor que el número de nodos infectados. Matemáticamente, podemos expresarlo de la siguiente manera

$$V_E(t) \geq V_I(t) \tag{4.1}$$

donde V_E es el conjunto de nodos explorados de la red IoT y V_I es el conjunto de nodos infectados.

El MDP en este contexto se puede definir como una tupla (S, A, P_a, R_a) donde S es el conjunto de estados, A es el conjunto de las acciones, P_a es la función de transición del agente y R_a es la recompensa para el agente (nótese que hemos añadido un subíndice "a" tanto en la función de transición como en la recompensa, esto es debido a que cada uno de los agentes del MARL tendrá su propia función de transición y de recompensa en cada tiempo t). Debido a que en el capítulo anterior detallamos completamente el

MDP, en este caso, vamos a proceder a un diseño más dinámico solo detallando las componentes importantes. Para el entorno, en el problema de propagación de malware, consideraremos un grafo para describir la red IoT, mientras que también consideraremos una matriz de adyacencia para representar la red. Por lo tanto, cada estado $S = \{M_{adj}, ID, VT, I, D\}$ tendrá la siguiente información: matriz de adyacencia, ID del nodo IoT (ID), número de veces que el agente ha visitado el nodo (VT), si el nodo está infectado o no (I), y si el agente ha detectado que el nodo está infectado (D). Sea A el conjunto de acciones. El agente puede realizar dos acciones: verificar el estado del nodo donde se encuentra el agente y moverse al siguiente nodo. En ambos casos, el agente consulta el estado del entorno usando observaciones del entorno. En la primera acción, el agente decidirá si el nodo de IoT está infectado o no, y en caso de estar infectado, cambiará su estado a detectado. Mientras el agente se mueve, debe verificar en la matriz de adyacencia que el nodo al que desea moverse sea accesible desde el nodo actual. Por otro lado, podemos definir la función de probabilidad de transición que depende de las acciones del agente como $P_a = P(S_{t+1}|S_t, A_t)$. Esta función denota la probabilidad de transición al siguiente estado S_{t+1} desde el estado S_t aplicando la acción $a \in A$.

En el paso de tiempo t, si se satisface la restricción de movimiento, el agente recibe una recompensa $R(t)$ definida como la diferencia entre los nodos infectados y los nodos detectados. De lo contrario, el agente recibe recompensas cero.

$$R(t) = \begin{cases} \log(|V_I(t) - V_D(t)|) & \text{si} \quad V_E(t) \geq V_I(t) \\ \\ 0, & \text{en otro caso} \end{cases} \tag{4.2}$$

donde V_D es el conjunto de nodos detectados infectados por malware. Es importante destacar que para cada problema que se resuelve con aprendizaje por refuerzo, puede haber una función de recompensa totalmente diferente. En el capítulo anterior, la función de recompensa del algoritmo que tiene que descubrir la topología de la red IoT (3.6) está diseñada para penalizar los movimientos que no interesa que haga el agente restando valor a la recompensa. De esta forma, cuando el agente se entrene basado en su experiencia, verá que esos caminos no son óptimos. Mientras que en este caso, los movimientos que no resultan interesantes para la detección de nodos IoT infectados se han decidido que no tengan impacto en el comportamiento del agente con castigos, es decir, solo se recompensa positivamente que detecte los nodos infectados.

Cada acción realizada por un agente tiene una recompensa asociada. Y en cada paso de tiempo t, hay una recompensa. Luego se genera una secuencia de recompensas $R_{t+1}, R_{t+2}, ..., R_T$ donde T es el último paso de tiempo. El objetivo de aprendizaje del

agente es, generalmente, maximizar la suma de estas recompensas. Así, el agente intenta seleccionar acciones de tal manera que su suma futura se maximice. Para modelar esto, introducimos el concepto de descuento. En particular, si el agente selecciona la acción A_t, el agente debe maximizar el retorno descontado esperado.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4.3}$$

donde γ es un parámetro $0 \leq \gamma \leq 1$, llamado la tasa de descuento.

Los problemas de aprendizaje por refuerzo requieren la estimación de funciones de valor, funciones que estiman qué tan buena sería la recompensa futura de una acción que realiza un agente en un estado en el paso de tiempo t. Estas funciones de valor se definen con respecto a la forma en que los agentes actúan, esto se conoce como una política. Formalmente, una política π es un mapeo de estados a probabilidades de seleccionar cada posible acción. Si el agente está siguiendo la política π en el tiempo t, entonces $\pi(a|s)$ es la probabilidad de que $A_t = a$ si $S_t = s$ [Sutton and Barto, 2018]. En este caso, dado que la función de recompensa funciona de manera distinta a la del capítulo anterior, es decir, en este caso solo se premian las acciones que interesan, tenemos que añadir estos dos nuevos conceptos que son la tasa de descuento (γ) y el retorno descontado esperado (G), que nos van a permitir entrenar al algoritmo con distintas técnicas que explicaremos a lo largo de esta misma sección. El concepto detrás de este entrenamiento es encontrar una política esperada (desde el punto de vista estadístico de la esperanza) que maximice la suma de las recompensas (G) en cada instante de tiempo. Para ello, un atajo común es definir el valor de realizar una acción $a \in A$ en un estado $s \in S$ bajo la política π, denotado como $q_\pi(s, a)$ como

$$q_\pi(s, a) = E_\pi[G_t | A_t = a, S_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a] \tag{4.4}$$

donde q_π se llama función de valor de acción para la política π. Resolver un problema mediante el aprendizaje por refuerzo, en un enfoque amplio, consiste en buscar una política que tenga la máxima recompensa esperada a largo plazo. En MDP finitos, si consideramos la política óptima, nos permite calcular la función de valor de acción, llamada q_*, que se define como:

$$q_*(s, a) = \max_{\pi} \; q_\pi(s, a) \tag{4.5}$$

para todo $s \in S$ y $a \in A$. Desde un punto de vista teórico, encontrar la solución a este problema consiste en estimar q_*. Esto se puede ampliar a la solución del problema en el caso MARL. Aun resolver estos problemas

MARL en la práctica es muy complicado, gracias al trabajo que la empresa OpenAI (creadora de ChatGPT) ha desarrollado un entorno llamado Gymnasium (https://gymnasium.farama.org/) junto con Deepmind (creadores de Gemini) de Google (https://github.com/google-deepmind). En estos entornos, se nos dan programados algoritmos de aprendizaje por refuerzo y MARL. Dejando a los investigadores la tarea de elegir el más adecuado a sus necesidades, solo habría que programar las acciones, estados, recompensa y funciones de transición. Pero la gran ventaja que ofrecen estos entornos es que nos dan algoritmos de entrenamiento MARL como por ejemplo (https://github.com/google-deepmind/acme/tree/master/examples/multiagent/multigrid). A continuación vamos a discutir desde un punto de vista teórico los retos que supone ampliar el problema de un agente a un MARL.

En este sistema multiagente, cada agente tiene el objetivo de identificar todos los nodos infectados en la red IoT que estén en su camino. Un agente individual sigue un proceso específico para lograrlo . Sin embargo, debido a la necesidad de analizar toda la red antes de que el malware se propague, se requiere una gran cantidad de agentes trabajando en conjunto.

Las decisiones tomadas por los agentes sobre qué acciones realizar se pueden modelar como un MDP. En este contexto, los agentes colaboran para encontrar la mejor acción colectiva para el sistema multiagente. Por tanto, se espera que cada agente maximice su recompensa individual. Aun así, la colaboración asegura la maximización de la recompensa esperada para todo el sistema en cada paso del tiempo. Una solución a este problema MDP en cada paso de tiempo se logra mediante el equilibrio de Nash. Este equilibrio define un conjunto de estrategias, una para cada agente, donde cada estrategia individual es óptima para los demás agentes en su conjunto. Si bien la propiedad de Markov se utiliza para modelar los entornos en los problemas MDP, las recompensas de los agentes solo se basan en las acciones que toman en el paso actual. Un MDP para un agente está compuesto por un conjunto finito de acciones, un conjunto finito de estados y la función de transición de estado. A continuación, describiremos el algoritmo para resolver el MDP para un solo agente. Este enfoque, sin pérdida de generalidad, se puede extender a los n agentes que participan en la detección de nodos infectados. Como ya se ha comentado, se utilizarán algoritmos ya entrenados de de Deepmind como por ejemplo Deep Q-learning [Zhang and Wang, 2024]. Esto nos permite explorar los fundamentos teóricos para entender cómo funciona, pero utilizando algoritmos de entrenamiento ya contrastados y que están teniendo algunos resultados tan impresionantes como conseguir jugar a juegos online y ganar a los grandes maestros de estos, juegos de Go, StarCraft II y Atari entre otros [Mnih et al., 2013, Silver et al., 2016, Vinyals et al., 2019].

4.2.2.1 Arquitectura MARL

La mayoría de los algoritmos utilizados en aprendizaje por refuerzo (Q-learning, Deep Q-learning (DQN), DQN Actor-Critic, etc.) involucran a un único agente [Oroojlooy and Hajinezhad, 2023]. Sin embargo, la mayoría de los problemas reales involucran a un conjunto de agentes que tienen que interactuar para lograr el objetivo final [Oroojlooy and Hajinezhad, 2023]. Las interacciones entre ellos pueden ser cooperativas, competitivas o ambas [Tan, 1993]. Modelizando problemas de aprendizaje por refuerzo en los cuales varios agentes interactúan, se establece una clara limitación en el entorno que puede ver cada uno de los agentes, así en cada tiempo t, cada agente $i \in \mathbb{N}$ observa una representación parcial del estado del ambiente $o \in \mathbb{O}_i$ y realiza una acción $a \in A$ determinada por una función de política

$$\pi_{\mathbb{O}_i} : \mathbb{O}_i \times A_i \longrightarrow [0, 1] \tag{4.6}$$

donde $\mathbb{O}_i$ es el conjunto de todas las observaciones parciales y A_i es el conjunto de todas las acciones. Como se puede ver, se eligen opciones MARL debido a que cada uno de los agentes presentes en el problema está aportando información parcial sobre el entorno y realiza acciones motivadas por esa información parcial que recibe. Sin embargo, si todos esos agentes actúan de forma colaborativa, todos poseen toda la información parcial, obteniendo así una visión global del entorno.

En cada paso de tiempo t, el entorno evoluciona a un nuevo estado $s \in S$ para cada acción tomada en el tiempo t, de acuerdo con la función de transición

$$P : S \times A_1 \times ... \times A_N \longrightarrow [0, 1] \tag{4.7}$$

que depende del estado actual y las acciones de todos los agentes. Finalmente, la recompensa recibida por cada agente es dada por la función de recompensa

$$R_i : S \times A_1 \times ... \times A_N \longrightarrow \mathbb{R} \tag{4.8}$$

los problemas MARL cooperativos usualmente involucran agentes que comparten la recompensa. De esta forma, una de las métricas objetivo más utilizadas es la evaluación de la recompensa del MARL total, y las recompensas medias para cada tiempo t (ver [Zhang and Zavlanos, 2024]). Aunque esta es una métrica muy utilizada, en la realidad no sirve para comparar los algoritmos MARL que tengan tareas distintas asignadas, es decir, supongamos que tenemos dos algoritmos basados en machine learning, uno para clasificar manzanas rojas y otras verdes, y el otro para clasificar coches rojos y verdes. Aunque las métricas de los algoritmos de machine learning son

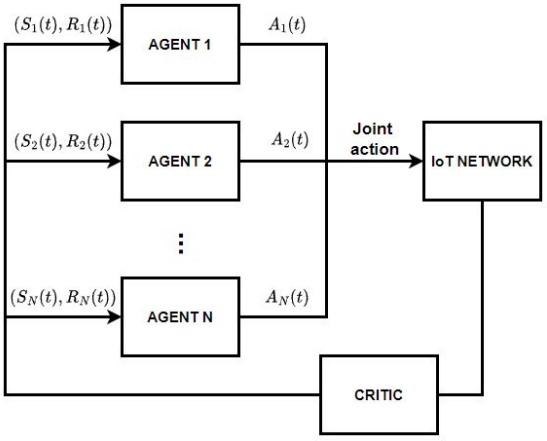


FIGURA. 4.1: Arquitectura MARL para el descubrimiento de malware en redes IoT.

las mismas (accuracy, precisión, recall, f1-score), las tareas de clasificación son distintas y, por tanto, no se pueden comparar. Por tanto, comparar el algoritmo MARL diseñado en este capítulo, con los algoritmos MARL diseñados en la literatura para distintas tareas es una tarea muy compleja.

El enfoque más común en MARL es utilizar agentes de aprendizaje independientes [Sutton and Barto, 2018]. Desafortunadamente, este enfoque no funciona bien en la práctica. En un sistema multiagente, los agentes aprenden simultáneamente durante el proceso de aprendizaje y el entorno se vuelve no estacionario, lo que invalida la hipótesis de Markov y todos los algoritmos que el gymnasium de OpenAI nos ofrece ya no pueden ofrecer garantías de convergencia Dalmau and Allard [2020]. Como se puede ver, esta es una limitación muy seria ya que, si no hay garantías de convergencia, es muy posible que el algoritmo se quede iterando para siempre o hasta que el ordenador detenga su ejecución. Por otro lado, en entornos cooperativos, otro problema común es la asignación de mérito cuando la recompensa compartida resulta de una acción conjunta que no puede ser asociada con cada agente. Es decir, al tratarse de una recompensa conjunta, como se puede ver en la ecuación (4.8), no se sabe cuáles de todos los agentes están realizando una acción que contribuye a resolver el problema y cuáles están haciendo acciones menos efectivas. Esto dificulta el proceso de aprendizaje de los agentes y el algoritmo puede no converger durante el entrenamiento Mondal et al. [2022]. Para resolver este problema, algunos investigadores utilizan un enfoque actor-crítico como solución preferida [Padhye and Lakshmanan, 2023, Zhu et al., 2024]. Sin embargo, vamos a utilizar una solución híbrida entre el uso de un crítico y la teoría de juegos. La limitación del modelo que proponemos en esta investigación de usar un crítico centralizado es que el número de agentes está limitado ya que la entrada para el crítico son las acciones y observaciones de todos los agentes. Esto puede causar que la complejidad del problema aumente demasiado si hay muchos agentes.

Los agentes del MARL tienen como objetivo identificar todos los nodos infectados en la red IoT. A lo largo de esta sección, hemos descrito cómo funciona un agente. En este problema, se necesita una gran cantidad de agentes para completar el escaneo de la red IoT antes de que el malware infecte toda la red. Las decisiones que los agentes deben tomar sobre qué acciones tomar son un MDP en el que los agentes colaborarán para alcanzar la mejor acción para el sistema multiagente. Por lo tanto, se espera que los agentes maximicen la recompensa esperada. Es decir, los agentes trabajarán colaborativamente para lograr la máxima recompensa esperada en cada paso de tiempo. Esta solución al problema MDP en cada paso de tiempo se hará mediante el equilibrio de Nash [Zhou et al., 2023]. Un equilibrio de Nash es un conjunto de políticas, una para cada agente, tal que cada política individual es la mejor para los otros agentes en su conjunto. Aunque la propiedad de Markov sirve para modelar los ambientes en problemas MDP, las recompensas de los agentes se basan únicamente en las acciones que toman en el paso de tiempo actual. Un MDP para un agente está compuesto por un conjunto finito de acciones, un conjunto finito de estados y la función de transición de estado, como se ha descrito en este capítulo. La Figura 4.1 describe los componentes principales de la arquitectura MARL propuestos en esta sección. En resumen, cada agente del MARL recibe un estado y una recompensa, que lo motiva a realizar una acción para cambiar su estado en el entorno. Una vez que todos los agentes han realizado sus acciones sobre el entorno, es decir, sobre la red IoT. El crítico realiza las observaciones, calcula las nuevas recompensas, y envía a los agentes la información sobre los nuevos estados y las recompensas que han obtenido por realizar las acciones que hicieron en el periodo de tiempo anterior.

4.2.3 Solución al MDP para el MARL con N-step bootstrapping

Para concluir esta sección en la que hemos detallado el diseño del algoritmo para identificar los nodos infectados en la red IoT. Vamos a revisar los fundamentos teóricos que permiten que este algoritmo se entrene. Aunque, como se ha comentado en esta sección, la técnica de N-step bootstrapping es una técnica para resolver MDP con un solo agente, se puede generalizar sin pérdida de precision [Zhang and Wang, 2024]. Para este tipo de problemas, existen dos métodos principales para estimar el valor de q_*: el método de estimación Monte Carlo (MC) y el método de estimación de diferencia temporal de un paso (TD). En este trabajo, comenzaremos desde los métodos TD de n pasos que unifican y generalizan los métodos MC y TD para diseñar nuestro algoritmo de aprendizaje por refuerzo. Los métodos TD de n pasos permiten resolver una gran variedad de métodos donde los entornos son finitos pero su tamaño puede ser muy grande. La ventaja de este tipo de método es que se ajusta al problema que se está

resolviendo, es decir, si el entorno es finito y muy grande, utilizará métodos MC, por el contrario, si el entorno tiene pocos elementos, utilizará métodos TD [Parisi et al., 2019, Yuan et al., 2019]. Por lo tanto, estamos diseñando un algoritmo que detectará nodos infectados independientemente del tamaño de la red IoT. A diferencia de lo que ocurría en la problemática resultante en el capítulo anterior, en la que desconocíamos la topología de la red, y por tanto, se utilizaban técnicas MC para estimar las soluciones óptimas, en este caso, como se conoce previamente la red, se utilizará un diseño que permite al algoritmo adaptarse al tamaño de la red conocido previamente.

Los métodos MC actualizan $v_\pi(S_t)$ de manera completa como se muestra en la siguiente ecuación:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-t-1} R_T \tag{4.9}$$

donde T es el último paso de tiempo del episodio. Podemos adaptar esta definición al algoritmo basado en n pasos de la siguiente manera:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) \tag{4.10}$$

para todo n, t tal que $n \geq 1$ y $0 \leq t < T-n$. Suponemos que todos los retornos de n pasos se pueden considerar aproximaciones al retorno completo, truncados después de n pasos, y corregidos los términos faltantes con $V_{t+n-1}(S_{t+n})$. Nótese que los retornos de n pasos para $n > 1$ involucran recompensas futuras y estados que aún no se han descubierto en el paso de tiempo moviendo t a $t+1$. Corregimos este problema utilizando el algoritmo de aprendizaje de valor de estado natural llamado TD de n pasos, definido de la siguiente manera:

$$V_{t+n-}(S_t) = V_{t+n-1}(S_t) + \alpha[B_{t:t+n} - V_{t+n-1}(S_t)] \tag{4.11}$$

Sean π_1, π_2 dos políticas. π_1 es una política codiciosa para la estimación actual de la función de valor de acción y π_2 es una política más exploratoria, elegimos una política $\epsilon-$codiciosa. Para diseñar nuestro algoritmo, seleccionamos la versión off-policy del aprendizaje por refuerzo. Por lo tanto, la actualización para el paso de tiempo t en los algoritmos TD de n pasos debe ser ponderada. Sea $\rho_{t:t+n-1}$ la proporción de muestreos de importancia. Definimos

$$\rho_{t:t+n-1} = \prod_{k=t}^{min(h,T-1)} \frac{\pi_1(A_k|S_k)}{\pi_2(A_k|S_k)} \tag{4.12}$$

como la probabilidad bajo las dos políticas de seleccionar las n acciones del conjunto A_t a A_{t+n-1}. Finalmente, sea $\bar{V}$ el valor aproximado esperado del estado s, bajo la política objetivo:

$$\bar{V} = \sum_a \pi(a|s) Q_t(s,a) \tag{4.13}$$

para todo $s \in S$.

Para lograr una variación continua, definimos $\sigma_t \in [0,1]$ como el grado de muestreo de bootstrap en el paso t, con $\sigma = 1$ que denota un muestreo completo y $\sigma = 0$, no muestreo. Esta nueva variable aleatoria σ_T es el último término de nuestro algoritmo. Llamamos a nuestro algoritmo n-step $Q(\sigma)$. A continuación, se expone el pseudocódigo del algoritmo para actualizar la función acción-valor y así poder encontrar el q_* óptimo. Este pseudocódigo ha sido modificado del original que se puede encontrar en [Sutton and Barto, 2018].

Algorithm 1: Actualización de la función de valor de acción de bootstrapping n-step $Q(\sigma)$ pseudocódigo

Entradas;
  Sea τ el tiempo cuya estimación se está actualizando. $\tau \leftarrow t - n + 1$;
  Para todo $a \in A$, $s \in S$, inicialice $Q(s,a)$ de forma aleatoria;
  Para todo $a \in A$, $s \in S$, elija y almacene una acción $A_{t+1} \leftarrow \pi_2(\cdot|S_{t+1})$;
  Inicialice π_1 como voraz con respecto a Q;
  $\rho_{t+1} \leftarrow \frac{\pi_1(A_{t+1}|S_{t+1})}{\pi_2(A_{t+1}|S_{t+1})}$;
Parámetros;
  Sea $\alpha \in [0,1]$ el tamaño del paso;
Loop para cada episodio;
  T $\leftarrow \infty$;
for *k=min(t+1,T) iterando en* τ **do**
  if *k=T* **then**
    G $\leftarrow R_T$;
  else
    $\bar{V} \leftarrow \sum_a \pi_1(a|S_k) Q(S_k|a)$;
    $G \leftarrow R_k + \gamma(\sigma_k \rho_k + (1-\sigma_k)\pi_1(A_k|S_k))(G - Q(S_k, A_k)) + \gamma \bar{V})$;
  $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$;
  Loop termina con $\tau = T - 1$;

4.3 Metodología para la identificación del malware con datos reales

En esta segunda parte del capítulo vamos a presentar una solución a la última parte del objetivo general de esta tesis, es decir, la parte de identificar el malware. Durante el capítulo anterior y el presente capítulo hemos discutido y propuesto soluciones al problema inicial que se había destacado en el objetivo general. En primer lugar, hemos diseñado un algoritmo basado en aprendizaje por refuerzo, el cual es capaz de descubrir la topología de una red IoT con información parcial o nula. Teniendo en cuenta que, de entre las limitaciones que hemos indicado que tiene este algoritmo, quizá la más crítica es que necesitamos tener un punto de acceso a dicha red, lo que hemos considerado como el nodo inicial a partir del cual vamos a realizar la exploración. En este capítulo continuamos abordando el objetivo general, y lo dividimos en dos subobjetivos: 1) identificar los nodos IoT que están infectados y 2) identificar el malware que está infectado dichos nodos (en el caso de que sea conocido). De la propia naturaleza de este segundo subobjetivo destacamos que los cibercriminales diseñan nuevos malwares constantemente. Por tanto, la primera limitación que vamos a establecer en diseño de esta nueva metodología para identificar el malware que se está propagando por la red IoT, es que el malware sea conocido. Actualmente, es fácil encontrar muchos artículos de revisión del estado del arte sobre la clasificación de malware [Abusitta et al., 2021, Yan et al., 2022].

En ellos se pueden ver diferentes formas de tratar de clasificar a los malwares más típicos utilizando diferentes técnicas como el machine learning y el deep learning entre otras. Sin embargo, todos coinciden generalmente en el mismo diseño. En primer lugar, se realiza un experimento, ya sea sintético o real, para recopilar datos y crear un dataset. Este punto se puede saltar si ya se cuenta con un dataset. Este dataset contendrá toda la información sobre el ataque a la red IoT o a uno de sus nodos y deberá estar etiquetado correctamente con el tipo de malware que está atacando la red IoT o si son datos benignos. Nosotros, en esta investigación, hemos decidido tratar con el más común de todos ellos, el ataque de denegación de servicio (DDoS por sus siglas en inglés) [Vishwakarma and Jain, 2020]. El paso siguiente de las metodologías que se encuentran en la literatura consiste principalmente en la limpieza de los datos. En esta fase de la metodología, se aplican técnicas de limpieza de datos faltantes, se normalizan los datos, se eliminan características según su coliniaridad, es decir, si están relacionados o correlacionados con las otras características del dataset dando como resultado un dataset preparado para aplicar técnicas de machine learning o deep learning. Por último, para concluir el diseño de la metodología, nosotros hemos optado por utilizar

técnicas de aprendizaje supervisado de machine learning Nasteski [2017]. Estas tecnicas utilizan una gran cantidad de datos para analizarlos y buscar patrones con los que poder clasificar o predecir los nuevos datos que se les pasen. En nuestro caso, hemos hecho una selección amplia de diferentes métodos tanto lineales, no lineales, probabilísticos, y de *ensemble*. Generalmente, los algoritmos de *ensemble* tienen el mejor desempeño en tareas complicadas como la que estamos tratando, pero nosotros queríamos satisfacer nuestra curiosidad científica y comprobar cómo se comportaban los algoritmos ante esta tarea de clasificación.

Una de las limitaciones más comunes cuando se está tratando con este tipo de datasets en el que hay muchas muestras del caso negativo, es decir, no hay malware atacando la red IoT, y pocas muestras del caso positivo, es decir, si hay malware atacando la red, es el desbalanceo de las clases. Este tipo de datasets es muy complicado de manejar debido a todos los problemas que tiene asociados, entre ellos el más común es que los algoritmos aprendan a clasificar la clase mayoritaria teniendo bastantes errores en la clasificación de la clase minoritaria [Kotsiantis et al., 2006]. Sin embargo, en la literatura se pueden encontrar muchas recomendaciones para tratar con este tipo de datasets, entre ellas cabe destacar el oversampling de la clase minoritaria y el downsampling de la clase mayoritaria [Kotsiantis et al., 2006]. Aunque estas técnicas suelen dar buen resultado, la verdad es que se están introduciendo muchos datos sintéticos y pueden alterar los resultados en entornos reales. En nuestro caso, se ha diseñado una investigación prospectiva con pruebas en laboratorio sobre datos reales. Así que se ha decidido no usar técnicas de muestreo para alterar el desbalanceo del dataset, ya que se pretende en trabajo futuro aplicarlo en situaciones reales.

Para concluir, hay que destacar que se ha elegido el malware DDoS debido a que generar datos es muy sencillo desde el punto de vista técnico y además existen varios dataset con datos reales realizados por grandes empresas de antivirus para que los investigadores puedan aportar conocimiento al estado del arte [Garcia et al., 2020].

El resto de la sección está organizada según la metodología discutida durante esta misma sección. En primer lugar, se han hecho varias simulaciones para crear los datos siguiendo el diseño del experimento realizado por Avast para crear el dataset [Garcia et al., 2020] en entornos virtuales. Una vez que el dataset está creado, se exploraban los datos detallando la información del dataset y se ha terminado con la limpieza de datos. Para concluir, se describen y listan los modelos de machine learning que se van a utilizar para esta tarea de identificación del malware en las redes IoT.

4.3.1 Analisis y adquisición de datos reales o simulados

Uno de los principales retos de la investigación en redes basadas en IoT es la falta de un conjunto de datos exhaustivo basado en la red que refleje los escenarios de tráfico de red actuales, una amplia variedad de intrusiones de baja huella e información estructurada en profundidad sobre el tráfico de red [Shafiq et al., 2020]. Siguiendo el experimento detallado por Garcia et al. [2020], se diseñó una pequeña red virtual (ver [Ahmad, 2019]) en la que se puso un nodo de una red IoT y siguiendo las indicaciones presentadas por varios autores (ver [Maiwada et al., 2024, Yaacoub et al., 2023] y usando el software Kali Linux (https://www.kali.org/) se realizaron varios ataques DDoS [Veloz et al., 2017]. El tráfico de la red generado se registró utilizando la herramienta Wireshark (https://www.wireshark.org/) y siguiendo las indicaciones de Garcia et al. [2020] se construyó un dataset similar al que ellos diseñaron. Aunque en nuestro caso, virtualizamos una red y un nodo IoT, Garcia et al. [2020] utilizó diferentes tipos de dispositivos que incluyen, entre otros dispositivos para el hogar inteligente, cámaras de seguridad, sensores y muchos otros tipos de dispositivos IoT. Dado que los datos que registramos tenían el mismo formato que los datos del dataset de Avast, decidimos unificarlos y etiquetarlos como ataques DDoS o benignos, lo que nos aportó un total de 485.000 registros. El conjunto de datos consta de archivos PCAP que contienen tráfico de red capturado de dispositivos IoT. Cada archivo de captura contiene datos de tráfico de red de un solo dispositivo IoT. Estas colecciones contienen datos de diferentes capas de red, incluidas las capas de transporte, red y aplicación [Azab et al., 2022].

Las etiquetas, ataque DDoS o benigno, tienen como objetivo proporcionar a los investigadores una base auténtica para entrenar y validar sus técnicas de detección, y se derivan del análisis manual de los registros de tráfico de red. Este conjunto de datos se destina a la investigación y desarrollo de aprendizaje automático y otras técnicas para la identificación de malware y anomalías benignas en el IoT. Su naturaleza etiquetada lo hace idóneo para métodos de aprendizaje supervisado. Puede ser utilizado para desarrollar diversos enfoques de detección, como sistemas de detección de intrusiones para dispositivos IoT, sistemas de inteligencia de amenazas y otras aplicaciones de seguridad. Dado que la mayor parte de las capturas de tráfico de red provienen de dispositivos IoT reales, este conjunto de datos resulta particularmente valioso para evaluar la efectividad de estas técnicas en entornos prácticos. Aunque nosotros hemos conseguido aportar cierta información sintética, que enriquece el dataset añadiendo información sobre entornos virtualizados.

4.3.2 Descripción del dataset

Debido a que el dataset que hemos utilizado se ha basado en el diseño del dataset IoT-23 y la mayor parte de la información es de ese dataset, vamos a proceder a describir este dataset. El conjunto de datos IoT-23 proporciona datos sobre una red basada en IoT con 23 escenarios en los que intervienen sistemas operativos Windows y Linux en diversos tipos de ataques relacionados con IoT. Muchos escenarios de red publicados en 2020 y recopilados durante un año están pendientes de análisis. Los datos de IoT-23 se recopilaron durante al menos una hora y hasta un máximo de 112 horas, dependiendo de lo rápido que aumentara el tamaño del archivo pcap. Los tres primeros escenarios incluyen únicamente datos normales de tres dispositivos IoT: Amazon Echo, Philip Hue y Somfy. Los 20 escenarios restantes incluyen datos normales y de ataque junto con diferentes periodos y tipos de ataque, al tiempo que nombran a cada escenario un tipo de ataque específico: Mirai para los escenarios 1 a 5, 14 y 15, Torri para los escenarios 6 y 7, Trojan para el escenario 8, Gagfyt para el escenario 9, Kenjiro para los escenarios 10 y 12, Okiru para el escenario 11, Hakaj para el escenario 13, IRCBot para el escenario 16, Linux Mirai para el escenario 17, Linux Hajime para el escenario 18, Mushstik para el escenario 19 y Hide and Seek para el escenario 20. Los tres dispositivos son reales, por lo que los escenarios no son simulados ni emulados, y el conjunto de datos se genera a partir de un entorno real [Kim et al., 2022]. En concreto, nosotros vamos a utilizar los ataques hechos por la Botnet Mirai. Cabe destacar que el ataque DDoS es muy efectivo en las redes IoT ya que obliga a los sensores a estar siempre encendidos consumiendo batería y, por tanto, reduciendo considerablemente el tiempo de vida de la batería [Gopal et al., 2021].

Las características o *features* del dataset son las siguientes:

- **Tiempo de conexión**: Esta es la duración de tiempo durante la cual una conexión de red particular ha estado activa. El conocimiento de la duración de una conexión puede ser útil para identificar posibles ataques o comportamientos sospechosos. Por ejemplo, un atacante puede intentar extraer datos de la red si una conexión dura mucho más de lo esperado.
- **Bytes de origen y respuesta**: Al monitorear la cantidad de bytes que son enviados y recibidos por un dispositivo en cada paquete, podemos detectar una serie de anomalías diferentes en el tráfico de red. Por ejemplo, enviar repentinamente paquetes con mucha información o recibir muchos paquetes vacíos puede indicar que un dispositivo está siendo sometido a un DDoS.
- **Bytes perdidos**: El número de bytes que se esperaba recibir, pero que no se entregaron con éxito mientras estaba conectado. El número de bytes perdidos

puede indicar posible pérdida de paquetes o manipulación. Esto podría ser un signo de un ataque en curso o un intento de exfiltración de datos.

- **Paquetes de origen y respuesta**: El número total de paquetes (o unidades pequeñas de datos) que son enviados o recibidos desde la fuente hasta el destino durante la conexión. Los posibles ataques de red, como los ataques DDoS, o las anomalías en el tráfico de red que pueden indicar una violación de seguridad pueden ser identificados mediante el análisis del número de paquetes enviados y recibidos.

- **Bytes de IP de origen y Bytes de IP de respuesta**: El número total de bytes del Protocolo de Internet (IP) enviados y recibidos desde el origen hasta el destino mientras está conectado. Los posibles ataques utilizando protocolos basados en IP, como el spoofing de IP o la inyección de paquetes, pueden ser detectados mediante la monitorización del tráfico IP.

4.3.3 Pre-procesado de los datos

El preprocesado de datos es una parte imprescindible para el desarrollo de modelos de machine learning, ya que garantiza que los datos estén en el formato adecuado y sean aptos para poder entrenar los modelos. En este trabajo hemos seguido los pasos recomendados por [Alasadi and Bhaya, 2017]. En primer lugar, hemos explorado el dataset buscando valores faltantes y NaNs (Not a Number por su nombre en inglés), valores atípicos, y cualquier otro tipo de error en la codificación de los datos. Respecto a los NaNs se tomo la decisión de convertirlos en 0s, y en relación con los valores faltantes, se fijó el umbral de que si en la fila faltaban al menos el 80% de los datos, se borraba completamente. Esto es posible debido a que se tiene un gran número de filas, si esto no fuera posible, habría que utilizar otras técnicas detalladas en cómo imputar valores faltantes como la media, interpolar, etc (ver [Alasadi and Bhaya, 2017]). A continuación se ha revisado que no haya multicolinearidad, es decir, columnas con relaciones entre ellas. Para ello, se lleva a cabo un estudio de la correlación estadística entre las diferentes variables en el conjunto de datos para mejorar el rendimiento de los distintos modelos, reducir costos computacionales y ayudar a la interpretación de los resultados, como se muestra en la Figura 4.2.

Las variables correlacionadas pueden proporcionar información redundante o irrelevante, que puede ser eliminada al seleccionar características. Esto puede simplificar el modelo, reducir el sobreajuste y mejorar el rendimiento de generalización. Como podemos ver en la Figura 4.2, no hay variables con una correlación relevante entre ellas, lo que elimina la posibilidad de información irrelevante o redundante.

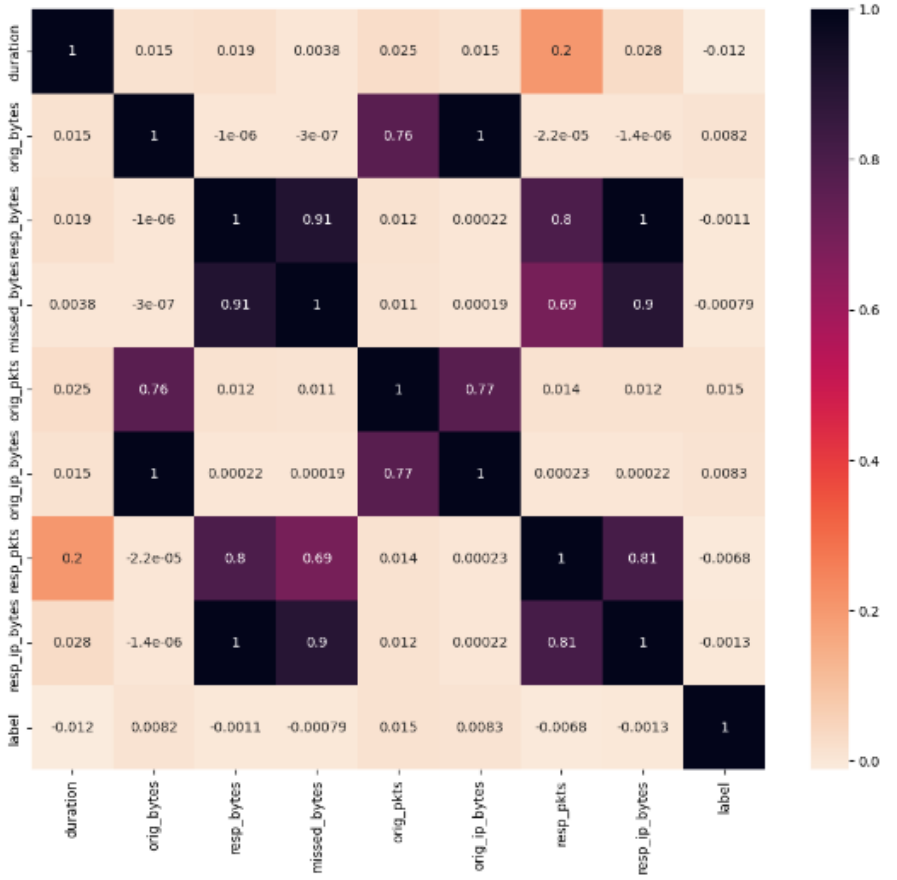


FIGURA. 4.2: Correlación estadística entre las distintas variables del conjunto de datos.

El siguiente paso es la normalización de los datos. Dependiendo del algoritmo de machine learning que se vaya a utilizar, puede ser muy beneficioso normalizar o estandarizar las características o features (en inglés) numéricas para que estén en la misma escala. La normalización ajusta los valores en un rango especificado, el más común es el rango $[0, 1]$, mientras que en la estandarización se centran los datos alrededor del cero con una desviación estándar de uno (ver [Raju et al., 2020]).

En muchos datasets, que contienen mucha información extra que no es necesaria, se recomienda el uso de técnicas de reducción de la dimensionalidad para mejorar el rendimiento de los algoritmos. En este caso, como es un dataset ya preparado para el estudio y el diseño de algoritmos basados en machine learning, hemos apreciado que no es necesario eliminar más características de las que se han eliminado usando el estudio de la multicolinearidad.

A continuación, se hace la división de los datos. En este caso hemos utilizado la división más común, es decir, 70% para el entrenamiento de los algoritmos y un 30% para las pruebas o test. Algunos estudios más recientes, recomiendan utilizar 70% para entrenamiento de los algoritmos y un 20% para las pruebas y 10% para la validación,

pero dado que este dataset cuenta con muchos datos, no hemos considerado necesario utilizar estas técnicas nuevas para mejorar la precisión de los modelos que vamos a desarrollar (ver [Nguyen et al., 2021]).

Para finalizar, y como recomendación de buenas prácticas, se guardó el dataset resultante en formato CSV para su posterior explotación tantas veces como fuera necesario.

4.3.4 Descripción de los algoritmos de machine learning utilizados en esta investigación

Debido a que se puede encontrar en la literatura mucha información sobre los diferentes algoritmos de machine learning, se ha decidido no dedicar un capítulo por si solo a los algoritmos de machine learning, sino que se describirán brevemente los algoritmos que se van a usar, pero si se quiere información mas detallada, se pueden consultar los trabajos listados a continuación [Shetty et al., 2022, Vercio et al., 2020]. A partir de la revisión del estado del arte, poniendo especial atención en los problemas de clasificación binarios de ataques DDoS, se ha hecho una selección de artículos que se puede consultar en la Tabla 4.1. En esta tabla se ha hecho una selección de trabajos, eligiendo uno por cada año de los últimos cinco años (2018-2023), si se quiere tener información más detallada sobre los trabajos presentes en la literatura relacionados con ataques DDoS y su detección con machine learning se puede consultar el trabajo de [Kadri et al., 2023] y de [Singh and Jain, 2024].

Decidimos utilizar seis tipos de modelos de clasificación diferentes con algunas variantes dentro de ellos, como modelos lineales, modelos basados en instancias, modelos SVM, modelos probabilísticos, modelos basados en árboles y modelos de ensemble, en concreto, bagging y boosting (normalmente se ponen sus nombres en inglés), con el fin de comparar y probar cuál de estos algoritmos es más adecuado para el problema de clasificar un paquete de red que se lee de un dispositivo IoT [Tarekegn et al., 2021].

Para empezar, en cuanto a los modelos lineales, se elige la Regresión Logística. Al resolver problemas de clasificación, este tipo de modelo es uno de los más comunes. Funciona estimando los parámetros de una función logística para predecir la probabilidad de pertenecer a una clase particular. Esta función logística se encarga de asignar una probabilidad de estar en las clases de salida a cada característica de entrada. Funciona bien para problemas de clasificación binaria y es simple y fácil de interpretar.

Además, se elige el método de los k-Vecinos Más Cercanos (KNN) como el modelo para los modelos basados en instancias. KNN es un modelo no paramétrico que hace predicciones basadas en los k-vecinos más cercanos en el espacio de características. El

Autores	Algoritmos	Métricas	Resultados	Trabajo futuro	Año
[Khuphiran et al., 2018]	SVM, DFF	Accuracy, Recall, Precision y F1 score	DFF puede clasificar los datos con mayor precisión. SVM es una opción adecuada para un método de clasificación más rápido.	Examinar los dos algoritmos con los datos en tiempo real para desarrollar un método útil que esté disponible con las redes reales	2018
[Kim, 2019]	BNN, LSTM, RNN	TPR, PPV, Accuracy, Recall, Precision y F1 score	métodos de preprocesamiento, optimizadores y arquitecturas de red adecuados	Comparar las tecnologías avanzadas de ML relacionadas con los datos de series temporales	2019
[Gopal et al., 2021]	ANN-MLP	Accuracy, Recall, Precision y F1 score	método de detección dinámico basado en MLP contra el ataque DDoS	investigar una solución más eficaz y ligera para percibir los errores de detección	2021
[Wang et al., 2022]	SVM, NB, KNN, RF, XGB, GBC, Adaboost	Accuracy, Recall, Precision y F1 score	método de detección de DDoS en un testbed real	investigar la reducción de características y su aplicación al tráfico real de una red IoT	2022
[Nguyen and Le, 2023]	SVM, GLM, NB, DA, NN, DT, KNN, RF	Accuracy, Recall, Precision y F1 score	defensa óptima para ataques DDoS en la red SDN-IoT	Investigar técnicas de ML más avanzadas	2023

Tabla. 4.1: Revisión de la literatura de los principales trabajos relacionados con la aplicación de machine learning en la detección de ataques DDoS.

modelo funciona calculando las distancias entre cada uno de los puntos de prueba y cada uno de los puntos de entrenamiento, y luego seleccionando los k puntos más cercanos para ser considerados como los vecinos. Al final, asigna a este punto de prueba la clase que es más común entre sus vecinos. Este modelo funciona bien para problemas multi-clase y binarios, a diferencia del anterior [Lavate and Srivastava, 2023].

Además, se elige el SVC (Support Vector Classifier) como el modelo para los modelos de máquinas de vectores de soporte (SVM). Este modelo funciona estableciendo un límite máximo. Luego encuentra el hiperplano que separa los datos en las diferentes clases dentro de ese límite. Específicamente, el SVC es un tipo de SVM (Máquina de Vectores de Soporte) utilizado para clasificar datos binarios [Abdullah and Abdulazeez, 2021].

Además, Naive Bayes es el modelo de elección para los modelos probabilísticos. Es un modelo que tiene la suposición de que las diferentes variables son condicionalmente independientes de la asignación de etiquetas. El modelo funciona estimando las probabilidades a priori de las etiquetas y de las variables como una función de las etiquetas. Finalmente, utiliza el teorema de Bayes para calcular las probabilidades a posteriori. Se pueden abordar problemas de clasificación binaria y multi-clase con este modelo [Wickramasinghe and Kalutarage, 2021].

Además, se elige el clasificador de Decision Tree como modelo basado en árboles y el Random Forest y el clasificador Extra Trees como ejemplos de modelos de ensemble con técnicas de bagging. Estos algoritmos funcionan construyendo un conjunto de árboles de decisión utilizando subconjuntos aleatorios de las características de entrada y muestras de entrenamiento. Luego, sus predicciones se promedian para producir una salida final. Se pueden abordar problemas de clasificación binaria y multi-clase con este modelo [Çetinkaya and Horasan, 2021].

Por último, se eligen el clasificador AdaBoost y el clasificador Gradient Boosting como los modelos de ensemble con técnicas de boosting. Estos algoritmos funcionan entrenando de forma iterativa clasificadores débiles sobre los errores residuales de los clasificadores anteriores. Luego, sus predicciones se suman a la predicción general. La salida final se obtiene combinando las predicciones de todos los clasificadores débiles, ponderadas por su contribución [Tanha et al., 2020].

4.3.5 Evaluación del rendimiento de los algoritmos de machine learning

Para evaluar adecuadamente cada modelo para la clasificación de un paquete de red de un dispositivo IoT como benigno o parte de un ataque DDoS, se han utilizado y se presentarán en esta sección varias métricas. Antes de detallar las métricas, es necesario definir una serie de conceptos que son la base para el cálculo de estas métricas:

- Verdaderos Positivos (**TP**, por sus siglas en inglés): Después de la clasificación, esto corresponde al número total de paquetes de ataque DDoS correctamente clasificados como tales.

- Verdaderos Negativos (**TN**, por sus siglas en inglés): Después de la clasificación, esto corresponde al número total de paquetes benignos correctamente clasificados como tales.
- Falsos Positivos (**FP**, por sus siglas en inglés): Después de la clasificación, esto corresponde al número total de paquetes benignos mal clasificados como DDoS.
- Falsos Negativos (**FN**, por sus siglas en inglés): Después de la clasificación, esto corresponde al número total de paquetes DDoS mal clasificados como benignos.

La primera de las métricas utilizadas para evaluar los modelos se llama Precisión. La Precisión es la proporción de positivos correctamente identificados y es una medida del rendimiento del modelo. Su fórmula es la siguiente:

$$\text{Precisión} = \frac{TP}{TP + FP}. \tag{4.14}$$

La segunda métrica utilizada para evaluar el modelo se llama Accuracy. La Accuracy es una métrica que evalúa el modelo calculando la proporción de predicciones que son correctas respecto al total de predicciones realizadas. Su fórmula es la siguiente:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.15}$$

La tercera métrica utilizada para evaluar el modelo se llama Recall. Recall es una métrica que evalúa el modelo calculando la proporción de positivos reales que han sido correctamente identificados. Su fórmula es la siguiente:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{4.16}$$

La cuarta métrica utilizada es lo que hemos denominado como la Relación de TN. Esta métrica es exactamente la misma que la métrica de Recall, pero se aplica a los TN. Su fórmula es la siguiente:

$$\text{Relación de TN} = \frac{TN}{TN + FP}. \tag{4.17}$$

La última métrica utilizada se llama puntuación $F1$. Esta es una métrica que se calcula tomando la media armónica de la puntuación de precisión y la puntuación de recall. Debido a que tiene en cuenta tanto los falsos positivos como los falsos negativos, se considera una extensión de la métrica de precisión. Su fórmula es la siguiente:

$$F1 = 2 \times \frac{\text{Relación} \times \text{Recall}}{\text{Precisión} + \text{Recall}}. \tag{4.18}$$

4.4 Metodología para la identificación del malware con datos sintéticos

En esta tercera y última parte del capítulo, se presenta una solución para la identificación del malware en el caso de que no se tenga constancia de su existencia. Es decir, cuando la metodología anterior, basada en casos reales o simulados y utilizando machine learning para clasificarlos, no dé resultados porque el malware sea desconocido o no esté catalogado aún en las bases de datos de entrenamiento. Proponemos una aproximación teórica, es decir, utilizaremos modelos de epidemiología matemática para intentar describir el comportamiento del malware, y en caso de que sea conocido, identificarlo. En caso de que este sea desconocido o no se haya diseñado aún, se dirá que modelos epidemiológicos son los más próximos para ayudar a los investigadores a identificarlo. Como ya se ha discutido a lo largo de esta tesis, el malware es uno de los principales retos actuales en la seguridad de las redes IoT. Por este motivo, muchos investigadores dedican esfuerzos a estudiar el malware, su detección, posibles contramedidas y medidas de mitigación. Sin embargo, el costo económico y logístico de preparar experimentos reales para capturar datos y crear nuevos datasets de estudio hacen que este procedimiento solo esté al alcance de las grandes compañías antivirus y otras organizaciones con presupuestos similares [Canto et al., 2008]. Esto hace que se hayan buscado otras formas, tanto prácticas como teóricas, de conseguir diseñar experimentos para capturar datos sobre la propagación del malware [del Rey, 2015].

En la literatura se pueden encontrar una amplia variedad de modelos epidemiológicos con todas sus variantes, intentando imitar el comportamiento de los virus o del malware, según el campo de estudio de los investigadores que los diseñan [Martín del Rey, 2023b]. Este tipo de datos generados con modelos teóricos se conoce como datos sintéticos. Este tipo de datos es muy útil porque permite estudiar el comportamiento del malware y, si además, añadimos una topología de contacto como las redes IoT, se pueden crear datos sintéticos que permitan realizar un estudio en profundidad de la propagación del malware en redes IoT. Con esto en mente, nos hicimos una pregunta: ¿es posible conocer un malware a partir de los datos sintéticos de un experimento o simulación? Para contestar esta pregunta, necesitamos tener un poco de conocimiento de la epidemiología matemática. En resumen, hay dos tipos de modelos: los globales y los individuales. Los globales son modelos controlados por ecuaciones diferenciales, mientras que los individuales, en general, están controlados por reglas de transferencia [del Rey et al., 2022]. Si nos centramos en los modelos globales, sabemos que conociendo los parámetros de las ecuaciones que definen un modelo de propagación, se pueden replicar los resultados obtenidos. Por tanto, uno puede pensar que haciendo ingeniería inversa, a los resultados

de una simulación se podrían obtener los parámetros del modelo. Esto es cierto, y de hecho, en el campo de las matemáticas e ingenierías tiene un nombre, problema inverso de estimación de parámetros (ver [Tarantola, 2005]).

Entonces, al resolver el problema inverso y estimar los parámetros del modelo de propagación del malware, se puede conocer el malware que se está propagando por una red IoT. En primer lugar, esto solo tiene sentido hacerlo sobre modelos globales compartimentales, ya que los parámetros son iguales para todo el modelo. Por tanto, la primera limitación que vamos a imponer es que el modelo no puede tener topología de contacto. Aunque esta limitación es un punto muy interesante que en trabajos futuros trataremos de eliminar para que la aproximación sea más realista. Por tanto, la metodología que se propone comienza con la generación de datos sintéticos utilizando modelos epidemiológicos compartimentales globales como, por ejemplo, el modelo SIR [Lazebnik, 2023]. Una vez que se tienen los datos sintéticos, se va a resolver el problema inverso para la estimación de parámetros suponiendo que el malware es desconocido. Aquí tenemos que volver a imponer una limitación en nuestra propuesta, aunque dados los datos sintéticos suponemos que el modelo es desconocido. Por otra parte, tenemos que asumir que el modelo epidemiológico que ha generado los datos es conocido y está en una base de datos, ya que de lo contrario, no podríamos identificarlo al resolver el problema inverso. En primer lugar, se propone estimar los parámetros mediante técnicas de Monte Carlo y mediante redes neuronales físicamente informadas (PINN por sus siglas en inglés). Una vez que se conocen los parámetros, solo hay que realizar simulaciones con dichos parámetros sobre los modelos ya conocidos y comparar las curvas de infectados, susceptibles y recuperados (para el caso de un modelo SIR), y usando medidas de error adecuadas, encontrar el modelo epidemiológico que se está propagando por la red IoT.

4.4.1 Generación de datos sintéticos de propagación del malware en redes IoT: Enfoques y metodologías

Debido a la reciente pandemia, se ha prestado mucha atención a los científicos y a los modelos teóricos capaces de reproducir la realidad con un margen de error mínimo. Esto es importante, ya que la relevancia del asesoramiento político dependió de la capacidad de los modelos para capturar los aspectos esenciales del sistema que son relevantes para el problema en cuestión [Barlas, 1996, Lee et al., 2019]. Sin embargo, los modeladores enfrentan un dilema al determinar la importancia de cada aspecto. En [Savage, 1972], se utilizó la metáfora de mundos grandes y pequeños para ilustrar este proceso. El mundo grande representa una entidad compleja con información parcial, confusa y ambigua. Los modelos están diseñados para comprender, influir, gestionar y controlar esta complejidad

intrincada [Bar-Yam, 2019]. Por otro lado, el mundo pequeño se refiere a la realidad lógica autocontenida del modelo, donde se pueden tomar acciones correctivas exhaustivas y probar sus consecuencias bajo condiciones favorables y extremas [Mingers, 2006]. Sin embargo, estas pruebas de consistencia lógica están limitadas al mundo pequeño y no pueden aplicarse directamente al mundo grande. Se utilizan procedimientos de validación para aumentar la confianza en la aplicabilidad del modelo al mundo grande. Sin embargo, navegar entre estos dos mundos y reconocer sus diferencias sigue siendo un desafío fundamental en la modelización de problemas reales. Encontrar el equilibrio adecuado y garantizar la relevancia del modelo para las complejidades del mundo real es esencial para el asesoramiento político efectivo [Andrade and Duggan, 2020, Kaniadakis et al., 2020].

En la identificación de modelos epidemiológicos, el mundo grande es el proceso de generación de datos, que abarca las interacciones ecológicas que contribuyen a la propagación de enfermedades infecciosas. Sin embargo, las observaciones parciales de esta actividad son limitadas. Los informes de incidencia de las entidades de vigilancia pueden proporcionar datos de series temporales para la selección de modelos, que pueden utilizarse para obtener estimaciones de parámetros [Hattaf et al., 2012, Miao et al., 2011]. Sin embargo, generar tasas de incidencia precisas a partir de un modelo no es suficiente para la validez, ya que no garantiza que se hayan considerado todos los factores relevantes o que las estimaciones de parámetros estén cerca de las cantidades reales. Para abordar este problema, se pueden utilizar datos sintéticos para representar el mundo grande y garantizar que ambos mundos estén perfectamente alineados. El modelo utilizado para estimar los parámetros es estructuralmente idéntico al proceso de generación de datos, lo que evita inconsistencias en el proceso de calibración. El proceso de generación de datos se basa en investigaciones anteriores en el campo, y el flujo de trabajo desde la producción de incidencias sintéticas específicas por edad hasta la estimación de parámetros se detalla en [Figueira and Vaz, 2022].

4.4.1.1 Modelos epidemiológicos utilizados para la generación de datos sintéticos en este estudio

Los modelos matemáticos pueden simular la propagación del malware basado en modelos desarrollados para estudiar enfermedades infecciosas. Estos modelos son compartimentales, dividiendo la población en diferentes tipos de comportamiento basados en las características de la enfermedad. En el caso de la propagación del malware, estas categorías incluyen susceptibles, expuestos, infectados, en cuarentena y recuperados. Estos modelos pueden ayudar a entender la dinámica de la infección y su impacto en las redes Brauer et al. [2008], Diekmann and Heesterbeek [2000].

Tales modelos incluyen el modelo SIS, SIR, SIRS, SEIR, y variantes SI y SIRS, por mencionar solo algunas del Rey [2015], Kwok et al. [2019]. Si se desea más detalle sobre estos modelos, se puede consultar [Hoang and Ehrhardt, 2024, Kalachev et al., 2023, Martcheva, 2015].

El **modelo SIR** es un modelo matemático utilizado en epidemiología para describir la propagación de enfermedades infecciosas en una población. Consta de tres compartimentos principales: Susceptibles (S), Infectados (I) y Recuperados (R). Los hiperparámetros clave son la tasa de transmisión (β), que representa la tasa a la que los individuos susceptibles se infectan al entrar en contacto con individuos infectados, y la tasa de recuperación (γ), que representa la tasa a la que los individuos infectados se recuperan y pasan al compartimento de recuperados. Estos hiperparámetros son fundamentales para entender cómo evoluciona una epidemia, ya que determinan la dinámica de la enfermedad. Variando β y γ, se pueden analizar diferentes escenarios epidemiológicos y evaluar estrategias de control de enfermedades. Aquí, S_0 e I_0 son los valores iniciales para S e I en el tiempo 0, respectivamente.

$$\frac{dS}{dt} = -\beta S(t)I(t) \tag{4.19}$$

$$\frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) \tag{4.20}$$

$$\frac{dR}{dt} = \gamma I(t) \tag{4.21}$$

$$S(0) = S_0, I(0) = I_0, R(0) = 1 - S_0 - I_0 \tag{4.22}$$

El **modelo SIRS** es un modelo matemático utilizado en epidemiología para describir la dinámica de enfermedades infecciosas en una población. Tiene tres secciones principales: "Susceptibles" (S), individuos que son susceptibles a la infección; "Infectados" (I), individuos infectados que son capaces de transmitir la enfermedad; y "Recuperados" (R), individuos que se han recuperado de la infección y pueden volver a ser susceptibles con el tiempo. El sistema SIRS se caracteriza por varios hiperparámetros clave: la tasa de transmisión β, que representa la tasa de infección; la tasa de recuperación γ, que representa la velocidad de recuperación; y la tasa de pérdida de inmunidad δ, que modela la pérdida gradual de inmunidad con el tiempo. Estos hiperparámetros determinan la dinámica del sistema y son críticos para predecir la propagación de la enfermedad y evaluar estrategias de control. S_0, I_0 son los valores iniciales para S e I en el tiempo 0,

es decir, $R(0), I(0)$.

$$\frac{dS}{dt} = -\beta S(t)I(t) + \delta R(t) \tag{4.23}$$

$$\frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) \tag{4.24}$$

$$\frac{dR}{dt} = \gamma I(t) - \delta R(t) \tag{4.25}$$

$$S(0) = S_0, I(0) = I_0, R(0) = 1 - S_0 - I_0 \tag{4.26}$$

4.4.2 Resolviendo el problema inverso: estimación de parámetros

Si bien la epidemiología matemática puede modelar la propagación de malware en una red IoT al requerir conocimiento de la dinámica de propagación, en realidad, cuando se detecta malware, la información más fácilmente disponible es el número de infecciones en la red. Esto significa que, para entender su dinámica de propagación y mitigar su propagación, la primera tarea cuando se descubre malware en una red IoT es identificarlo. En la literatura existen varias técnicas con este propósito, principalmente basadas en técnicas estadísticas y optimización Dangerfield and Duggan [2020], Driggers et al. [2019], Kanaan and Farrington [2005], Reis and Yang [2011], Reis et al. [2021]. En los últimos años, ha surgido una nueva estrategia para estimar los parámetros de sistemas dinámicos utilizando aprendizaje profundo. Esta técnica consiste en utilizar redes neuronales, con la función de pérdida modificada, para tener en cuenta las ecuaciones que definen los sistemas dinámicos. Estas redes neuronales se llaman PINNs, y se describen por primera vez en Raissi et al. [2019]. Los PINNs se han utilizado ampliamente para estimar los parámetros de modelos epidemiológicos matemáticos utilizados para modelar el COVID-19 y realizar estimaciones de parámetros Berkhahn and Ehrhardt [2022], Grimm et al. [2022], así como en otros campos Jiang et al. [2022], Tartakovsky et al. [2020], Zhao et al. [2022]. Después de revisar el estado del arte, proponemos una comparación entre los métodos basados en Monte Carlo y los PINNs para estimar los parámetros de modelos matemáticos con el fin de identificar con precisión la dinámica de propagación de malware en redes. Esta aproximación se eligió porque, entre las técnicas de estimación de parámetros estadísticos, la inferencia basada en Monte Carlo ha demostrado ser la más eficiente Luengo et al. [2020], Marino et al. [2017]; por otro lado, entre las técnicas de estimación de parámetros basadas en inteligencia artificial, las redes neuronales, en concreto las PINNs, han demostrado ser las más eficientes Gadewadikar and Marshall [2024]. Cuando se han estimado los parámetros de todos los modelos conocidos, se evalúan la forma de la curva de infectados producida por el modelo ideal y la curva de infectados medida en la red con un error cuadrático medio (MSE). Nótese que los parámetros deben estar entre 0 y 1; de lo contrario, el modelo debe ser rechazado.

Suponiendo que $\delta > 0$ es la tolerancia, se puede suponer que si el MSE de ambas curvas es menor que δ, hemos modelado con éxito la propagación de malware a través de la red. Si ninguno de los modelos epidemiológicos coincide con el que se está propagando en la red, se puede suponer que es un modelo desconocido, en cuyo caso la tarea se convierte en modelarlo matemáticamente y estudiar sus puntos de estabilidad y equilibrio.

4.4.2.1 Estimación de parámetros con el método Monte Carlo

Diseñamos un algoritmo de Monte Carlo para estimar los parámetros de los modelos descritos en la Sección 4.4.1. Para aplicar la teoría de inferencia estadística, asumimos que los parámetros son variables aleatorias al diseñar algoritmos basados en técnicas de Monte Carlo Robert et al. [2010]. El objetivo es estimar la distribución posterior de los parámetros, referida como $\pi(\theta|y)$, o la distribución objetiva en el contexto del análisis bayesiano. El propósito es localizar la distribución objetiva dentro del espacio de parámetros, es decir, ubicar las regiones de la masa de probabilidad que describen las observaciones y, lo que puede describirse como el cálculo de la esperanza de g:

$$\mathbb{E}_\pi(g) = \int g(\theta)\pi(\theta|y)d\theta. \tag{4.27}$$

En raros casos, es posible resolver o calcular el valor esperado de una función analíticamente. Sin embargo, cuando esto no es posible, se pueden utilizar métodos de simulación. Al usar la simulación, existe una solución general. Esto implica utilizar técnicas de Monte Carlo de cadenas de Markov (MCMC) para estimar las cantidades de interés. Mediante simulaciones de cadenas de Markov, los valores deseados pueden calcularse de manera efectiva simulando, ya sea a partir de la distribución real o algunas distribuciones sustitutas adecuadas Durmus et al. [2022], Jin et al. [2019]. Para mejorar la precisión de estimación de los algoritmos desarrollados, se utilizan dos funciones de pérdida: el error cuadrático medio (consulte el Algoritmo 2), y otra basada en el error cuadrático logarítmico (consulte el Algoritmo 3). Ambos algoritmos tienen el mismo diseño y se basan en el conocimiento de modelos de propagación de malware para estimar los parámetros desconocidos utilizando inferencia estadística con MCMC. De esta manera, se genera una muestra de hasta 50,000 casos y se compara la solución del algoritmo con la curva de infección obtenida de la red. A continuación se presenta el pseudocódigo de los algoritmos de estimación de parámetros diseñados con técnicas Monte Carlo.

Algorithm 2: Estimación de parámetros de Monte Carlo con función de pérdida MSE (MC-MSE)

Entrada: modelo, t, datos, n_iter=100, limites=None

Salida : mejores_params

```
mejor_perdida ← ∞;
mejores_params ← None;
for _ ← 1 to n_iter do
    params ← valores de parámetros aleatorios dentro de los límites especificados;
    params ← [redondear(num, 3) for num in params];
    params_ajustados, _ ← ajuste_de_curva(modelo, t, datos, p0=params,
     maxfev=50000);
    perdida ← media((datos − modelo(t, *params_ajustados))^2);
    if perdida < mejor_perdida then
        mejor_perdida ← perdida;
        mejores_params ← params_ajustados;
    end
end
return mejores_params;
```

Algorithm 3: Estimación de parámetros de Monte Carlo con función de pérdida logarítmica cuadrada (MC-LS)

Entrada: modelo, t, datos, n_iter=100, limites=None

Salida : mejores_params

```
mejor_perdida ← ∞;
mejores_params ← None;
for _ ← 1 to n_iter do
    params ← valores de parámetros aleatorios dentro de los límites especificados;
    params ← redondear(params, 3);
    params_ajustados, _ ← ajuste_de_curva(modelo, t, datos, p0=params,
     maxfev=50000);
    perdida ← log_perdida_cuadrada(datos, modelo(t, params_ajustados));
    if perdida < mejor_perdida then
        mejor_perdida ← perdida;
        mejores_params ← params_ajustados;
    end
end
return mejores_params;
```

4.4.2.2 Estimación de parametros con PINNs

En esta sección vamos a presentar los fundamentos teoricos de las PINN. Para realizar esta sección nos hemos basado en el trabajo realizado por Raissi *et al.* en las secciones 2 y 4 de [Raissi et al., 2019]. Su trabajo utiliza redes neuronales profundas como aproximadores de funciones universales [Hornik et al., 1989] para abordar problemas no lineales sin suposiciones previas o linealización. Se utilizan técnicas de diferenciación automática para diferenciar redes neuronales con coordenadas de entrada y parámetros del modelo [Baydin et al., 2018], lo que resulta en redes neuronales informadas por la física. Este enfoque aborda varios problemas computacionales e introduce tecnología transformadora para máquinas de aprendizaje informadas por la física y eficientes en datos, solucionadores numéricos y enfoques basados en datos para inversión de modelos e identificación de sistemas. En este trabajo, consideramos ecuaciones diferenciales parciales parametrizadas y no lineales de la forma general

$$u_t + N[u;\lambda] = 0, \quad x \in \Omega, \quad t \in [0,T], \tag{4.28}$$

donde $u(t,x)$ denota la solución latente (oculta), $N[u;\lambda]$ es un operador no lineal parametrizado por λ, y Ω es un subconjunto de $\mathbb{R}^D$. Para ilustrar cómo funcionan los PINN, examinamos el caso del modelo SIR, es decir, detallamos el problema de estimación de parámetros basado en datos sinteticos producidos por el modelo SIR.

Sea $t \in \mathbb{R}_+$ la entrada del PINN, y sea $f(t;\theta) \in \mathbb{R}_+^{m+1}$ la salida del PINN, donde m es el número de capas ocultas. Basado en el modelo de Kermack y McKendrick [Kermack and McKendrick, 1927], el modelo PINN tiene el modelo SIR básico de tres compartimentos con sus parámetros. Entonces, la salida del PINN es

$$f(t;\theta) = \Big[f_1(t;\theta) \; f_2(t;\theta) \Big] \tag{4.29}$$

donde $f_1(t;\theta), f_2(t;\theta)$ aproximan $S(t), I(t)$, respectivamente. Nótese que $R(t)$ está completamente determinado por los otros, ya que $R(t) = N - S(t) - I(t)$. Entonces, podemos reducir la complejidad computacional reduciendo el sistema $[S(t), I(t)]^T$ mostrado en [Grimm et al., 2022]. Suponiendo que no hay datos disponibles para los compartimentos S y R y que ${u_k}_{k=0}^K$ es una serie de tiempo discreta de observaciones en el compartimento I en el tiempo t_k, la pérdida de datos del ECM se define como

$$MSE_{datos} = \frac{1}{K+1} \sum_{k=0}^{K} (u_k - f_2(t_k;\theta))^2 \tag{4.30}$$

donde MSE es la función de pérdida.

Entonces, el problema inverso se puede describir de la siguiente manera. Para un conjunto de datos incompleto, el PINN tiene como objetivo aprender una asignación del tiempo t a cada una de las variables de estado en el modelo existente. Por lo tanto, usando el conjunto de datos incompleto, podemos extrapolar las series de tiempo desconocidas de los compartimentos S y R y aprender la dinámica de transmisión representada por los valores de los parámetros SIR β y γ. El PINN debe acceder a información del modelo preexistente durante el entrenamiento (es decir, el modelo SIR). Entonces, el subsistema se puede escribir como

$$G\left(y, \frac{dy}{dt}; \lambda\right) = \frac{dy}{dt} + N[y] = 0, \tag{4.31}$$

donde $N[\cdot]$ es generalmente un operador diferencial (aunque en el caso de ecuaciones diferenciales ordinarias es posible que $N[\cdot]$ represente una función no lineal de la variable y) y

$$y(t) := \left[S(t)\ I(t)\right], \quad \frac{dy}{dt} = \left[\frac{dS(t)}{dt}\ \frac{dI(t)}{dt}\right], \quad N[y] = \left[\frac{\beta}{N}SI\ -\frac{\beta}{N}SI + \gamma I\right]. \tag{4.32}$$

Si $N[y; \lambda]$ depende de $\lambda = (\beta, \gamma)^T \in \mathbb{R}^2$ con λ que no se conocen a priori, entonces

$$G(y, y_t; \lambda) = y_t + N[y; \lambda], \quad t \in [0, T]. \tag{4.33}$$

Por lo tanto, para entrenar eficazmente el PINN necesitamos minimizar objetivos de la siguiente forma:

$$min_{\theta,\lambda}(MSE_{datos}(\theta) + MSE_G(\theta, \lambda)), \tag{4.34}$$

permitiendo que el PINN aprenda los parámetros del modelo a partir de los datos Grimm et al. [2022], Raissi et al. [2019].

Nótese que si el sistema tiene condiciones iniciales, entonces la función a minimizar es

$$min_{\theta,\lambda}(MSE_{datos}(\theta) + MSE_G(\theta, \lambda) + MSE_{CI}(\theta)). \tag{4.35}$$

En la modelización del problema, hemos restringido los parámetros para que sean independientes del tiempo. Si fueran dependientes del tiempo, sería necesario crear una ventana deslizante con una amplitud de $\alpha\Delta t$ con $\alpha \in \mathbb{R}$ para pasar como entrada a los pequeños marcos PINN de la siguiente manera: $(t, t + \alpha\Delta t)$. Para los modelos restantes, es suficiente sustituir las ecuaciones SIR por las ecuaciones del modelo a estudiar. Por tanto, hemos introducido la limitación de que los parametros de los modelos epidemiologicos deben ser independientes del tiempo en nuestro estudio.

Capítulo 5

Resultados

Capítulo 5

Resultados

5.1 Introducción

En este capítulo se describen los casos de estudio (reales o simulaciones) cuyo objetivo ha sido evaluar el rendimiento de los algoritmos y metodologías diseñados en los capítulos 3 y 4, así como los resultados obtenidos a partir de cada uno de ellos. Cada caso de estudio está diseñado para probar cada uno de los aspectos y funcionalidades de las metodologías y algoritmos. Estas técnicas y tecnologías tienen como principal objetivo mejorar, detectar e identificar el malware que se está propagando por una red IoT.

El objetivo de estos casos de estudio prácticos es evaluar cada una de las propuestas de la investigación realizada. En primer lugar, se han diseñado para comprobar la eficacia de los algoritmos y metodologías propuestas. Posteriormente, usando los casos de estudio diseñados, se optimizaban las propuestas realizando la iteración diseño-prueba-retroalimentación. Todos estos casos de estudio se han diseñado siguiendo las propuestas del capítulo 3 y del capítulo 4. La estructura de cada uno de los casos de estudio es la siguiente: primero se hace una introducción al caso de estudio y una revisión de la literatura motivando el diseño del caso de estudio. A continuación, se describe el experimento y finalmente se muestran los resultados. A continuación, se presenta un breve resumen de los casos de estudio presentados en este cápitulo:

El primer caso de estudio se ha diseñado para probar el algoritmo basado en aprendizaje por refuerzo para descubrir la topología de una red IoT parcial o totalmente desconocida.

El segundo caso de estudio nos propone una red IoT ya conocida en la que se está propagando un malware. En este caso, se quiere validar el algoritmo MARL diseñado para identificar los nodos de la red IoT que están infectados por malware.

El tercer caso de estudio está diseñado para evaluar el rendimiento de la nueva metodología propuesta para la identificación de los ataques DDoS en las redes IoT.

El cuarto caso de estudio se ha diseñado para comprobar la eficacia de la metodología propuesta para la identificación del malware que se está propagando en una red IoT, mediante la resolución del problema inverso para el cálculo de sus parámetros.

El resto del capítulo está organizado como sigue: en la sección 5.2 se propone el caso de estudio I para validar el algoritmo para descubrir la topología de las redes IoT. El caso de estudio II se puede encontrar en la sección 5.3 en el que se quiere evaluar el rendimiento del algoritmo MARL para identificar los nodos infectados en una red IoT. En la sección 5.4 está el caso de estudio III para validar la metodología que descubre o identifica el tipo de malware que está atacando la red IoT. El caso de estudio IV está en la sección 5.5 y está diseñado para evaluar el rendimiento de la metodología para identificar el tipo de malware que está atacando una red IoT a partir de datos sintéticos.

5.2 Caso de estudio I: Descubrir la topología de una red IoT con un punto de entrada conocido

5.2.1 Introducción

En la última década (2014-2024), las redes informáticas, en particular las redes IoT, han tenido una implementación global en todas las áreas, lo que provoca su creciente complejidad [Shafiq et al., 2022]. Por ello, generar una topología de red se está convirtiendo en una tarea cada vez más difícil pero sumamente útil para simulaciones precisas. Descubrir y monitorear la red es una tarea esencial [Jihong et al., 2021]. Además, dada la naturaleza dinámica y de gran escala de las redes IoT actuales, el proceso de descubrimiento debe ser completamente automático y proporcionar resultados correctos en el menor tiempo posible. De hecho, las herramientas y métodos existentes para descubrir diferentes topologías despiertan un interés creciente entre los proveedores de aplicaciones y los administradores de redes, ya que son extremadamente útiles para planificar y gestionar cualquier red, sin importar su tamaño [Docquier et al., 2023]. Conocer la topología de la red es fundamental para comprender cómo se comportan los distintos dispositivos que comparten información dentro de la red. La importancia de contar con un método automático, eficiente y confiable para descubrir la topología de la red radica en que permite detectar errores y resolver problemas de manera más rápida, manteniendo así nuestra red limpia y protegida contra posibles amenazas [Rana et al., 2021].

Actualmente, existen varias investigaciones en la literatura sobre el descubrimiento de redes WAN que ignoran la necesidad de conocer información topológica a nivel de LAN, aunque algunos trabajos han demostrado la relevancia de esta información, aunque asumen un conocimiento completo de los routers entre sí [Liu et al., 2021]. Con el auge de tecnologías como IoT, hay numerosos estudios sobre cómo descubrir la topología en diferentes niveles, es decir, a nivel WAN, LAN y físico. Estos estudios proponen un método basado en los diferentes protocolos de comunicación existentes en estos tres niveles, como por ejemplo [Sikimić et al., 2020]. Por otro lado, en lugar de utilizar múltiples protocolos, existen numerosos estudios basados en el uso de un solo protocolo en un único nivel de red, como el Protocolo Simple de Administración de Redes (SNMP) [Abijaude et al., 2021] o el Protocolo de Descubrimiento de Capa de Enlace (LLDP) [Mohammadi et al., 2022]. Si nos centramos en el nivel de red, otro de los protocolos más utilizados para el descubrimiento de topologías de red es el Protocolo de Mensajes de Control de Internet (ICMP) [Chaudhary et al., 2022], que es utilizado por los diferentes dispositivos conectados a la red para enviar mensajes de error e información

operativa que indica éxito o fallo al comunicarse con otra dirección IP. Los experimentos más recientes concluyen que el algoritmo de descubrimiento de topología con fusión de múltiples protocolos ofrece mejores resultados y adaptabilidad a los cambios de topología [Shukla, 2021].

Dado que diseñar soluciones basadas en entornos reales puede llegar a ser muy costoso económica y logísticamente, además de computacionalmente, usualmente se simulan las redes IoT usando grafos, en concreto, usando la librería NetworkX de Python. Por tanto, el proceso de descubrir una red IoT desconocida se puede simular mediante la exploración de un grafo desconocido partiendo desde algunas posiciones iniciales fijadas, que podemos llamar puntos de entrada. En cuanto a la exploración del grafo que se genera en la red a medida que se descubre, existen diferentes enfoques, como aquellos basados en algoritmos clásicos de teoría de grafos [Krishna et al., 2011]. Otro de los métodos más ampliamente utilizados para la exploración de grafos en las redes informáticas hoy en día, dada su complejidad y dinamismo, es el uso de diferentes técnicas de aprendizaje automático que se adaptan a la topología cambiante de la red [Chami et al., 2022]. Este último paradigma para la exploración de grafos basada en aprendizaje automático se ha potenciado mediante la incorporación de técnicas avanzadas de aprendizaje profundo [Yang et al., 2021].

En esta tesis, en la sección 3.2, se propone un nuevo algoritmo basado en aprendizaje por refuerzo que cubre el punto de mejora detectado en la revisión de la literatura. Para optimizar el proceso de descubrir la topología de la red IoT a partir de información parcial o nula, asumiendo que se tiene un punto de entrada a partir del cual empezar a descubrir la red IoT, se ha modelizado este problema como un MDP. La novedad de esta investigación radica en asumir que no existe conocimiento previo de la red y que el proceso estocástico de obtención de datos de la red se modela mediante un MDP. Este enfoque permitirá a los administradores de red o proveedores de aplicaciones aprender dinámicamente la topología de una red desconocida de manera automática y con un bajo nivel computacional.

El objetivo de este caso de estudio es demostrar la eficiencia del algoritmo desarrollado en una red IoT diseñada para este propósito. Además, queremos demostrar que el algoritmo no necesita conocer nada de la red IoT o, en el mejor de los casos, disponer de información parcial para poder descubrir la topología de la red IoT. Las simulaciones que se han diseñado para este caso de estudio tienen como inicio un punto de entrada diferente, en concreto, se han seleccionado los nodos 6, 14 y 20 para ser los puntos iniciales de las simulaciones. En estas simulaciones, se quiere observar la recompensa y el número de pasos que necesita el algoritmo para descubrir completamente la red.

Como limitación de este caso de estudio, vamos a suponer que el número total de nodos de la red IoT es conocido para poder finalizar las simulaciones.

5.2.2 Descripción general del experimento

Para diseñar un entorno virtual en el que validar nuestra propuesta, empleamos una topología de red real que abarca varias subredes correspondientes a un edificio al que hemos obtenido acceso para su estudio, buscando reflejar condiciones lo más fieles posible a la realidad. Para poner a prueba el modelo propuesto, hemos seleccionado una red informática compleja perteneciente a un edificio real. El edificio seleccionado es el dedicado a Investigación y Desarrollo de la Universidad de Salamanca, sobre el cual hemos creado una réplica virtual de su red real. En esta configuración de red, distinguimos dos tipos de dispositivos informáticos: por un lado, los enrutadores y, por otro lado, cualquier otro dispositivo que se conecte y comunique con el resto de la red. Es crucial subrayar que el tipo de dispositivo ejerce una influencia considerable al obtener nueva información durante las pruebas de penetración en su interior.

La estructura de la red se conforma mediante la topología de interconexiones de los diversos equipos dentro del edificio. Por tanto, se establece un gráfico donde los nodos representan diferentes computadoras y los enlaces físicos entre ellos forman las aristas (sin alteración de la conexión entre dos nodos conectados por una arista). Un ejemplo ilustrativo de este gráfico se presenta en la Figura 5.1, donde los círculos indican los enrutadores y los cuadrados representan el resto del equipamiento informático. Este modelo de grafo es la representación de la topología de la red. La red se considerará como un entramado complejo y será de tipo no dirigido. Dado que, inicialmente, el algoritmo no posee conocimiento acerca de esta topología, construirá dinámicamente esta matriz de adyacencia añadiendo más nodos a medida que surjan tras cada fase de descubrimiento de información.

El experimento diseñado en este caso de estudio estará dividido en tres partes diferentes. En cada una de ellas se iniciará la simulación desde un punto inicial diferente, asumiendo que el algoritmo tiene un nivel de permisos suficientes para moverse por la red libremente. Esta condición que asumimos es posible ya que es consecuencia directa de asumir que el proceso de pentesting es automatico y siempre tiene exito, ya que queda fuera del alcance de nuestra investigación. Se espera que la recompensa total del algoritmo, la recompensa media en cada paso, así como el número de pasos que dedique el algoritmo a descubrir la red completa sean similares.

Respecto a la programación del algoritmo de aprendizaje por refuerzo, se ha elegido el entorno 'Taxi' que forma parte de los 'Toy Text environmnets' de Gym, de

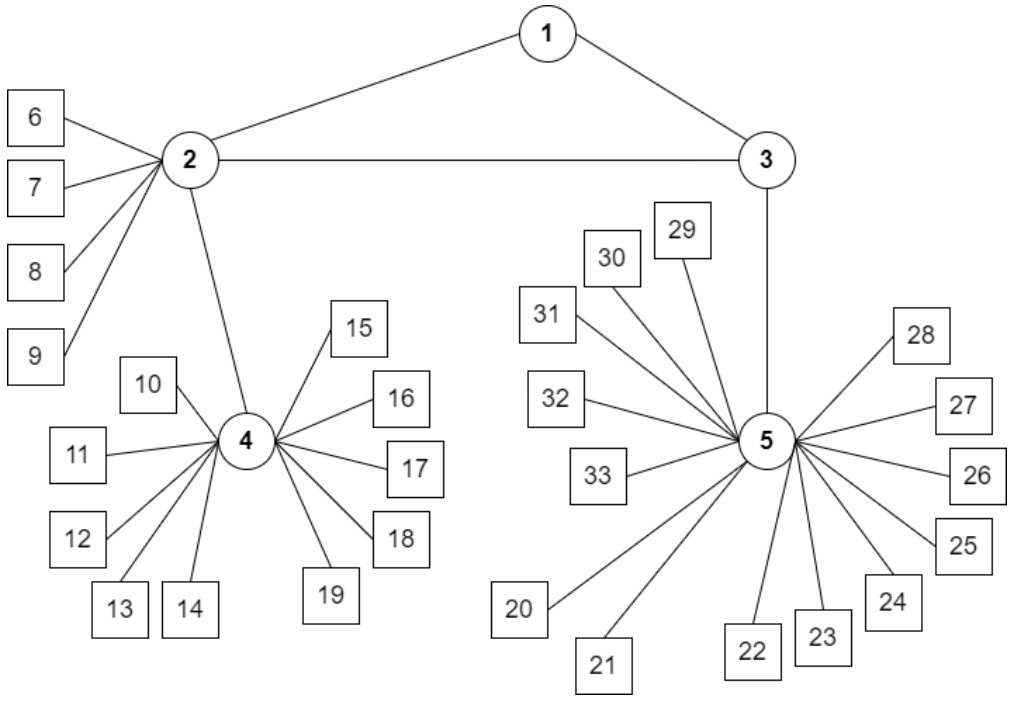


FIGURA. 5.1: Topología de red del caso de estudio.

la empresa OpenAi (https://www.gymlibrary.dev/environments/toy_text/taxi/). Este entorno es un caso práctico de la aplicación de un algoritmo por refuerzo en un plano, en el que el taxi tiene que ir cumpliendo ciertos objetivos pudiendo realizar un número de acciones concretas. Para importarlo en Python se usa el comando "gym.make("Taxi-v3"), una vez importado, lo que se hace es reescribir el entorno y las acciones. En la propia documentación del Gym se pueden encontrar fácilmente ejemplos y tutoriales de cómo personalizar los entornos (https://www.gymlibrary.dev/content/environment_creation/). Además, en una revisión rápida de la literatura se pueden encontrar ejemplos y código fácilmente [Habib, 2019, Saito et al., 2018].

La personalización principal que se hace es transformar el grafo en una superficie 2D. La forma de hacer esto es sacando la matriz de adyacencia del grafo de forma que donde haya ceros, es decir, no hay aristas que unan esos nodos del grafo, pondremos un espacio en negro que desde un punto de vista técnico significa que el agente no podrá pasar por ahí cuando realice sus acciones. De esta forma, las acciones que puede hacer el agente son moverse en todas las direcciones del entorno 2D, más concretamente arriba, abajo, izquierda, derecha y las 4 diagonales. Esto supone que el agente puede realizar 8 acciones que se codifican del 0 al 7. Antes de realizar las acciones, el agente tendrá que comprobar en la matriz de adyacencia si las puede realizar. Además, se eliminó la renderización del entorno 'GridWorldEnv', que es el que estamos utilizando, ya que consideramos que no era importante. Para el aprendizaje del algoritmo, en este caso

se ha utilizado una tabla Q con el algoritmo de aprendizaje asociado a las tablas Q, 'Q-learning' [Jang et al., 2019].

5.2.3 Resultados

En este caso de estudio se han propuesto tres simulaciones para su comparación. Se ha seleccionado estás simulaciones debido a que en el dieño de la red IoT, se han puesto tres subredes, y por tanto, se queria ver si el algoritmo tenía el mismo desempeño independientemente del punto de entrada o punto inicial en la red IoT. Se han comparado los resultados de las simulaciones durante el proceso de descubrimiento de la red IoT y se ha utilizado com metrica principal la recompensa total acumulada como se puede ver en la Figura 5.2. Aqui se puede ver que independientemente del punto de inicio del agente. Para las simulaciones realizadas, se han elegido como nodos iniciales el nodo número 14 perteneciente a la subred formada por los dispositivos número 10 a 14 y el enrutador número 4; el nodo número 20 perteneciente a la subred formada por los dispositivos número 20 a 21 y el enrutador número 5; y finalmente, el nodo número 6 perteneciente a la subred formada por los dispositivos número 6 a 9 y el enrutador número 2. Así se cubre cualquier subred del edificio representada en la topología de este caso de uso. De esta manera, podemos evaluar el rendimiento del algoritmo al comenzar en un nodo específica en lugar de otro. Además, cada vez que el algoritmo no encuentre nueva información en ninguno de los nodos, automáticamente regresará a un nodo visitado disponible desde su nodo actual para navegar y descubrir toda la topología de la red. Este proceso de volver a nodos ya visitados, penaliza al agente por visitar nodos ya conocidos, pero se le fuerza a hacerlo, ya que en caso contrario vimos que el agente no se movia del sitio como acción preferida ya que moverse a nodos conocidos le hacia bajar su recompensa total.

En la Figura 5.2, podemos ver el gráfico comparativo de la recompensa acumulada por el agente para cada una de las tres simulaciones óptimas al comenzar en cada nodo. Una vez vistos esta figura, se ve que los resultados de las recompensas totales del agente durante las tres simulaciones osn muy similares y al final de la simulación tienden a converger.

En la tabla 5.1, se pueden ver las diferentes medidas estadísticas calculadas para las recompensas acumuladas totales obtenidas en cada una de las tres simulaciones óptimas realizadas. De esta manera, es posible comparar si existe o no una diferencia notable al comenzar en un nodo específico y al comenzar en otro nodo perteneciente a otra subred.

FIGURA. 5.2: Recompensa obtenida en cada una de las tres simulaciones realizadas para cada paso.

Simulación	Nodo inicial	Desviación estandar	Recompensa total	Recomenpensa media	Pasos totales
1	14	66,85	810,99	13,74	59
2	20	65,44	855,37	14,25	60
3	6	66,26	836,35	13,93	60

TABLA. 5.1: Datos estadísticos recogidos de las tres simulaciones del caso de estudio I.

5.3 Caso de estudio II: Identificación de nodos infectados en una red IoT

5.3.1 Introducción

El malware es una gran amenaza de seguridad para el IoT, y por esta razón fundamental, la detección de malware que infecta las redes IoT es uno de los principales desafíos de seguridad [Faruk et al., 2021]. Esta razón radica en el hecho de que los dispositivos que forman una red IoT se caracterizan por compartir algunos problemas, en lo referente a la ciberseguridad, como el bajo consumo de energía, baja capacidad de almacenamiento o la baja capacidad de cómputo. Estas limitaciones mecánicas y electrónicas deben tenerse en cuenta ya que podrían representar una vulnerabilidad crítica al impedir la implementación de mecanismos avanzados de detección de malware para defenderse de intrusiones [Lvovich et al., 2020]. Por ejemplo, los dispositivos IoT

necesitan comunicarse entre ellos y con los dispositivos de control o almacenamiento de información. Por tanto, necesitan generar la suficiente "potencia" o voltaje para poder hacer estas comunicaciones inalámbricas. Aquí surge un problema en las comunicaciones con las interferencias. Por lo tanto, es esencial seleccionar un voltaje bajo pero suficiente para realizar las tareas de cambio de canal físico, procesamiento de información, intercambio de mensajes y defensa contra amenazas malware (si el dispositivo está preparado para esta tarea o tiene el poder de computación suficiente para llevarlo a cabo). Un sistema tradicional de detección de malware requeriría un voltaje que podría causar interferencias dentro de la red IoT, por lo que su uso no se recomienda [Aman et al., 2021]. Hay que tener en cuenta que muchas veces, por el propio diseño de los dispositivos IoT, el voltaje y/o la cantidad de batería no van a poder ser una variable que se pueda modificar, dificultar la aplicación de contramedidas o incluso dejar de funcionar porque el malware provoque que se acabe la batería. Por otro lado, una de las limitaciones inherentes al propio diseño de los dispositivos IoT es su, a veces nula, capacidad de almacenamiento. Por tanto, muchas veces, se puede explotar esa poca capacidad de almacenamiento para saturar la memoria de los dispositivos y que dejen de funcionar. Aunque esta característica propia del diseño también se vuelve en nuestra contra, debido a que tampoco se van a poder instalar parches de seguridad o programas antimalware. Es decir, tener un sistema tradicional de detección de malware requeriría un espacio considerable dentro de los diversos dispositivos IoT de la red para manejar información sobre amenazas conocidas y el estado de la red, lo cual es contrario al paradigma de la red IoT. [Zaza et al., 2020]

Con todos estos problemas, nos encontramos con que la falta de mecanismos de defensa a nivel de dispositivo y la gestión de seguridad en toda la red abren varias lagunas para la intrusión en la red y la infiltración de malware. Aunque antes de empezar a aplicar técnicas de defensa o mitigación, el problema que hemos identificado es que debido al enorme tamaño de las redes IoT actuales, no tendría sentido aplicar medidas de seguridad a gran escala como actualizaciones de software. Esto es debido principalmente a los costes económicos desorbitados que podrían tener que afrontar las empresas. Para resolver el problema de seguridad, algunas de las nuevas propuestas de investigación sobre técnicas de detección de malware se basan en la aplicación del Sistema Inmunológico Artificial (AIS, por sus siglas en inglés). Este método emula el comportamiento del sistema inmunológico humano, donde es capaz de defender de forma adaptativa a diferentes dispositivos contra el malware. El propio sistema de defensa es capaz de detectar diferentes amenazas, como intrusiones en dispositivos, caídas de red debido a ataques de conectividad o incluso diferentes tipos de malware que tienen como objetivo infectar dispositivos [Alrubayyi et al., 2023].

Otro de los paradigmas más utilizados actualmente para solucionar este problema de la detección de malware se basa en técnicas de machine learning. El uso del machine learning como método de detección de malware requiere una fase de entrenamiento con datos supervisados de ataques e infecciones anteriores en la red de dispositivos IoT para permitir que los modelos se ajusten para un rendimiento óptimo con buena precisión [Hussain et al., 2020, Madan et al., 2022]. Es importante que los modelos clasifiquen con precisión el tráfico de red malicioso y benigno para proteger los recursos críticos. Un modelo que informa un resultado falso negativo afirma que el tráfico de la red es benigno cuando, de hecho, era tráfico malicioso. Esto conducirá a eludir los controles de mitigación e infectar los nodos de IoT. De manera similar, un modelo que informa un resultado falso positivo también afecta el tráfico con buenas intenciones porque las herramientas y técnicas de mitigación de ataques bloquearán el tráfico de ataque benigno, considerándolo un escenario de ataque.

Las redes del IoT albergan una gran cantidad de dispositivos, lo que dificulta implementar contramedidas anti-malware efectivas y podría resultar costoso. Por ello, nuestra hipótesis principal es que la detección temprana de nodos infectados, antes de que el malware se propague por toda la red, permitiría aplicar medidas correctivas para detener su avance e incluso eliminarlo. Los recursos computacionales y de almacenamiento limitados de los dispositivos IoT restringen seriamente la aplicación de métodos tradicionales de detección de malware. Estos sistemas resultan ineficaces en entornos IoT debido a la heterogeneidad y tamaño de las redes. Esta situación propicia la aparición de diversas vulnerabilidades que los atacantes aprovechan para tomar control de los nodos, robar información, incorporarlos a redes botnet o secuestrar la red IoT [Alrubayyi et al., 2021]. Por estas razones, se necesita un cambio de paradigma para adaptarse dinámicamente a las características de la red y la evolución del malware. En este nuevo paradigma, las técnicas de detección de anomalías cobran relevancia, siendo el machine learning la más utilizada actualmente. Dentro de este campo, encontramos modelos de aprendizaje supervisado con datos etiquetados y modelos de aprendizaje no supervisado con datos sin etiquetar, ambos enfocados en distinguir entre el comportamiento normal de los nodos y patrones anómalos .

Una técnica específica dentro del aprendizaje automático es el aprendizaje por refuerzo, similar al aprendizaje no supervisado por no requerir datos etiquetados [Uprety and Rawat, 2020]. Esta técnica emplea un agente que toma decisiones óptimas para maximizar una recompensa y alcanzar el objetivo, en este caso, detectar malware en los nodos. El algoritmo propuesto en la sección 4.2 tiene por objetivo detectar nodos infectados en una red IoT. Para ello, partimos del problema de predicción para llegar al problema de control. Primero, abordamos la solución del método n-step TD bootstrapping, que obtiene la función de valor óptima para una política previamente

fijada. A continuación, utilizamos este resultado para extender la idea a los valores de acción y los métodos de control. Además, como el tamaño de las redes IoT puede variar, dotamos al algoritmo con la capacidad de adaptarse a dicho tamaño sin afectar su eficacia en la detección de nodos infectados, pudiendo incrementar el número de agentes que operan de forma colaborativa.

El objetivo de este caso de estudio es la detección temprana de los nodos que están infectados en una red IoT de manera que se puedan aplicar las contramedidas o medidas de mitigación oportunas de forma específica y no a gran escala. El aprendizaje por refuerzo ofrece una novedosa solución para detectar el malware en redes IoT, aprovechando las fortalezas del aprendizaje automático y adaptándose a las limitaciones y dinámica de estos entornos tan dinámicos. Se espera que este caso de estudio ponga a prueba el algoritmo en situaciones estresantes en las que el malware se va propagando por la red IoT mientras que el algoritmo trata de identificar los nodos infectados.

5.3.2 Descripción general del experimento

Para simular el rendimiento del algoritmo propuesto en un entorno que se asemeje lo más posible a una situación real, hemos desarrollado una simulación basada en un edificio inteligente equipado con diversos dispositivos IoT distribuidos en cada una de sus unidades residenciales. Esta red IoT comprende un total de 1000 dispositivos interconectados entre sí. Dentro de esta red simulada, encontramos una variedad de dispositivos IoT como sensores de temperatura, humedad, luminosidad, entre otros. La interconexión de estos dispositivos IoT dentro del edificio define la topología de la red, que puede ser representada como un grafo donde los vértices representan los dispositivos y las aristas las conexiones entre ellos.

Para llevar a cabo las simulaciones y los cálculos necesarios, disponemos de un equipo de hardware dedicado con capacidades robustas:

- Procesador: 24 núcleos con una memoria caché de 36 MB y una frecuencia máxima de hasta 5.8 GHz.
- Memoria RAM: DDR5 con una capacidad de 32 GB y una velocidad de 4800 MHz.

Este hardware proporciona el rendimiento necesario para ejecutar simultáneamente un número significativo de agentes dentro de la red IoT simulada, sin que el rendimiento de la CPU sea un factor limitante. Esto es esencial para garantizar que las simulaciones sean precisas y que puedan procesar adecuadamente la complejidad de la red IoT con sus múltiples interacciones entre dispositivos. El entorno de simulación está diseñado para

reflejar de manera precisa las condiciones de un entorno realista de red IoT en un edificio inteligente, utilizando recursos computacionales avanzados que permiten realizar cálculos detallados y exhaustivos sin comprometer la velocidad ni la precisión de los resultados obtenidos.

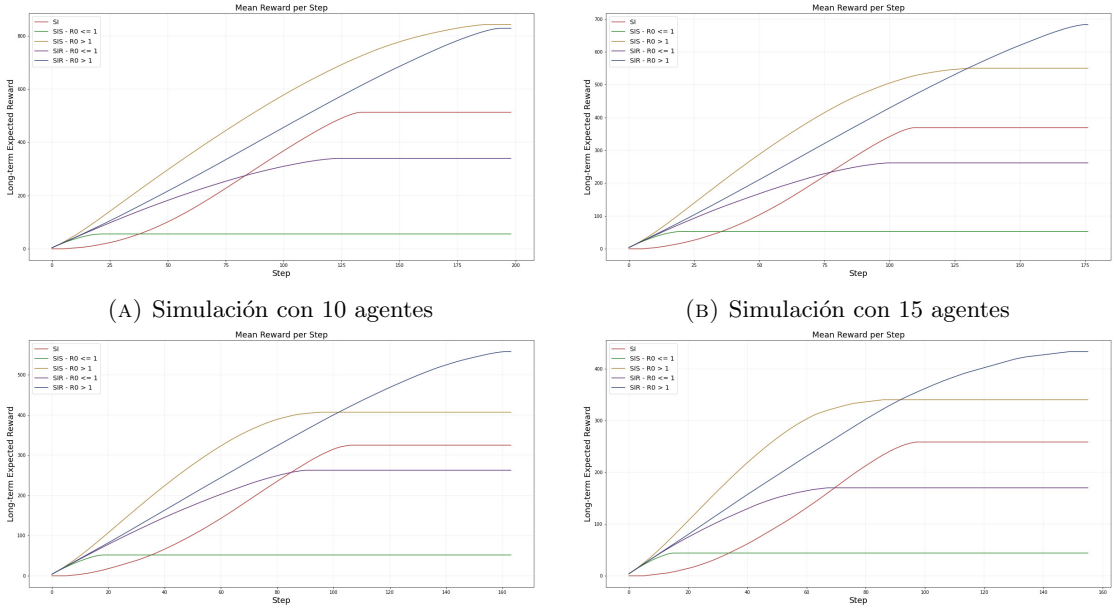
Para una evaluación exhaustiva del rendimiento y la efectividad del algoritmo, es crucial examinar detalladamente cómo se propaga el malware en un entorno realista. Con este propósito, hemos realizado simulaciones exhaustivas de la propagación del malware a través de los dispositivos de la red IoT, utilizando una variedad de algoritmos que pertenecen a las tres principales familias de modelos de propagación de malware: Susceptible-Infectado (SI), Susceptible-Infectado-Susceptible (SIS) y Susceptible-Infectado-Recuperado (SIR), todos bien documentados en la literatura por su capacidad para modelar diferentes escenarios de propagación de enfermedades y epidemias informáticas.

Además de estos modelos básicos, hemos introducido variantes que se diferencian en el número básico de reproducción (R_0), un indicador crítico que determina la velocidad y la extensión de la propagación del malware dentro de la red IoT. Específicamente, hemos considerado dos variantes en los modelos SIS y SIR, una para situaciones donde $R_0 > 1$ y otra para $R_0 \leq 1$, ya que estas condiciones tienen impactos distintos en la dinámica de la propagación del malware.

Es fundamental reconocer una limitación inherente al diseño experimental: la premisa de que el malware ya está presente y se está propagando en la red IoT antes de que se implemente el algoritmo de detección de nodos infectados. Aunque este enfoque simula una detección tardía del malware, permite explorar una gama completa de escenarios realistas que pueden influir en los resultados de las simulaciones. Estos resultados no solo validan la capacidad del algoritmo para detectar nodos infectados bajo condiciones diversas, sino que también proporcionan información crucial sobre cómo podría comportarse el malware en entornos dinámicos y heterogéneos como la red IoT. Mediante estas simulaciones detalladas y variadas, buscamos no solo probar el desempeño del algoritmo en escenarios realistas de propagación de malware, sino también proporcionar insights valiosos que puedan orientar el desarrollo de estrategias efectivas de ciberseguridad y mitigación de riesgos en entornos IoT cada vez más complejos y conectados.

5.3.3 Resultados

Los resultados obtenidos del rendimiento de prueba del algoritmo de detección de malware se presentan en la Figura 5.3. En esta figura, se puede encontrar el resultado



(A) Simulación con 10 agentes

(B) Simulación con 15 agentes

(C) Simulación con 20 agentes

(D) Simulación con 25 agentes

FIGURA. 5.3: Recompensa esperada obtenida en cada una de las simulaciones realizadas para cada paso de tiempo. La simulación se realizó aumentando el número de agentes para verificar el rendimiento del MARL.

de la aplicación del algoritmo en cuatro escenarios en los que cambiamos el número de agentes del MARL propuesto. En estas simulaciones, el algoritmo encuentra todos los nodos infectados independientemente del modelo de propagación de malware con el que ejecutamos las simulaciones. Se observa que una vez que el algoritmo ha identificado todos los nodos infectados en la red IoT, la recompensa del agente es cero, y por lo tanto, a partir de este punto, sus recompensas son franjas horizontales. En el caso de las simulaciones para el modelo de propagación SI, el modelo tarda un promedio de 100 pasos en detectar todos los nodos infectados. En aquellos correspondientes al modelo de propagación SIS con $R_0 > 1$, el algoritmo tarda un promedio de 90 pasos en detectar todos los nodos infectados, mientras que en el modelo de propagación SIS con $R_0 \leq 1$, el algoritmo encuentra todos los nodos infectados en un promedio de 25 pasos, ya que los nodos infectados tienden a disminuir con el tiempo. En aquellos correspondientes al modelo de propagación SIR con $R_0 > 1$, el algoritmo tarda un promedio de 175 pasos en detectar todos los nodos infectados, mientras que al ejecutar el modelo de propagación SIR con $R_0 \leq 1$, el algoritmo encuentra todos los nodos infectados en un promedio de 80 pasos, ya que los nodos infectados también tienden a disminuir, pero a una velocidad más lenta que en el modelo de propagación SIS. También es importante destacar que hay una variación en el tiempo promedio de detección cuando aumentamos el número de agentes. Esto se debe al hecho de que al aumentar el número de agentes, podemos cubrir y escanear diferentes partes del grafo de red simultáneamente, reduciendo así el tiempo que tarda nuestro algoritmo en detectar todos los nodos infectados.

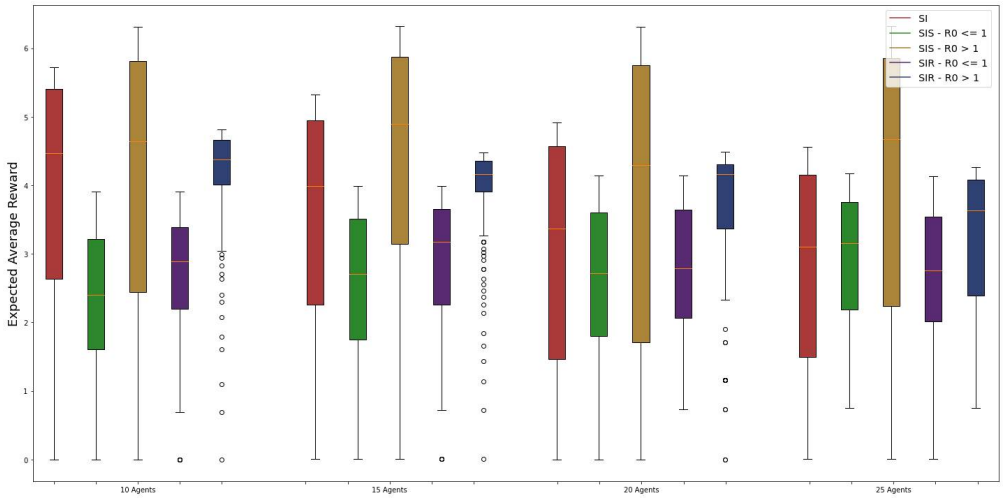


FIGURA. 5.4: Recompensa media del estado de acción al final de los episodios en las cuatro simulaciones con el aumento del número de agentes.

La Figura 5.4 muestra las medidas estadísticas de las recompensas obtenidas en cada paso de tiempo por sus cuartiles para cada modelo de propagación en cada una de las cuatro simulaciones. En cuanto a la variabilidad fuera de los cuartiles superior e inferior, observamos que para todos los modelos de propagación y simulaciones, hay una clara asimetría que favorece recompensas más pequeñas. Esto se debe a que cuanto menor sea el número de nodos que permanezcan sin detectar, menor será la recompensa. Eventualmente, la recompensa será 0 cuando todos los nodos con infecciones hayan sido detectados. Con respecto a la dispersión de los datos, se puede observar que es similar para todos los modelos de propagación en las cuatro simulaciones con diferentes números de agentes, excepto para el modelo de propagación SIR con $R_0 <= 1$. A medida que aumenta el número de agentes, aparece un rango más amplio de recompensas, aumentando así su dispersión. En cuanto a la recompensa promedio obtenida para cada modelo de propagación y cada simulación, se puede ver que aunque el 25% de las recompensas más altas disminuyen a medida que aumenta el número de agentes, el promedio permanece prácticamente sin cambios para todos. De manera similar, para los modelos de propagación que tienden a autodestruirse (SIS y SIR con $R_0 <= 1$), las recompensas promedio obtenidas por el algoritmo son prácticamente las mismas, mientras que para los otros tres modelos son muy similares. A medida que aumenta el número de agentes, la recompensa promedio obtenida por el algoritmo para cada modelo de propagación se vuelve más similar.

5.4 Caso de estudio III: Descubriendo el tipo de malware que ataca a una red IoT con datos reales

5.4.1 Introducción

En los últimos años, la proliferación de dispositivos del IoT ha conducido a un aumento en los ataques de denegación de servicio distribuido en redes IoT [Kumari and Jain, 2023]. Esto requiere métodos efectivos para clasificar el tráfico en redes IoT como benigno o tráfico que contiene trazas de un ataque DDoS. La metodología propuesta en la sección 4.3 no ha permitido comparar el rendimiento de algunos de los modelos de clasificación existentes más comunes en la tarea de detección de ataques DDoS, para determinar cuál tiene el mejor desempeño para detectar paquetes sospechosos de contener un ataque DDoS en una red IoT.

Ser capaz de clasificar de forma eficaz, eficiente y fiable los diferentes paquetes de red entre dispositivos de red IoT es esencial para determinar y hacer frente a posibles amenazas como los ataques DDoS. Estudios previos han propuesto diferentes técnicas de ML para esta tarea [Abbasi et al., 2021, Idrissi et al., 2020], como las redes neuronales convolucionales (CNNs), aquellas basadas en técnicas de Deep Leaning. Otros estudios recientes exploran la posibilidad de utilizar modelos clásicos de ML, como árboles de decisión, bosques aleatorios y SVMs [Ferrag et al., 2020]. Además, algunas investigaciones se han centrado en técnicas de selección de características y reducción de dimensionalidad para mejorar la precisión de la clasificación de paquetes en redes IoT [Pour et al., 2019]. Debido a la baja capacidad de recursos de los dispositivos IoT, estos ataques son especialmente complicados de detectar o mitigar en redes IoT. Otro paradigma ampliamente utilizado en la actualidad se basa en modelos híbridos, como un modelo que combina redes neuronales y SVM para la clasificación y detección de paquetes que forman parte de un ataque DDoS [Ahmed et al., 2023]. Este último paradigma, basado en modelos híbridos, también es muy utilizado para detectar problemas y posibles amenazas en la comunicación entre diferentes dispositivos IoT de la red, así como para detectar si el dispositivo está infectado y forma parte de una botnet [Mahboubi et al., 2020].

Una de las técnicas más habituales en el machine learning es, a partir de un dataset seleccionado, aplicar un conjunto de algoritmos seleccionado para buscar cuál es el más eficiente en la situación correspondiente. Debido a que la propagación del malware se va a comportar de forma diferente dependiendo de muchos parámetros, como por ejemplo el tipo de red IoT por la que se propague, la metodología diseñada en la sección 4.3 nos va a permitir la comparación de diferentes algoritmos de clasificación supervisada. De

esta forma, con esta metodología se podrá abordar de forma semi-supervisada una gran variedad de problemas de identificación de malware en diferentes redes IoT. A pesar de todos estos estudios y enfoques, la clasificación de paquetes de red en dispositivos IoT sigue siendo un área de investigación abierta. Hay muchos retos que deben seguir explorándose [Ferreira et al., 2023]. En este estudio, pretendemos contribuir a esta área de investigación realizando una evaluación exhaustiva de diferentes algoritmos de ML para la clasificación de paquetes en redes IoT.

Detectar y clasificar los ataques DDoS con métodos tradicionales resulta imposible debido al gran número de dispositivos en las diversas redes IoT y a la alta frecuencia de paquetes que reciben de manera constante. Estos enfoques no son adecuados para la naturaleza cambiante y heterogénea de este entorno [Munshi et al., 2020, Yang et al., 2020]. Por lo tanto, se requiere un cambio de paradigma que posibilite la clasificación dinámica, efectiva y eficiente de los paquetes, adaptándose a las características fluctuantes de la red. Dentro de este nuevo enfoque, es crucial determinar qué algoritmo de clasificación se ajusta mejor a las demandas de las redes IoT [Vishwakarma and Jain, 2020]. Estos modelos serán responsables de diferenciar entre paquetes benignos y aquellos asociados a ataques DDoS.

En esta tesis se propone este caso de estudio para determinar qué algoritmo del conjunto de algoritmos propuestos es el más idóneo para resolver el problema de clasificación de paquetes provenientes de una red IoT. Con este objetivo, se llevó a cabo el siguiente experimento. En primer lugar, se recolectó una gran cantidad de tráfico de una red IoT. En segundo lugar, se preparó el conjunto de datos para ser utilizado en el entrenamiento y prueba de los modelos. Luego, se dividió el conjunto de datos en un conjunto de prueba y uno de entrenamiento. Utilizando el conjunto de datos preparado, se entrenaron los diferentes modelos y se calcularon las métricas correspondientes. En este estudio, que involucra un conjunto de datos muy desbalanceado y una gran cantidad de información, se determinó cuál de los paradigmas es el más adecuado para abordar este problema de clasificación.

El objetivo de este caso de estudio es proporcionar una comparación detallada y basada en evidencias del desempeño de los modelos de machine learning en la detección de ataques DDoS en redes IoT. Los resultados obtenidos ayudarán a identificar el modelo más eficiente para este propósito, lo que beneficiará la seguridad y la protección de las redes IoT contra ataques maliciosos en las redes IoT. Hay que tener en cuenta que se ha elegido el ataque DDoS, ya que es uno de los ataques más estudiados y esto facilitaba la elección con argumentos basados en la literatura de los algoritmos que iban a componer el conjunto de algoritmos presentes en la metodología. Por otro lado, este caso de estudio lleva impuesto una limitación en su propio diseño, y es que si el malware no se conoce,

no se podrá hacer un dataset etiquetado, que es el requisito principal para poder aplicar la metodología diseñada en este trabajo.

5.4.2 Descripción general del experimento

En este caso de estudio, se presenta una comparación exhaustiva de nueve algoritmos de machine learning para la detección de ataques DDoS en redes IoT. La proliferación de dispositivos IoT ha incrementado la superficie de ataque, convirtiendo a estas redes en blancos atractivos para los ataques DDoS. La identificación precisa y oportuna de estos ataques es esencial para mantener la integridad y la disponibilidad de las redes IoT. Se han seleccionado nuevos modelos de machine learning ampliamente utilizados y reconocidos por su eficacia en la detección de anomalías y patrones en conjuntos de datos complejos. Los modelos seleccionados son: regresión logística, KNN, SVC, Naive Bayes, Decision Tree, Random Forest, AdaBoost, Gradient Boosting y Extra Tree. Estos modelos representan una variedad de enfoques y técnicas en el aprendizaje automático, lo que permite una evaluación comparativa integral de su desempeño en la detección de ataques DDoS en redes IoT.

Para evaluar la eficiencia de los modelos, se ha utilizado un conjunto de datos de ataques DDoS previamente recopilado de una red IoT simulada integrado con el dataset IoT-23, en concreto, la parte que trata de ataques DDoS. El experimento se ha diseñado de la siguiente manera. En primer lugar, se realizó el preprocesamiento de datos, es decir, se realizó una limpieza de datos para eliminar valores atípicos y completar los datos faltantes. Además, se llevó a cabo una normalización de características para asegurar una comparación justa entre los modelos. El siguiente paso previsto es la división de datos para entrenamiento y pruebas, es decir, el conjunto de datos se divide en conjuntos de entrenamiento y prueba en una proporción del 70-30%, respectivamente, para entrenar y evaluar el rendimiento de los modelos. Posteriormente, se hará el entrenamiento de modelos. Cada uno de los nueve modelos seleccionados se entrenará utilizando el conjunto de datos de entrenamiento. Por último, la fase de evaluación de modelos. Una vez entrenados, los modelos se evaluarán utilizando el conjunto de datos de prueba para medir métricas de desempeño, como precisión, recall, F1-score y accuracy. Las métricas de evaluación se utilizarán para comparar el rendimiento de los modelos en la detección de ataques DDoS en la red IoT. Estas métricas proporcionarán una comprensión detallada de la capacidad de cada modelo para distinguir entre tráfico normal y ataques DDoS, permitiendo así la identificación del modelo más eficiente para este propósito.

5.4.3 Resultados

Para probar los algoritmos bajo las mismas condiciones en el caso de estudio III, se dividio el mismo conjunto de datos en datos de prueba y entrenamiento. Con este propósito, el dataset se dividió en un 70% de paquetes de red para entrenamiento y un 30% para pruebas. Una vez hecha esta división se entrenaron los algoritmos obteniendo los resultados que se muestran en la Tabla 5.2. Aunque la metodología tiene prevista una parte de optimización de los hiperparametros de los algoritmos utilizados en esta metodología, debido al elevado número de muestras presentadas en este dataset, los algoritmos captaron los patrones sin ningún tipo de problema, por tanto, no fue necesario hacer este paso.

Algoritmo	Precision	Accuracy	Recall	TN Ratio	F1
Regresión Logística	0.97	0.97	1.0	0.00	0.98
KNN	1.00	1.00	1.0	0.99	1.00
SVC	0.97	0.97	1.0	0.00	0.98
Naive Bayes	0.97	0.97	1.0	0.00	0.98
Decision Tree	1.00	1.00	1.0	0.99	1.00
Random Forest	1.00	1.00	1.0	0.99	1.00
Ada Boost	1.00	1.00	1.0	0.99	1.00
Gradient Boosting	1.00	1.00	1.0	0.99	1.00
Extra Trees	1.00	1.00	1.0	0.99	1.00

Tabla. 5.2: Métricas de los algoritmos de clasificación.

La Tabla 5.2 muestra que todos los algoritmos tienen buen rendimiento y todos obtienen puntuaciones próximas al 100% en precisión, recuperación y F1. Sin embargo, algunos algoritmos tienen ratios TN tan bajos como 0%. A primera vista, podría pensarse que todos los modelos mencionados son, por tanto, adecuados para la clasificación del tráfico paquetes en la red. Sin embargo, esto sería un error, ya que el dataset está muy desbalanceado.

Los dataset en los que los algoritmos tienden a ser más precisos en la clase mayoritaria y menos precisos en la minoritaria presentan varios retos para el entrenamiento y la evaluación de los modelos de clasificación [Ganganwar, 2012]. Esto se debe a que el modelo tiene un sesgo hacia la predicción de la clase mayoritaria en lugar de la minoritaria, y el resultado es un sesgo hacia la clase mayoritaria. En este caso, el 98% de los paquetes se clasificarán como benignos y el 2% como paquetes de ataque DDoS. Además, la selección de la métrica de evaluación adecuada puede verse afectada por conjuntos de datos desbalanceados. La precisión, que mide la proporción de predicciones correctas en comparación con todas las predicciones, puede no ser la mejor métrica en un conjunto de datos desbalanceados. Esto se debe a que el modelo puede ser muy preciso en la clase mayoritaria, pero muy impreciso en la clase minoritaria. Por lo tanto,

FIGURA. 5.5: Matriz de confusión del algoritmos Random Forest.

es importante elegir una métrica que tenga en cuenta el desequilibrio del conjunto de datos, como el ratio de TN, es decir, el ratio de falsos negativos que tiene el algoritmo, cuanto más alto sea, menos TN detecta, y cuando mas bajo sea, mayor número de TN ha detectado. Teniendo en cuenta estas métricas y la naturaleza desbalanceada del dataset descrito anteriormente, podemos ver cómo los algoritmos basados en modelos lineales, SVM y probabilísticos son incapaces de clasificar correctamente el tráfico DDoS, dando como resultado un ratio TN de 0. Por otro lado, podemos ver cómo los modelos basados en árboles, ensemble (bagging y boosting) resuelven el problema de clasificación con métricas muy próximas al 100%.

Se ha evaluado la matriz de confusión del Random Forest, ya que se ha seleccionado como el algoritmo con el mejor rendimiento para esta tarea de clasificación en el caso concreto del tráfico de red con ataques DDoS. Como se puede ver en la Figura 5.5, el Random Forest es capaz de clasificar correctamente el 2.98% de los casos como casos positivos de ataque DDoS, el 96.99% de los casos como casos negativos, el 0% como falsos positivos (son 3 casos, pero el total de casos de entrenamiento es aproximadamente 120.000 casos, por tanto, estadisticamente esta próximo al 0%) y el 0.02% como casos de falsos negativos. Como se puede ver, este dataset esta muy desbalanceado, con un ratio de desbalanceo de 1:33, es decir, por cada caso de ataque DDoS, hay 33 casos de tráfico benigno. Usualmente, se suelen utilizar técnicas de oversampling [Wongvorachan et al., 2023], en este caso se ha decidio no usarlo, ya que los algoritmos han tenido muy buen rendimiento.

5.5 Caso de estudio IV: Identificar el tipo de malware que ataca a una red IoT con datos sintéticos

5.5.1 Introducción

En una era de tecnología de la información y conectividad en constante expansión, debido principalmente al desarrollo del IoT, la ciberseguridad se ha convertido en una preocupación clave [Khan and Chowdhury, 2021]. Se ha creado un entorno propicio para la proliferación de amenazas cibernéticas debido a la creciente interconexión de dispositivos y sistemas a través de redes de IoT [Sadhu et al., 2022, Wu et al., 2020]. Esta interconectividad, mediante la integración de dispositivos inteligentes en redes a gran escala, ha permitido la transformación prácticamente de todos los aspectos de la sociedad, desde la industria y la atención médica hasta la vida cotidiana de las personas. Con esta revolución tecnológica han surgido desafíos de ciberseguridad [Clim et al., 2022]. Han surgido nuevos ataques y vulnerabilidades debido a la amplitud y diversidad de los dispositivos IoT y su capacidad para intercambiar datos y comunicarse entre sí. Una de las amenazas más pervasivas y difíciles de prevenir es el malware [Aslan et al., 2023]. El malware viene en muchas formas, desde software malicioso que roba información sensible hasta código diseñado para deshabilitar sistemas y lanzar ataques DDoS [Mittal et al., 2023]. Para garantizar la seguridad, privacidad y confiabilidad de las redes IoT, la necesidad de detectar y mitigar eficazmente estas amenazas se ha convertido en un imperativo crítico. Los ciberataques pueden tener consecuencias devastadoras, desde la interrupción de servicios esenciales hasta la exposición de datos sensibles y el menoscabo de la confianza en los sistemas digitales [Victoire et al., 2023].

A medida que el ecosistema IoT continúa creciendo (y con este crecimiento, los retos de seguridad asociados), la necesidad de una defensa efectiva contra el malware y otras ciberamenazas es más urgente que nunca. Debido a que obtener datos realistas de malware atacando las redes IoT puede ser costoso desde el punto de vista logístico y económico, es muy común utilizar modelos teóricos para hacer simulaciones. Estos modelos teóricos están basados en la epidemiología matemática [Kermack and McKendrick, 1927]. La aplicación de modelos epidemiológicos adaptados a la propagación de malware en redes IoT ofrece un enfoque prometedor para comprender y mitigar estas amenazas [del Rey et al., 2022]. Mediante el uso de conceptos y métodos de la epidemiología, es posible tener en cuenta la dinámica de la propagación de malware y desarrollar estrategias de defensa proactivas. Identificar el malware que se propaga a través de las redes IoT es crucial para desarrollar estrategias efectivas de mitigación de ciberataques. Las metodologías existentes implican la estimación de parámetros

en modelos epidemiológicos; sin embargo, estimar estos parámetros es desafiante debido a las dificultades inherentes para comprender y modelar las características del malware [Ferrández et al., 2023]. Identificar con precisión los parámetros del malware, como las tasas de propagación y recuperación, es esencial para anticipar el comportamiento e implementar contramedidas.

La disponibilidad limitada de datos reales de ataques complica los esfuerzos de ciberseguridad. Debido a la naturaleza sensible de la información y a la falta de acceso a registros completos de incidentes de seguridad, obtener conjuntos de datos representativos se convierte en una tarea desalentadora. Esto dificulta las capacidades de los métodos de estimación tradicionales, que a menudo se basan en conjuntos de datos grandes y diversos para producir resultados confiables. La falta de datos relevantes puede conducir a estimaciones inexactas de los parámetros, limitando en última instancia la efectividad de las estrategias de defensa basadas en modelos. En general, superar los desafíos de la estimación de parámetros en ciberseguridad requiere enfoques que puedan manejar la incertidumbre y los datos incompletos. El uso de métodos adaptativos y flexibles es esencial para el desarrollo de sistemas de ciberdefensa robustos y confiables en un entorno en constante evolución [Shandilya et al., 2022]. Los modelos epidemiológicos, como los enfoques SIR y SIRS, se utilizan para comprender y predecir la propagación de enfermedades en poblaciones. El modelo SIR divide a los individuos en tres compartimentos, a saber, susceptibles, infectados y recuperados, asumiendo que cuando se recuperan no pueden ser reinfectados. El modelo SIRS agrega un compartimento adicional para individuos que vuelven a ser susceptibles después de la recuperación. Estos modelos se pueden adaptar para analizar la propagación de malware en redes IoT, donde los dispositivos se consideran susceptibles, infectados y recuperados. La propagación del malware depende de factores como la interacción entre dispositivos y la efectividad de la defensa. La adaptación de los modelos SIR y SIRS a la ciberseguridad proporciona una comprensión más profunda de la propagación del malware y un marco para estimar parámetros críticos como la tasa de infección y la tasa de recuperación. Esta adaptación ofrece una nueva perspectiva para analizar y diseñar estrategias de defensa contra amenazas cibernéticas en un entorno cada vez más conectado.

Un paso esencial para identificar estos modelos epidemiológicos a través de datos reales, es la resolución del problema inverso junto con la estimación de parámetros del modelo de propagación como la tasa de transmisión y la tasa de recuperación en el modelo SIR. Existen varios métodos en la literatura que han sido ampliamente utilizados por los investigadores para estimar estos parámetros, incluyendo ajuste de curvas [Furtado, 2021, Wong and Juwono, 2022], mínimos cuadrados [Cantó et al., 2017, Marinov et al., 2014], máxima verosimilitud [Piazzola et al., 2021, Zang et al., 2015], cadena de Markov de Monte Carlo [da Silva et al., 2020, Taddy et al., 2009], y PINNs [Grimm

et al., 2022, Schiassi et al., 2021], por mencionar algunos. El método elegido para la estimación depende de los datos disponibles, la complejidad del modelo y la naturaleza de la epidemia. En la práctica, varias aproximaciones suelen combinarse para obtener estimaciones más precisas y robustas de los parámetros del modelo de propagación. Ciertos métodos, como PINNs y los métodos basados en Monte Carlo, tienen ventajas muy útiles cuando solo se dispone del número de infecciones a lo largo del tiempo. Estas incluyen adaptabilidad a datos irregulares, incorporación de ecuaciones físicas, escalabilidad, estimación de incertidumbre, o exploración del espacio de parámetros, entre otras. Sin embargo, en los mismos casos, donde todo lo que está disponible es el número de infecciones a lo largo del tiempo, la máxima verosimilitud y los mínimos cuadrados también tienen desventajas relevantes, a saber, alta sensibilidad a valores atípicos, falta de información sobre incertidumbre, no optimalidad para distribuciones no normales, necesidad de conocer la distribución, falta de una función de verosimilitud disponible, etc. Por esta razón, decidimos comparar PINNs y métodos de Monte Carlo en este estudio, excluyendo los métodos de máxima verosimilitud y mínimos cuadrados de la comparación. Comparar métodos basados en técnicas de Monte Carlo y PINNs para la estimación de parámetros en modelos SIR/SIRS proporciona una oportunidad única para determinar qué enfoque puede proporcionar una defensa robusta y eficiente en la detección y mitigación de malware en redes IoT.

En esta tesis, en la sección 4.4, se propone una nueva metodología para identificar el malware que se propaga por una red IoT cuando no se tengan datos de este, y, por tanto, la metodología propuesta en la sección 4.3 falle. En este caso, se propone resolver el problema inverso, y ya que los modelos epidemiológicos globales vienen determinados por ecuaciones diferenciales, si se conocen los parámetros de estas, es muy sencillo resolverlas y de esta forma tratar de identificar el modelo epidemiológico que mejor modelice al malware que se está propagando por la red IoT. En este caso, se han diseñado dos algoritmos basados en técnicas Monte Carlo y se ha utilizado PINN para la estimación de los parámetros. En este caso de estudio, se proponen varios escenarios para probar la eficiencia de los métodos de estimación de parámetros propuestos y realizar una comparativa entre ambos.

El objetivo principal de este caso de estudio es realizar una comparación exhaustiva entre dos métodos de estimación de parámetros basados en técnicas de Monte Carlo y un PINN en el contexto de modelos SIR/SIRS adaptados para describir la propagación de malware en redes IoT.

5.5.2 Descripción general del experimento

Para comparar los métodos propuestos, se generaron datos sintéticos para los modelos de propagación de malware SIR y SIRS; elegimos estos modelos porque tienen los mismos compartimentos. Se utilizó la función odeint de Python para generar los datos sintéticos. Esta función aplica el método numérico de Runge-Kutta $(4, 5)$ a los sistemas de ecuaciones de los modelos de propagación, y como resultado proporciona las secuencias de soluciones. En el caso del modelo SIR, proporciona las soluciones de susceptibles (S), infectados (I) y recuperados (R); sin embargo, para esta comparación, solo nos interesan los infectados, ya que en el mundo real generalmente solo se tiene acceso al número de dispositivos infectados (o al menos suele ser la medida más común que se consigue en el análisis forense de la red IoT). Los modelos de propagación de malware en el intervalo de tiempo $t \in [0, 2000]$.

Para comparar los dos métodos para la resolución del problema inverso para la estimación de los parámetros, se realizaron los siguientes pasos de preprocesamiento de datos: (1) se generaron datos de infectados sintéticos utilizando el modelo de propagación de malware en una red IoT simulada, variando los parámetros de interés, es decir, tasa de contagio, tasa de recuperación, etc.; (2) se añadió ruido artificial a las curvas infectadas para simular condiciones del mundo real y aumentar la complejidad de la tarea de estimación utilizando la función 'random.normal' de NumPy, y se programó el método de Monte Carlo (con ambas funciones de pérdida) en Python creando dos funciones personalizadas, como se muestra en los pseudocódigos de los algoritmos 2 y 3 en la sección 4.4.2.1. Ambos métodos Monte Carlo se usaran sobre los datos creados con los modelos SIR y SIRS para estimar los parámetros.

Por otra parte, se estimarán los parámetros con la PINN en Python usando la biblioteca deepxde para codificar los PINNs [Lu et al., 2021] en un ordenador personal (CPU: i7-8700 a 3.20 GHz; Memoria: 16 GB; SO: Microsoft Windows 10 de 64 bits). La arquitectura de las PINN utilizadas tiene una capa de entrada de una sola neurona, tres capas ocultas de 40 neuronas cada una, y una capa de salida de tres neuronas. La función de activación es tanh y la inicialización de los pesos de la red neuronal es Glorot uniform, el optimizador es Adam y la tasa de aprendizaje de 0.001. Además, la tasa de contagio β y la tasa de recuperación γ son las variables entrenables externas que tiene que estimar la PINN. Las PINNs se entrenan con 10,000 iteraciones en los datos proporcionados en cada bucle de entrenamiento de estimación de parámetros de algoritmo. Finalmente, antes de comenzar los experimentos para ambos escenarios, se entrenaron dos PINNs, uno con el modelo SIR y otro con el modelo SIRS. Por lo tanto, ambas PINNs fueron utilizadas en ambos escenarios.

Una vez estén hechas todas las simulaciones y pruebas de estimación de parámetros con ambos métodos, se evaluará la efectividad de ambos métodos para decidir cuál es el método más eficiente para esta tarea.

5.5.3 Resultados

En esta sección de resultados se van a evaluar dos simulaciones para la estimación de parámetros con datos sinteticos a partir del modelo SIR y del modelo SIS de epidemiología matemática.

5.5.3.1 Estimación de parámetros para el modelo SIR

En este caso de estudio, se considera un malware propagándose a través de una red IoT según un modelo SIR. Para esta simulación, las condiciones iniciales del modelo SIR han sido $S(0) = 0.99, I(0) = 0.01$, y los parámetros del modelo SIR han sido $\beta = 0.8$ y $\gamma = 0.25$ con un rango de tiempo $t \in [0, 2000]$. La Figura 5.6 muestra los resultados del método de estimación de parámetros basado en el Error Cuadrático Medio Monte Carlo (MC-MSE). Se puede observar que el método MC MSE estima los parámetros después de 52 s (tiempo computacional) y construye la curva de infectados para los modelos SIR (izquierda) y SIRS (derecha). El método MC-MSE identifica correctamente que el malware en este experimento sigue el modelo SIR, como se puede ver en esta figura. Por otro lado, en las mismas condiciones de la simulación del modelo SIR, se ha estimado los parámetros con el Monte Carlo con una función de pérdida log-cuadrada (MC-SL por sus siglas en ingles), que termina su ejecución después de 56 s consiguiendo el MC-SL unos resultados similares al caso del MC-MSE(ver Figura 5.7).

Los resultados obtenidos en este caso de estudio por las PINNs entrenadas con el modelo SIR (Figura 5.8 arriba a la izquierda) y con el modelo SIRS (Figura 5.8 arriba a la derecha) muestran que son capaces de estimar los parámetros de tal manera que los datos de infectados producidos por la PINN sea igual a los medidos en la red IoT. Sin embargo, en la estimación de parámetros, la PINN entrenada con los datos sinteticos generados con el modelo SIR estima correctamente los parámetros (Figura 5.8 abajo a la izquierda). Por otro lado, la PINN entrenada con los datos generados por el modelo SIRS (Figura 5.8 abajo a la derecha) estima correctamente β y γ, mientras que para δ proporciona un valor diferente en cada simulación que es muy cercano a 0 y a veces incluso negativo. Para ilustrar este ejemplo, hemos tomado el promedio del δ obtenido en diez simulaciones. La PINN entrenada con el modelo SIR tardó 154 s en estimar los parámetros, mientras que la PINN entrenado con el modelo SIRS tardó 163 s.

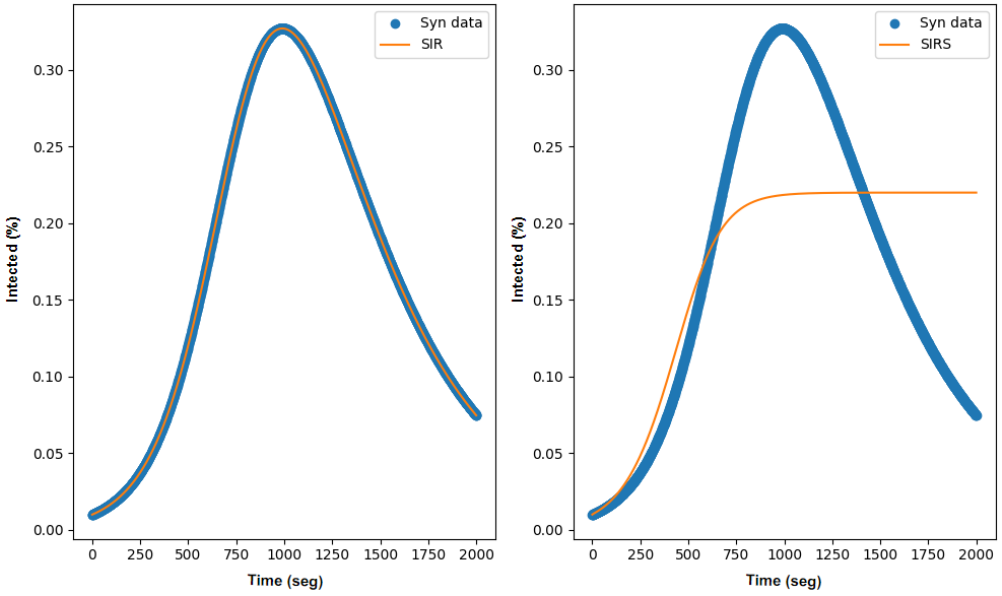


FIGURA. 5.6: Datos generados sintéticamente según el modelo SIR con parámetros $\beta = 0.8$ y $\gamma = 0.25$ (azul). Los datos fueron generados utilizando parámetros calculados utilizando el método MC-MSE para el modelo SIR y el modelo SIRS (naranja).

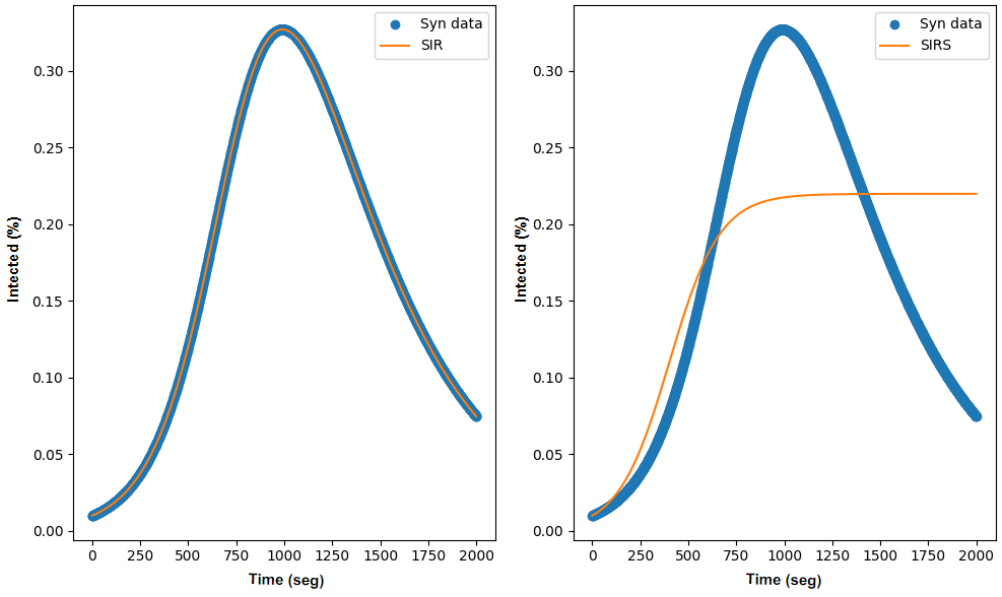


FIGURA. 5.7: Datos generados sintéticamente según el modelo SIR con parámetros $\beta = 0.8$ y $\gamma = 0.25$ (azul). Los datos fueron generados utilizando parámetros calculados utilizando el método MC-SL para el modelo SIR y el modelo SIRS (naranja).

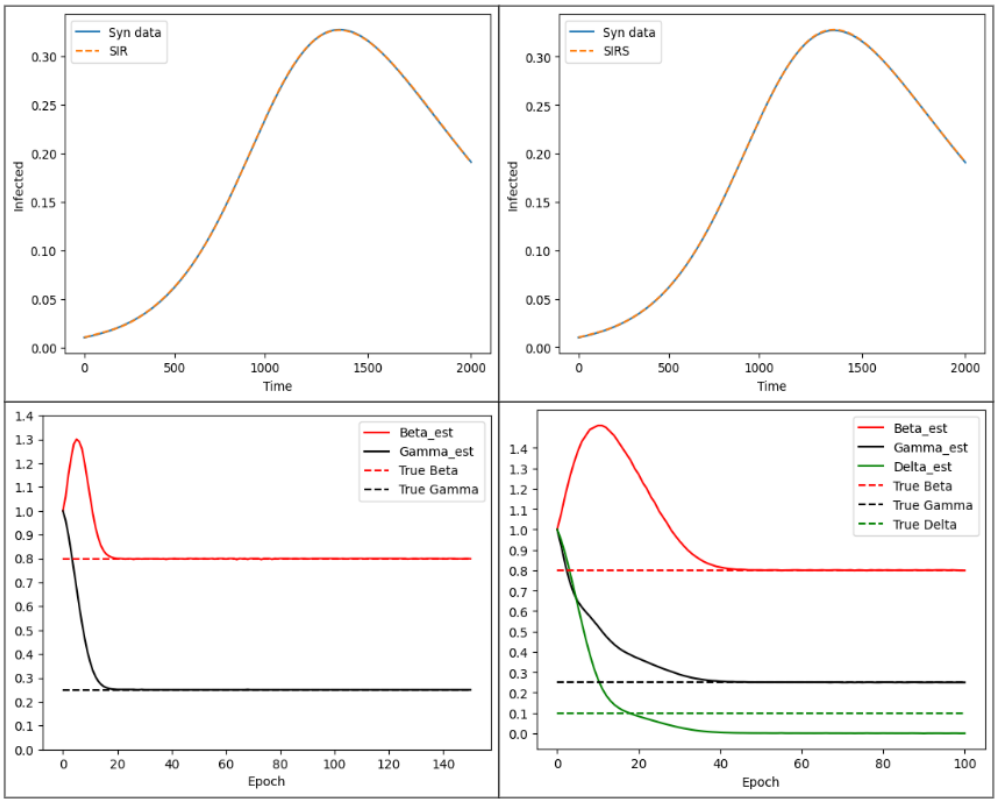


FIGURA. 5.8: Arriba: datos generados sintéticamente según el modelo SIR con parámetros beta = 0.8 y gamma = 0.25 (azul). Los datos fueron creados con los parámetros calculados por la PINN para el modelo SIR y el modelo SIRS (naranja). Abajo: parámetros estimados por las PINNs entrenadas con el modelo SIR (izquierda) y el modelo SIRS (derecha).

Finalmente, se puede ver una comparación de las estimaciones de los tres métodos comparados en este estudio en la Tabla 5.3. Ambos métodos basados en Monte Carlo estiman correctamente los parámetros del modelo SIR mientras encuentran parámetros para el modelo SIRS que están fuera de los límites de estos parámetros, es decir, $0 \leq \beta, \gamma, \delta \leq 1$. Por otro lado, PINN-SIR identifica correctamente los parámetros, mientras que PINN-SIRS no puede estimarlos, como se esperaba basado en su entrenamiento.

5.5.3.2 Estimación de parámetros para el modelo SIRS

En este caso de estudio, se considera un malware propagándose a través de una red IoT según un modelo SIRS. En esta simulación, las condiciones iniciales fueron $S(0) = 0.99, I(0) = 0.01$ y los parámetros fueron $\beta = 0.8, \gamma = 0.25$ y $\delta = 0.1$ con un rango de tiempo $t \in [0, 2000]$. La Figura 5.9 muestra los resultados del método de estimación de parámetros MC MSE. Se puede observar que el método MC MSE estima los parámetros después de 53 s y genera correctamente los datos de infectados con los modelos SIR (izquierda) y SIRS (derecha). El método MC MSE no puede identificar que el malware

Method	SIR	SIRS
MC MSE	$\beta = 0.8$ $\gamma = 0.25$	$\beta = -1.49$ $\gamma = -1.49$ $\delta = 1.46$
MC LS	$\beta = 0.8$ $\gamma = 0.25$	$\beta = -18.71$ $\gamma = -18.71$ $\delta = 18.69$
PINN	$\beta = 0.8$ $\gamma = 0.25$	$\beta = 0.8$ $\gamma = 0.25$ $\delta = 5.4 \times 10^{-4}$

TABLA. 5.3: Los parámetros estimados del modelo SIR y SIRS se obtuvieron mediante cada método comparado. Resaltados en rojo están los valores imposibles para los parámetros, ya que los parámetros están acotados $0 \leq \beta, \gamma, \delta \leq 1$.

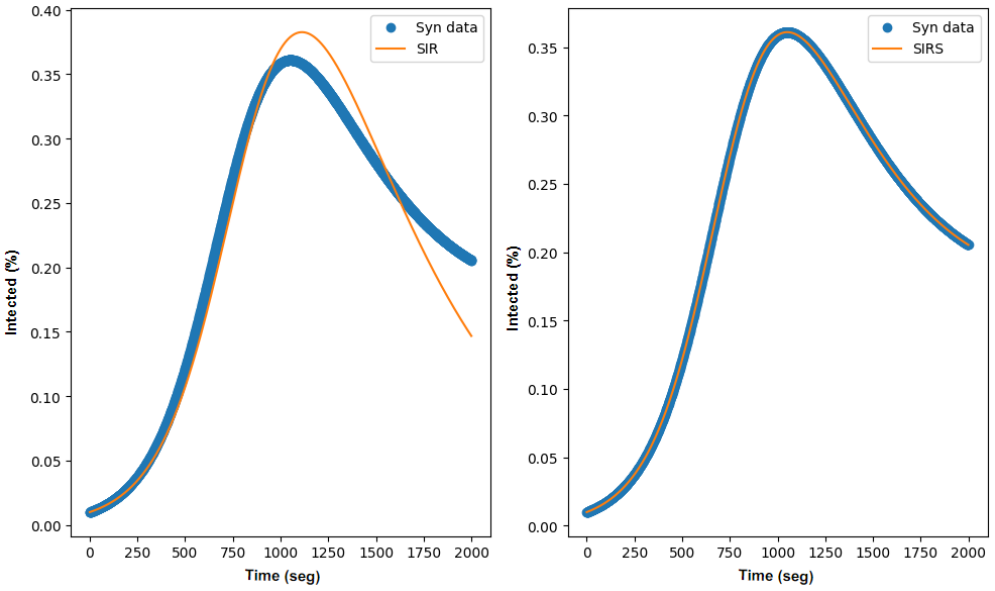


FIGURA. 5.9: Datos generados sintéticamente según el modelo SIRS con parámetros $\beta = 0.8$, $\gamma = 0.25$ y $\delta = 0.1$ (azul). Los datos fueron generados utilizando parámetros calculados utilizando el método MC-MSE para el modelo SIR y el modelo SIRS (naranja).

en esta simulación que sigue un modelo SIR, aunque si puede identificar correctamente los parámetros de un modelo SIRS. Por otro lado, en las mismas condiciones de la simulación del modelo SIRS, se ha estimado los parámetros con el MC-LS, que termina su ejecución después de 71 s consiguiendo el MC-SL unos resultados totalmente diferentes al caso del MC-MSE(ver Figura 5.10) ya que este método no es capaz de estimar los parámetros correctamente y por tanto, este método no es capaz de identificar el malware que se está propagando por la red IoT.

Los resultados obtenidos en este caso de estudio por las PINNs entrenadas con el modelo SIR (Figura 5.11 arriba a la izquierda) y el modelo SIRS (Figura 5.11 arriba a la derecha)

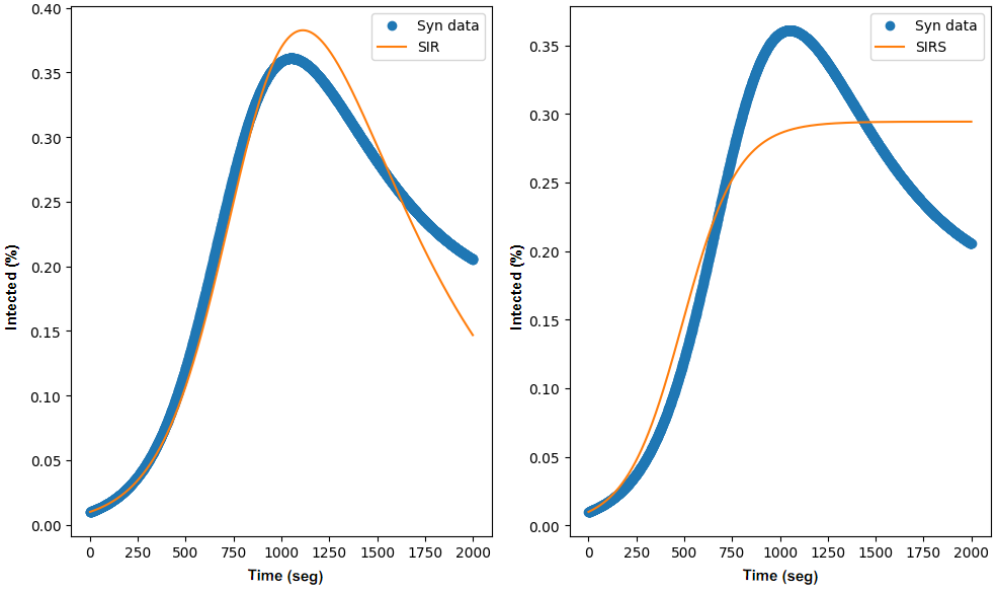


FIGURA. 5.10: Datos generados sintéticamente según el modelo SIRS con parámetros $\beta = 0.8$, $\gamma = 0.25$ y $\delta = 0.1$ (azul). Los datos fueron generados utilizando parámetros calculados utilizando el método MC-SL para el modelo SIR y el modelo SIRS (naranja).

muestran que son capaces de estimar los parámetros de tal manera que los infectados producidos por la PINN sea igual a los medidos en la red IoT, aunque la curva de infectados generada por PINN-SIR no es exactamente similar a la medida de infectados realizada en la red. En las tareas de estimación de parámetros, la PINN entrenada con el modelo SIRS estima correctamente los parámetros (Figura 5.11 abajo a la izquierda), mientras que la PINN entrenada con el modelo SIR (Figura 5.11 abajo a la derecha) no puede estimar β o γ. La PINN entrenada con el modelo SIR tomó 152 s para estimar los parámetros, mientras que el PINN entrenado con el modelo SIRS tomó 157 s. Finalmente, se puede ver una comparación de las estimaciones de los tres métodos comparados en este estudio en la Tabla 5.4. El método MC-MSE estima correctamente los parámetros del modelo SIRS mientras encuentra parámetros para el modelo SIRS que están fuera de los límites de estos parámetros, es decir, $0 \leq \beta, \gamma, \delta \leq 1$. Sin embargo, el modelo de MC-LS no puede estimar los parámetros. Por otro lado, PINN-SIRS identifica los parámetros, mientras que PINN-SIR no puede estimarlos, como se esperaba basado en su entrenamiento. Esta simulación nos ha demostrado que el método MC-MSE es mas robusto en las simulaciones que hemos utilizado en esta segunda parte del caso de estudio. Aunque solo hemos diseñado los métodos basados en Monte Carlo con los errores MSE y LS, en trabajo futuro se podría considerar probar otros errores o mejorar la robusted de la metodología actual.

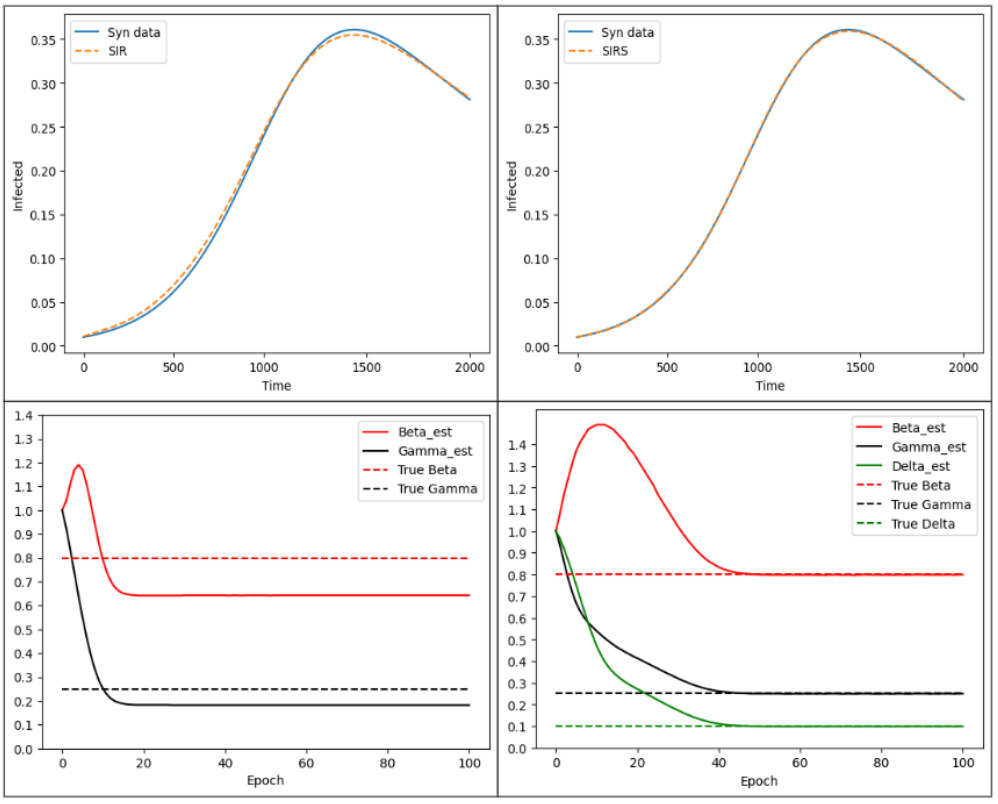


FIGURA. 5.11: Arriba: datos generados sintéticamente según el modelo SIRS con parámetros beta = 0.8 y gamma = 0.25 (azul). Los datos fueron creados con los parámetros calculados con un PINN para el modelo SIR y el modelo SIRS (naranja). Abajo: parámetros estimados por las PINNs entrenados con el modelo SIR (izquierda) y el modelo SIRS (derecha).

Method	**SIR**	**SIRS**
MC MSE	$\beta = 0.7$ $\gamma = 0.187$	$\beta = 0.8$ $\gamma = 0.25$ $\delta = 0.1$
MC LS	$\beta = 0.7$ $\gamma = 0.18$	$\beta = 1.9$ $\gamma = 1.22$ $\delta = 5.83$
PINN	$\beta = 0.7$ $\gamma = 0.187$	$\beta = 0.8$ $\gamma = 0.25$ $\delta = 0.1$

TABLA. 5.4: Los parámetros estimados del modelo SIR y SIRS se obtuvieron mediante cada método comparado. Resaltados en rojo están los valores que no pueden tomar los parámetros ya que están acotados $0 \leq \beta, \gamma, \delta \leq 1$.

Capítulo 6

Conclusiones y líneas de investigación futuras

Capítulo 6

Conclusiones y líneas de investigación futuras

6.1 Introducción

En este capítulo se describe cómo se han alcanzado los distintos objetivos definidos en esta investigación para validar y evaluar la hipótesis de partida: "*La hipótesis de este trabajo de investigación es que es posible mejorar y optimizar las técnicas y tecnologías actuales para la identificación de diversos tipos de malware que se propagan a través de redes IoT.*"

Esta investigación ha presentado un conjunto de algoritmos y metodologías para la mejorar la detección de la propagación del malware en una red IoT. La novedad principal que aporta esta investigación esta dividida en dos partes. La primera es que se parte de que ya no es necesario tener conocimiento total de la topología de la red IoT para poder descubrir los nodos IoT infectados por malware. La segunda parte de la novedad principal de esta investigación es que se proponen dos alternativas para detectar el malware en una red IoT en caso de que este sea conocido, es decir, se tenga ya mucha información de dicho malware. En caso contrario, se propone una modelización teorica del nuevo malware para poder recabar información y construir nuevos datasets para alimentar a la metodología de detección propuesta en esta misma tesis. La investigación presentada en esta tesis ha contado con el diseño de cuatro casos de estudio que han servido para validar las posibles casuisticas que tuviera cada uno de los algoritmos o metodologías propuestas.

El resto del capítulo esta organizado como sigue: la sección 6.2 presenta las conclusiones finales de esta tesis, la sección 6.3 muestra las aportaciones al estado del arte y, por último, la sección 6.4 presenta el trabajo futuro de este trabajo de investigación.

6.2 Conclusiones finales

En este apartado se presentan las conclusiones finales de esta Tesis Doctoral. Estas conclusiones se extraen de todo el trabajo de investigación realizado en esta tesis sobre la detección de la propagación del malware en las redes IoT. Las principales aportaciones realizadas en esta Tesis Doctoral son las siguientes:

- Se ha realizado un análisis de los problemas concretos de detección e identificiación del malware en redes IoT detectados en el estado del arte, abordando las deficiencias existentes relacionadas con la ciberseguridad de la redes IoT. Esto ha servido para establecer el conjunto de requisitos iniciales para el diseño y desarrollo de la solución propuesta.
- Se han estudiado las técnicas y tecnologías utilizadas o con una posible aplicación en el estudio de la propagación del malware que han permitido el diseño de una solución atendiendo a la mejora de la seguridad en las redes IoT. Estos requisitos han llevado al desarrollo de una nueva técnica para descubrir una red IoT parcial o totalmente desconocida en la que se tiene un punto de acceso y a la detección de los nodos infectados por malware de dicha red IoT.
- Se han analizado las propuestas de la literatura sobre la identificación del malware que se propaga por una red IoT. A partir de este analisis, se han propuestos dos metodologias para identificar el tipo de malware que se propaga por una red IoT en los casos en los que el malware sea conocido usando aprendizaje supervisado y en el caso de que el malware sea desconocido resolviendo el problema inverso para estimar los parámetros del modelo teórico que mas se asemeje al malware presente en la red IoT.
- Se ha demostrado el buen rendimiento de las metodologías propuesta y de los algoritmos diseñados mediante su implementación y evaluación en escenarios realistas. Esta implementación ha sido probada para la propagación de distintos tipos de malware mediante simulación en diferentes escenarios y mediante experimentación en casos reales. Los resultados obtenidos en los casos de estudio diseñados demuestran la efectividad de la propuesta en la detección e identificación del malware en las redes IoT.

A modo de conclusión, cabe señalar que esta Tesis Doctoral ha alcanzado los objetivos iniciales que se le plantearon al principio:

- Se han propuesto nuevos métodos para optimizar el descubrimiento de las redes IoT cuya información sea parcial o totalmente nula teniendo un punto de acceso.

- Se ha diseñado un nuevo algoritmo para aumentar la fiabilidad de la detección de los nodos infectados por malware en una red IoT.
- Se ha diseñado una nueva metodología para identificar malwares conocidos que se estén propagando por una red IoT a partir de una base de datos de información y aprendizaje supervisado.
- Se ha desarrolado una nueva metología para la identificacíon del modelo teórico de epidemiología matemática más similar a los malwares desconocidos que se propagen en las redes IoT.
- Se han realizado simulaciones realistas y experimentos empíricos para verificar la eficacia de las técnicas propuestas y demostrar la mejora de la seguridad en las redes IoT.

El objetivo principal de esta tesis doctoral es diseñar algoritmos y metodologías que permitan descubrir la topología de las redes IoT, identificar los nodos infectados por malware e identificar el malware que se propaga en dichas redes. Como se ha discutido en esta sección, este objetivo inicial se ha alcanzado con éxito.

6.3 Contribuciones al estado del arte

El trabajo de investigación presentado en esta Tesis Doctoral proporciona algunas nuevas aportaciones en los campos del descubrimiento de las topologías desconocidas de las redes IoT, así como, en la identificación de los nodos infectados en las redes IoT y el malware que los esta infectando:

- Desde el punto de vista de la ingeniería, se ha utilizado una metodología orientada al desarrollo de algoritmos para cubrir el descubrimiento e identificación de los malwares que se propagan en las redes IoT.
- Desde el punto de vista del desarrollo, la validez de este tipo de modelos podría verificarse en un entorno real, que consiste en la optimización de la detección e identificación temprana del malware que se propaga por una red IoT. Además, como se ha utilizado aprendizaje supervisado y por refuerzo en el diseño las nuevas metodologías y algoritmos, las soluciones son facilmente escalables y personalizables para las diferentes casuísticas que puedan aparecer.
- Los resultados obtenidos han sido analizados y estudiados para demostrar que los modelos propuestos son una solución viable para optimizar la monitorización e identificación del malware que se propaga en las redes IoT.

- Se ha realizado un importante trabajo para obtener información de distintos investigadores y grupos de investigación en áreas relacionadas. El objetivo ha sido fortalecer esta investigación mediante el intercambio mutuo de ideas y conocimientos. Se ha puesto especial interés en difundir nuestras experiencias y avances en esta investigación, desde sus etapas iniciales hasta su forma final, a través de publicaciones, asistencia a congresos, estancias y participación en sesiones especiales.

Además, en el Apéndice A figuran las publicaciones científicas y otros trabajos derivados de las contribuciones mencionadas.

6.4 Trabajo futuro

La investigación presentada en esta tesis doctoral valida las metodologías y algoritmos propuestos para mejorar la capacidad de descubrimiento e identificación del malware en las redes IoT. Estos resultados son solo un punto de partida en un ambito en clara expansión como son las redes IoT y en concreto su seguridad y ciberseguridad. En este sentido, la linea de investigación principal para el trabajo futuro será la mejora u optimización de los algoritmos propuestos con técnicas novedosas de inteligencia artificial y modelización matemática. Esto nos permitirá mejorar la detección e identificación del malware en las redes IoT, lo que supondra un gran avance en la seguridad y ciberseguridad de las redes IoT. Finalmente, se quiere profundizar en la complejidad y heterogeneidad de las redes IoT y sus dispositivos, lo que nos va a permitir acercar los modelos teóricos a escenarios reales.

A partir de los resultados obtenidos en este trabajo, se pueden concretar otras posibles líneas de trabajo en los siguientes aspectos:

- Mejorar la Complejidad del Algoritmo: Se quiere investigar la evaluación del algoritmo introduciendo nuevas restricciones y retos durante el proceso de descubrimiento de la topología de la red IoT. Además, se quiere probar la validez del algoritmo propuesto utilizando modelos de propagación mas sofisticados para una detección mas precisa y efectiva de los nodos infectos.

- Diversidad de Dispositivos y Estructuras de Red: Se valorará aplicar los algoritmos propuestos en redes IoT de mayor complejidad, incluyendo una diversidad más amplia de dispositivos y estructuras de la red IoT. Esto nos permitirá evaluar si las diferentes topologías influyen en el comportamiento de los algoritmos propuestos.

- Mínimo Número de Agentes para Detección Óptima: Se quiere investigar el número de agentes mínimo necesarios para que la detección de los nodos IoT infectados sea óptima dependiendo del tipo de malware identificado en la red IoT.

- Momento de Detección del Malware: Queremos explorar cómo afecta el momento de la detección del malware en la efectividad de los algoritmos propuestos. Esto nos va a permitir desarrollar estrategias de detección mas eficientes y adaptativas.

- Funciones de Recompensa Más Complejas: Se explorará la introducción de funciones de recompensa más complejas (desde un punto de vista matemático) para modelar y simular escenarios más realistas. Es enfoque mejorará la representación d ela realidad en las pruebas y simulaciones y nos permitirá una comprensión más profunda del comportamiento dle malware en las redes IoT más realistas.

Bibliografía

Abbasi, M., Shahraki, A., and Taherkordi, A. (2021). Deep learning for network traffic monitoring and analysis (ntma): A survey. *Computer Communications*, 170:19–41.

Abdullah, D. M. and Abdulazeez, A. M. (2021). Machine learning applications based on svm classification a review. *Qubahan Academic Journal*, 1(2):81–90.

Abijaude, J., Sobreira, P., Pinto, I., and Greve, F. (2021). Combining rest and snmp for http traffic optimization: A case study on gourmet cocoa drying. In *2021 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE.

Abusitta, A., Li, M. Q., and Fung, B. C. (2021). Malware classification and composition analysis: A survey of recent developments. *Journal of Information Security and Applications*, 59:102828.

Adler, S. and Subramanian, V. (2024). Bayesian learning of optimal policies in markov decision processes with countably infinite state-space. *Advances in Neural Information Processing Systems*, 36:3979–4033.

Agarwal, N., Chaudhuri, S., Jain, P., Nagaraj, D., and Netrapalli, P. (2021). Online target q-learning with reverse experience replay: Efficiently finding the optimal policy for linear mdps. *arXiv preprint arXiv:2110.08440*.

Ahmad, I., Niazy, M. S., Ziar, R. A., and Khan, S. (2021). Survey on iot: security threats and applications. *Journal of Robotics and Control*, 2(1):42–46.

Ahmad, M. A. (2019). The v-network: a testbed for malware analysis. *Science World Journal*, 14(3):70–76.

Ahmed, M., Afreen, N., Ahmed, M., Sameer, M., and Ahamed, J. (2023). An inception v3 approach for malware classification using machine learning and transfer learning. *International Journal of Intelligent Networks*, 4:11–18.

Al-Turjman, F., Abujubbeh, M., Malekloo, A., and Mostarda, L. (2020). Uavs assessment in software-defined iot networks: An overview. *Computer Communications*, 150:519–536.

Alasadi, S. A. and Bhaya, W. S. (2017). Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107.

Alenezi, M. N., Alabdulrazzaq, H., Alshaher, A. A., and Alkharang, M. M. (2020). Evolution of malware threats and techniques: A review. *International Journal of Communication Networks and Information Security*, 12(3):326–337.

Alhaidari, F. A. and Alqahtani, E. J. (2020). Securing communication between fog computing and iot using constrained application protocol (coap): A survey. *Journal of Communications*, 15(1):14–30.

Ali, I., Ahmed, A. I. A., Almogren, A., Raza, M. A., Shah, S. A., Khan, A., and Gani, A. (2020a). Systematic literature review on iot-based botnet attack. *IEEE Access*, 8:212220–212232.

Ali, I., Ahmedy, I., Gani, A., Talha, M., Raza, M. A., and Anisi, M. H. (2020b). Data collection in sensor-cloud: A systematic literature review. *IEEE Access*, 8:184664–184687.

AliAhmad, A., Eleyan, D., Eleyan, A., Bejaoui, T., Zolkipli, M. F., and Al-Khalidi, M. (2023). Malware detection issues, future trends and challenges: a survey. In *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE.

Alrubayyi, H., Goteng, G., and Jaber, M. (2023). Ais for malware detection in a realistic iot system: Challenges and opportunities. *Network*, 3(4):522–537.

Alrubayyi, H., Goteng, G., Jaber, M., and Kelly, J. (2021). Challenges of malware detection in the iot and a review of artificial immune system approaches. *Journal of Sensor and Actuator Networks*, 10(4):61.

Aman, M. N., Javaid, U., and Sikdar, B. (2021). Iot-proctor: a secure and lightweight device patching framework for mitigating malware spread in iot networks. *IEEE Systems Journal*, 16(3):3468–3479.

Amodu, O. A., Bukar, U. A., Mahmood, R. A. R., Jarray, C., and Othman, M. (2023). Age of information minimization in uav-aided data collection for wsn and iot applications: A systematic review. *Journal of Network and Computer Applications*, 216:103652.

Andrade, J. and Duggan, J. (2020). An evaluation of hamiltonian monte carlo performance to calibrate age-structured compartmental seir models to incidence data. *Epidemics*, 33:100415.

Ang, K. L. M., Seng, J. K. P., and Ngharamike, E. (2022). Towards crowdsourcing internet of things (crowd-iot): Architectures, security and applications. *Future Internet*, 14(2):49.

Ashraf, N., Sheikh, S. A., Khan, S. A., Shayea, I., and Jalal, M. (2021). Simultaneous wireless information and power transfer with cooperative relaying for next-generation wireless networks: A review. *IEEE Access*, 9:71482–71504.

Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A., and Akin, E. (2023). A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics*, 12(6):1333.

Aslan, Ö., Ozkan-Okay, M., and Gupta, D. (2021). A review of cloud-based malware detection system: Opportunities, advances and challenges. *European Journal of Engineering and Technology Research*, 6(3):1–8.

Aslan, Ö. A. and Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access*, 8:6249–6271.

Azab, A., Khasawneh, M., Alrabaee, S., Choo, K.-K. R., and Sarsour, M. (2022). Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks*, In press.

Balaji, S., Nathani, K., and Santhakumar, R. (2019). Iot technology, applications and challenges: a contemporary survey. *Wireless Personal Communications*, 108:363–388.

Bansal, U. et al. (2021). A review on ransomware attack. In *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pages 221–226. IEEE.

Bar-Yam, Y. (2019). *Dynamics of complex systems*. CRC Press.

Barlas, Y. (1996). Formal aspects of model validity and validation in system dynamics. *System Dynamics Review: The Journal of the System Dynamics Society*, 12(3):183–210.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43.

Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., and Khan, M. K. (2021). Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & Security*, 111:102490.

Bella, G., Biondi, P., Bognanni, S., and Esposito, S. (2023). Petiot: Penetration testing the internet of things. *Internet of Things*, 22:100707.

Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

Ben-Shlomo, Y., Mishra, G. D., and Kuh, D. (2023). Life course epidemiology. *Handbook of epidemiology*, pages 1–31.

Berkhahn, S. and Ehrhardt, M. (2022). A physics-informed neural network to model covid-19 infection and hospitalization scenarios. *Advances in Continuous and Discrete Models*, 2022(1):61.

Bhowmik, R. and Riaz, M. H. (2023). An extended review of the application layer messaging protocol of the internet of things. *Bulletin of Electrical Engineering and Informatics*, 12(5):3124–3133.

Bodkhe, U. and Tanwar, S. (2020). Taxonomy of secure data dissemination techniques for iot environment. *IET Software*, 14(6):563–571.

Brandt, N., Ahlemann, F., Reining, S., and Rehring, K. (2022). Internet of things in product lifecycle management–a review on value creation through product status data. *zur Erlangung des Doktorgrades Dr. rer. pol. der Fakultät für Wirtschaftswissenschaften der Universität Duisburg-Essen*, page 22.

Brauer, F., Van den Driessche, P., Wu, J., and Allen, L. J. (2008). *Mathematical epidemiology*, volume 1945. Springer.

Cantó, B., Coll, C., and Sánchez, E. (2017). Estimation of parameters in a structured sir model. *Advances in Difference Equations*, 2017:1–13.

Canto, J., Dacier, M., Kirda, E., and Leita, C. (2008). Large scale malware collection: lessons learned. In *IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*, page 6.

Carvajal, R. G. L. and Inlago, M. (2021). The spyware: Preparación de contribuciones. *Innovation & Development in Engineering and Applied Sciences*, 3(2):8.

Caviglione, L., Choraś, M., Corona, I., Janicki, A., Mazurczyk, W., Pawlicki, M., and Wasielewska, K. (2020). Tight arms race: Overview of current malware threats and trends in their detection. *IEEE Access*, 9:5371–5396.

Çelebi, M., Özbilen, A., and Yavanoğlu, U. (2023). A comprehensive survey on deep packet inspection for advanced network traffic analysis: issues and challenges. *Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi*, 12(1):1–29.

Çetinkaya, Z. and Horasan, F. (2021). Decision trees in large data sets. *International Journal of Engineering Research and Development*, 13(1):140–151.

Chakraborty, C. and Rodrigues, J. J. (2020). A comprehensive review on device-to-device communication paradigm: trends, challenges and applications. *Wireless Personal Communications*, 114(1):185–207.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., and Murphy, K. (2022). Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research*, 23(89):1–64.

Chanal, P. M. and Kakkasageri, M. S. (2020). Security and privacy in iot: a survey. *Wireless Personal Communications*, 115(2):1667–1693.

Chataut, R., Phoummalayvane, A., and Akl, R. (2023). Unleashing the power of iot: A comprehensive review of iot applications and future prospects in healthcare, agriculture, smart homes, smart cities, and industry 4.0. *Sensors*, 23(16):7194.

Chaudhary, D. K., Yadav, P., Gupta, S., and Jha, K. (2022). Iot network feature based intrusion detection techniques-review. In *2022 IEEE International Conference on Current Development in Engineering and Technology (CCET)*, pages 1–5. IEEE.

Clifton, J. and Laber, E. (2020). Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, 7:279–301.

Clim, A., Toma, A., Zota, R. D., and Constantinescu, R. (2022). The need for cybersecurity in industrial revolution and smart cities. *Sensors*, 23(1):120.

da Silva, W. B., Dutra, J. C., Knupp, D. C., Abreu, L. A., and Silva Neto, A. J. (2020). Estimation of timewise varying boundary heat flux via bayesian filters and markov chain monte carlo method. *Computational Intelligence in Emerging Technologies for Engineering Applications*, pages 137–153.

Dalmau, R. and Allard, E. (2020). Air traffic control using message passing neural networks and multi-agent reinforcement learning. *Proceedings of the 10th SESAR Innovation Days, Virtual Event*, pages 7–10.

Dangerfield, B. and Duggan, J. (2020). Optimization of system dynamics models. *System Dynamics: Theory and Applications*, pages 139–152.

De Fazio, R., Giannoccaro, N. I., Carrasco, M., Velazquez, R., and Visconti, P. (2021). Wearable devices and iot applications for symptom detection, infection tracking, and diffusion containment of the covid-19 pandemic: A survey. *Frontiers of Information Technology & Electronic Engineering*, 22(11):1413.

Debas, E., Alhumam, N., and Riad, K. (2023). Unveiling the dynamic landscape of malware sandboxing: A comprehensive review. *Preprints*.

del Rey, A. M. (2015). Mathematical modeling of the propagation of malware: a review. *Security and Communication Networks*, 8(15):2561–2579.

del Rey, A. M., Vara, R. C., and González, S. R. (2022). A computational propagation model for malware based on the sir classic model. *Neurocomputing*, 484:161–171.

Diekmann, O. and Heesterbeek, J. A. P. (2000). *Mathematical epidemiology of infectious diseases: model building, analysis and interpretation*, volume 5. John Wiley & Sons.

Docquier, T., Song, Y.-Q., Chevrier, V., Pontnau, L., and Ahmed-Nacer, A. (2023). Performance evaluation methodologies for smart grid substation communication networks: a survey. *Computer Communications*, 198:228–246.

Doyu, H., Morabito, R., and Brachmann, M. (2021). A tinymlaas ecosystem for machine learning in iot: Overview and research challenges. In *2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–5. IEEE.

Driggers, J. C., Vitale, S., Lundgren, A., Evans, M., Kawabe, K., Dwyer, S., Izumi, K., Schofield, R., Effler, A., Sigg, D., et al. (2019). Improving astrophysical parameter estimation via offline noise subtraction for advanced ligo. *Physical Review D*, 99(4):042001.

Durmus, A., Moulines, É., and Pereyra, M. (2022). A proximal markov chain monte carlo method for bayesian inference in imaging inverse problems: When langevin meets moreau. *SIAM Review*, 64(4):991–1028.

Ebad, S. A., Darem, A. A., and Abawajy, J. H. (2021). Measuring software obfuscation quality–a systematic literature review. *IEEE Access*, 9:99024–99038.

ElSawy, H., Kishk, M. A., and Alouini, M.-S. (2020). Spatial firewalls: Quarantining malware epidemics in large-scale massive wireless networks. *IEEE Communications Magazine*, 58(9):32–38.

Eppe, M., Gumbsch, C., Kerzel, M., Nguyen, P. D., Butz, M. V., and Wermter, S. (2022). Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nature Machine Intelligence*, 4(1):11–20.

Eschmann, J. (2021). Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, pages 25–33.

Farooq, M. S., Riaz, S., Abid, A., Umer, T., and Zikria, Y. B. (2020). Role of iot technology in agriculture: A systematic literature review. *Electronics*, 9(2):319.

Farooq, U. and Lokam, A. (2023). A compact 26/39 ghz millimeter wave mimo antenna design for 5g iot applications. *Journal of Infrared, Millimeter, and Terahertz Waves*, 44(5):333–345.

Faruk, M. J. H., Shahriar, H., Valero, M., Barsha, F. L., Sobhan, S., Khan, M. A., Whitman, M., Cuzzocrea, A., Lo, D., Rahman, A., et al. (2021). Malware detection and prevention using artificial intelligence techniques. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5369–5377. IEEE.

Feriani, A. and Hossain, E. (2021). Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 23(2):1226–1252.

Ferrag, M. A., Maglaras, L., Ahmim, A., Derdour, M., and Janicke, H. (2020). Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks. *Future Internet*, 12(3):44.

Ferrández, M. R., Ivorra, B., Redondo, J. L., Ramos, Á. M., and Ortigosa, P. M. (2023). A multi-objective approach to identify parameters of compartmental epidemiological models—application to ebola virus disease epidemics. *Communications in Nonlinear Science and Numerical Simulation*, 120:107165.

Ferreira, G. O., Ravazzi, C., Dabbene, F., Calafiore, G. C., and Fiore, M. (2023). Forecasting network traffic: A survey and tutorial with open-source comparative evaluation. *IEEE Access*, 11:6018–6044.

Figueira, A. and Vaz, B. (2022). Survey on synthetic data generation, evaluation methods and gans. *Mathematics*, 10(15):2733.

Fizza, K., Banerjee, A., Mitra, K., Jayaraman, P. P., Ranjan, R., Patel, P., and Georgakopoulos, D. (2021). Qoe in iot: a vision, survey and future directions. *Discover Internet of Things*, 1:1–14.

Furtado, P. (2021). Epidemiology sir with regression, arima, and prophet in forecasting covid-19. *Engineering Proceedings*, 5(1):52.

Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: a systematic review. *Archives of computational methods in engineering*, 29(5):2531–2561.

Gadewadikar, J. and Marshall, J. (2024). A methodology for parameter estimation in system dynamics models using artificial intelligence. *Systems Engineering*, 27(2):253–266.

Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47.

Garcia, S., Parmisano, A., and Erquiaga, M. J. (2020). Iot-23: A labeled dataset with malicious and benign iot network traffic. *Stratosphere Lab., Praha, Czech Republic, Tech. Rep.*

Gaurav, A., Gupta, B. B., and Panigrahi, P. K. (2023). A comprehensive survey on machine learning approaches for malware detection in iot-based enterprise information system. *Enterprise Information Systems*, 17(3):2023764.

Gopal, S., Poongodi, C., Nanthiya, D., Priya, R. S., Saran, G., and Priya, M. S. (2021). Mitigating dos attacks in iot using supervised and unsupervised algorithms–a survey. In *IOP Conference Series: Materials Science and Engineering*, volume 1055, page 012072. IOP Publishing.

Grimm, V., Heinlein, A., Klawonn, A., Lanser, M., and Weber, J. (2022). Estimating the time-dependent contact rate of sir and seir models in mathematical epidemiology using physics-informed neural networks. *Electronic Transactions on Numerical Analysis*, 56:1–27.

Guillén, J. H. and Del Rey, A. M. (2018). Modeling malware propagation using a carrier compartment. *Communications in Nonlinear Science and Numerical Simulation*, 56:217–226.

Gulatas, I., Kilinc, H. H., Zaim, A. H., and Aydin, M. A. (2023). Malware threat on edge/fog computing environments from internet of things devices perspective. *IEEE Access*, 11:33584–33606.

Gulati, K., Boddu, R. S. K., Kapila, D., Bangare, S. L., Chandnani, N., and Saravanan, G. (2022). A review paper on wireless sensor network techniques in internet of things (iot). *Materials Today: Proceedings*, 51:161–165.

Habib, N. (2019). *Hands-on Q-learning with python: Practical Q-learning with openai gym, Keras, and tensorflow*. Packt Publishing Ltd.

Hattaf, K., Yousfi, N., and Tridane, A. (2012). Mathematical analysis of a virus dynamics model with general incidence rate and cure rate. *Nonlinear Analysis: Real World Applications*, 13(4):1866–1872.

Hoang, M. T. and Ehrhardt, M. (2024). Differential equation models for infectious diseases: Mathematical modeling, qualitative analysis, numerical methods and applications.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hossein Motlagh, N., Mohammadrezaei, M., Hunt, J., and Zakeri, B. (2020). Internet of things (iot) and the energy sector. *Energies*, 13(2):494.

Hussain, F., Hussain, R., Hassan, S. A., and Hossain, E. (2020). Machine learning in iot security: Current solutions and future challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1686–1721.

Idrissi, I., Azizi, M., and Moussaoui, O. (2020). Iot security with deep learning-based intrusion detection systems: A systematic literature review. In *2020 Fourth international conference on intelligent computing in data sciences (ICDS)*, pages 1–10. IEEE.

Imteaj, A., Thakker, U., Wang, S., Li, J., and Amini, M. H. (2021). A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24.

Iqbal, M., Abdullah, A. Y. M., and Shabnam, F. (2020). An application based comparative study of lpwan technologies for iot environment. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 1857–1860. IEEE.

Jabraeil Jamali, M. A., Bahrami, B., Heidari, A., Allahverdizadeh, P., Norouzi, F., Jabraeil Jamali, M. A., Bahrami, B., Heidari, A., Allahverdizadeh, P., and Norouzi, F. (2020). Iot architecture. *Towards the Internet of Things: Architectures, Security, and Applications*, pages 9–31.

Jang, B., Kim, M., Harerimana, G., and Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7:133653–133667.

Jeon, J., Park, J. H., and Jeong, Y.-S. (2020). Dynamic analysis for iot malware detection with convolution neural network model. *IEEE Access*, 8:96899–96911.

Jiang, X., Wang, D., Chen, X., and Zhang, M. (2022). Physics-informed neural network for optical fiber parameter estimation from the nonlinear schrödinger equation. *Journal of Lightwave Technology*, 40(21):7095–7105.

Jihong, Z., Han, Z., Chuang, W., Lu, Z., Shangqin, Y., and Zhang, W. (2021). A review of topology optimization for additive manufacturing: Status and challenges. *Chinese Journal of Aeronautics*, 34(1):91–110.

Jin, Y.-F., Yin, Z.-Y., Zhou, W.-H., and Horpibulsuk, S. (2019). Identifying parameters of advanced soil models using an enhanced transitional markov chain monte carlo method. *Acta Geotechnica*, 14:1925–1947.

Jouhari, M., Saeed, N., Alouini, M.-S., and Amhoud, E. M. (2023). A survey on scalable lorawan for massive iot: Recent advances, potentials, and challenges. *IEEE Communications Surveys Tutorials*, 25(3):1841–1876.

Kabanda, G., Chipfumbu, C. T., and Chingoriwo, T. (2023). Utilizing deep reinforcement learning and q-learning algorithms for improved ethereum cybersecurity. *International Journal of Advanced Networking and Applications*, 14(6):5742–5753.

Kadri, M. R., Abdelli, A., Mokdad, L., et al. (2023). Survey and classification of dos and ddos attack detection and validation approaches for iot environments. *Internet of Things*, page 101021.

Kadrich, M. (2007). *Endpoint security*. Addison-Wesley Professional.

Kalachev, L., Landguth, E. L., and Graham, J. (2023). Revisiting classical sir modelling in light of the covid-19 pandemic. *Infectious Disease Modelling*, 8(1):72–83.

Kalsoom, T., Ramzan, N., Ahmed, S., and Ur-Rehman, M. (2020). Advances in sensor technologies in the era of smart factory and industry 4.0. *Sensors*, 20(23):6783.

Kanaan, M. and Farrington, C. (2005). Matrix models for childhood infections: a bayesian approach with applications to rubella and mumps. *Epidemiology & Infection*, 133(6):1009–1021.

Kaniadakis, G., Baldi, M. M., Deisboeck, T. S., Grisolia, G., Hristopulos, D. T., Scarfone, A. M., Sparavigna, A., Wada, T., and Lucia, U. (2020). The κ-statistics approach to epidemiology. *Scientific Reports*, 10(1):19949.

Kara, I. (2023). Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges. *Expert Systems with Applications*, 214:119133.

Karie, N. M., Sahri, N. M., Yang, W., Valli, C., and Kebande, V. R. (2021). A review of security standards and frameworks for iot-based smart environments. *IEEE Access*, 9:121975–121995.

Kassim, M. R. M. (2020). Iot applications in smart agriculture: Issues and challenges. In *2020 IEEE conference on open systems (ICOS)*, pages 19–24. IEEE.

Kearns, M., Mansour, Y., and Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49:193–208.

Kermack, W. O. and McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721.

Ketu, S. and Mishra, P. K. (2022). Cloud, fog and mist computing in iot: an indication of emerging opportunities. *IETE Technical Review*, 39(3):713–724.

Khalid, L. F. and Ameen, S. Y. (2021). Secure iot integration in daily lives: A review. *Journal of Information Technology and Informatics*, 1(1):6–12.

Khan, J. A. and Chowdhury, M. M. (2021). Security analysis of 5g network. In *2021 IEEE International Conference on Electro Information Technology (EIT)*, pages 001–006. IEEE.

Khanna, A. and Kaur, S. (2020). Internet of things (iot), applications and challenges: a comprehensive review. *Wireless Personal Communications*, 114:1687–1762.

Khatua, P. K., Ramachandaramurthy, V. K., Kasinathan, P., Yong, J. Y., Pasupuleti, J., and Rajagopalan, A. (2020). Application and assessment of internet of things toward the sustainability of energy systems: Challenges and issues. *Sustainable Cities and Society*, 53:101957.

Khuphiran, P., Leelaprute, P., Uthayopas, P., Ichikawa, K., and Watanakeesuntorn, W. (2018). Performance comparison of machine learning models for ddos attacks detection. In *2018 22nd international computer science and engineering conference (ICSEC)*, pages 1–4. IEEE.

Kim, D.-Y., Kim, S., and Park, J. H. (2017). Remote software update in trusted connection of long range iot networking integrated with mobile edge cloud. *IEEE Access*, 6:66831–66840.

Kim, M. (2019). Supervised learning-based ddos attacks detection: Tuning hyperparameters. *ETRI Journal*, 41(5):560–573.

Kim, Y. G., Mendoza, B., Kwon, O., and Yoon, J. (2022). Task-specific feature selection and detection algorithms for iot-based networks. *Journal of Computer and Communications*, 10(10):59–73.

Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36.

Krishna, V., Suri, N. R., and Athithan, G. (2011). A comparative survey of algorithms for frequent subgraph discovery. *Current Science*, pages 190–198.

Krishnamurthi, R., Kumar, A., Gopinathan, D., Nayyar, A., and Qureshi, B. (2020). An overview of iot sensor data processing, fusion, and analysis techniques. *Sensors*, 20(21):6076.

Kumari, P. and Jain, A. K. (2023). A comprehensive study of ddos attacks over iot network and their countermeasures. *Computers & Security*, 127:103096.

Kwok, K. O., Tang, A., Wei, V. W., Park, W. H., Yeoh, E. K., and Riley, S. (2019). Epidemic models of contact tracing: systematic review of transmission studies of severe acute respiratory syndrome and middle east respiratory syndrome. *Computational and Structural Biotechnology Journal*, 17:186–194.

Ladosz, P., Weng, L., Kim, M., and Oh, H. (2022). Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22.

Laghari, A. A., Wu, K., Laghari, R. A., Ali, M., and Khan, A. A. (2021). A review and state of art of internet of things (iot). *Archives of Computational Methods in Engineering*, pages 1–19.

Lai, C. S., Jia, Y., Dong, Z., Wang, D., Tao, Y., Lai, Q. H., Wong, R. T., Zobaa, A. F., Wu, R., and Lai, L. L. (2020). A review of technical standards for smart cities. *Clean Technologies*, 2(3):290–310.

Lavate, S. H. and Srivastava, P. (2023). An analytical review on classification of iot traffic and channel allocation using machine learning technique. In *2023 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pages 1–7. IEEE.

Lazebnik, T. (2023). Computational applications of extended sir models: A review focused on airborne pandemics. *Ecological Modelling*, 483:110422.

Le, N., Rathour, V. S., Yamazaki, K., Luu, K., and Savvides, M. (2022). Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, pages 1–87.

Lee, G., Kim, W., Oh, H., Youn, B. D., and Kim, N. H. (2019). Review of statistical model calibration and validation—from the perspective of uncertainty structures. *Structural and Multidisciplinary Optimization*, 60:1619–1644.

Liu, Y., Wang, H., Cai, L., Shen, X., and Zhao, R. (2021). Fundamentals and advancements of topology discovery in underwater acoustic sensor networks: A review. *IEEE Sensors Journal*, 21(19):21159–21174.

Loktongbam, P., Pal, D., Bandyopadhyay, A., and Koley, C. (2023). A brief review on mm-wave antennas for 5g and beyond applications. *IETE Technical Review*, 40(3):397–422.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228.

Luengo, D., Martino, L., Bugallo, M., Elvira, V., and Särkkä, S. (2020). A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020:1–62.

Luo, F.-M., Xu, T., Lai, H., Chen, X.-H., Zhang, W., and Yu, Y. (2024). A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2):121101.

Lvovich, I. Y., Lvovich, Y. I., Preobrazhenskiy, A., Preobrazhenskiy, Y. P., and Choporov, O. (2020). Analysis of integral characteristics in the iot system. In *IOP Conference Series: Materials Science and Engineering*, volume 734, page 012020. IOP Publishing.

Madan, S., Sofat, S., and Bansal, D. (2022). Tools and techniques for collection and analysis of internet-of-things malware: A systematic state-of-art review. *Journal of King Saud University-Computer and Information Sciences*, 34(10):9867–9888.

Madhvan, R. and Zolkipli, M. F. (2023). An overview of malware injection attacks: Techniques, impacts, and countermeasures. *Borneo International Journal eISSN 2636-9826*, 6(3):22–30.

Mahboubi, A., Camtepe, S., and Ansari, K. (2020). Stochastic modeling of iot botnet spread: A short survey on mobile malware spread modeling. *IEEE Access*, 8:228818–228830.

Maiwada, U. D., Imran, S. A., Danyaro, K. U., Janisar, A. A., Salameh, A., and Sarlan, A. B. (2024). Security concerns of iot against ddos in 5g systems. *International Journal of Electrical Engineering and Computer Science*, 6:98–105.

Mane, S. Y. (2021). Lpwan's–overview, market scenario and performance analysis of lora, sigfox using nb-fi range calculator. In *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, pages 1–4. IEEE.

Marino, I. P., Zaikin, A., and Miguez, J. (2017). A comparison of monte carlo-based bayesian parameter estimation methods for stochastic models of genetic networks. *PloS One*, 12(8):e0182015.

Marinov, T. T., Marinova, R. S., Omojola, J., and Jackson, M. (2014). Inverse problem for coefficient identification in sir epidemic models. *Computers & Mathematics with Applications*, 67(12):2218–2227.

Martcheva, M. (2015). *An introduction to mathematical epidemiology*, volume 61. Springer.

Martín del Rey, A. (2023a). New trends on malware propagation: From iot environments to drone swarms. In *International Conference on Mathematics and its Applications in Science and Engineering*, pages 197–207. Springer.

Martín del Rey, A. (2023b). New trends on malware propagation: From iot environments to drone swarms. In *International Conference on Mathematics and its Applications in Science and Engineering*, pages 197–207. Springer.

Martín del Rey, Á. M. et al. (2024). Modelos matemáticos para la propagación de malware: estado del arte y algunas reflexiones.

Maswadi, K., Ghani, N. B. A., and Hamid, S. B. (2020). Systematic literature review of smart home monitoring technologies based on iot for the elderly. *IEEE Access*, 8:92244–92261.

Miao, C., Chen, G., Yan, C., and Wu, Y. (2021). Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Computers & Industrial Engineering*, 156:107230.

Miao, H., Xia, X., Perelson, A. S., and Wu, H. (2011). On identifiability of nonlinear ode models and applications in viral dynamics. *SIAM review*, 53(1):3–39.

Mingers, J. (2006). A critique of statistical modelling in management science from a critical realist perspective: its role within multimethodology. *Journal of the Operational Research Society*, 57:202–219.

Mittal, M., Kumar, K., and Behal, S. (2023). Deep learning approaches for detecting ddos attacks: A systematic review. *Soft Computing*, 27(18):13039–13075.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mohammadi, R., Akleylek, S., Ghaffari, A., and Shirmarz, A. (2022). Taxonomy of traffic engineering mechanisms in software-defined networks: a survey. *Telecommunication Systems*, 81(3):475–502.

Mohanta, B. K., Jena, D., Satapathy, U., and Patnaik, S. (2020). Survey on iot security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet of Things*, 11:100227.

Mondal, W. U., Agarwal, M., Aggarwal, V., and Ukkusuri, S. V. (2022). On the approximation of cooperative heterogeneous multi-agent reinforcement learning (marl) using mean field control (mfc). *Journal of Machine Learning Research*, 23(129):1–46.

Montesinos-López, O. A. and Hernández-Suárez, C. M. (2007). Modelos matemáticos para enfermedades infecciosas. *Salud Pública de México*, 49:218–226.

Munshi, A., Alqarni, N. A., and Almalki, N. A. (2020). Ddos attack on iot devices. In *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–5. IEEE.

Nadim, M., Lee, W., and Akopian, D. (2023). Kernel-level rootkit detection, prevention and behavior profiling: A taxonomy and survey. *arXiv preprint arXiv:2304.00473*.

Nappi, I. and de Campos Ribeiro, G. (2020). Internet of things technology applications in the workplace environment: A critical review. *Journal of Corporate Real Estate*, 22(1):71–90.

Naser, M., Albazar, H., and Abdel-Jaber, H. (2023). Mobile spyware identification and categorization: A systematic review. *Informatica*, 47(8).

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*, 4(51-62):56.

Ndjuluwa, L. N., Adebisi, J. A., and Dayoub, M. (2023). Internet of things for crop farming: A review of technologies and applications. *Commodities*, 2(4):367–381.

Nelson, T., Nance, C., and Noteboom, C. (2023). Web injection and banking trojan malware-a systematic literature review. In *2023 6th International Conference on Information and Computer Technologies (ICICT)*, pages 45–53. IEEE.

Ngo, Q.-D., Nguyen, H.-T., Le, V.-H., and Nguyen, D.-H. (2020). A survey of iot malware and detection methods based on static features. *ICT Express*, 6(4):280–286.

Nguyen, P. D. and Kim, L.-w. (2021). Sensor system: A survey of sensor type, ad hoc network topology and energy harvesting techniques. *Electronics*, 10(2).

Nguyen, Q. H., Ly, H.-B., Ho, L. S., Al-Ansari, N., Le, H. V., Tran, V. Q., Prakash, I., and Pham, B. T. (2021). Influence of data splitting on performance of machine learning models in prediction of shear strength of soil. *Mathematical Problems in Engineering*, 2021:1–15.

Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9):3826–3839.

Nguyen, X.-H. and Le, K.-H. (2023). Robust detection of unknown dos/ddos attacks in iot networks using a hybrid learning model. *Internet of Things*, 23:100851.

Nita, S. L. and Mihailescu, M. I. (2022). Fuel monitoring system based on iot: Overview and device authentication. In *2022 14th International Conference on Communications (COMM)*, pages 1–4. IEEE.

Nižetić, S., Šolić, P., Gonzalez-De, D. L.-d.-I., Patrono, L., et al. (2020). Internet of things (iot): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of Cleaner Production*, 274:122877.

Núñez, I., Cano, E., Rovetto, C., Ojo-Gonzalez, K., Smolarz, A., and Saldana-Barrios, J. J. (2022). Key technologies applied to the optimization of smart grid systems based on the internet of things: A review. In *2022 V Congreso Internacional en Inteligencia Ambiental, Ingeniería de Software y Salud Electrónica y Móvil (AmITIC)*, pages 1–8. IEEE.

Oroojlooy, A. and Hajinezhad, D. (2023). A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722.

Padakandla, S. (2021). A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25.

Padhye, V. and Lakshmanan, K. (2023). A deep actor critic reinforcement learning framework for learning to rank. *Neurocomputing*, 547:126314.

Pandit, A. V. and Mondal, D. (2023). Real-time malware detection on iot devices using behavior-based analysis and neural networks. *Research Journal of Computer Systems and Engineering*, 4(2):117–129.

Parisi, S., Tangkaratt, V., Peters, J., and Khan, M. E. (2019). Td-regularized actor-critic methods. *Machine Learning*, 108:1467–1501.

Piazzola, C., Tamellini, L., and Tempone, R. (2021). A note on tools for prediction under uncertainty and identifiability of sir-like dynamical systems for epidemiology. *Mathematical Biosciences*, 332:108514.

Pour, M. S., Bou-Harb, E., Varma, K., Neshenko, N., Pados, D. A., and Choo, K.-K. R. (2019). Comprehending the iot cyber threat landscape: A data dimensionality reduction technique to infer and characterize internet-scale iot probing campaigns. *Digital Investigation*, 28:S40–S49.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. (2023). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Rahim, M. A., Rahman, M. A., Rahman, M. M., Asyhari, A. T., Bhuiyan, M. Z. A., and Ramasamy, D. (2021). Evolution of iot-enabled connectivity and applications in automotive industry: A review. *Vehicular Communications*, 27:100285.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707.

Raju, V. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., and Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 729–735. IEEE.

Rana, A., Kumar, A., and Ali, H. (2021). An overview of the network topologies for enterprises. *Asian Journal of Multidimensional Research*, 10(10):143–149.

Razzaq, A. (2020). A systematic review on software architectures for iot systems and future direction to the adoption of microservices architecture. *SN Computer Science*, 1(6):350.

Reason, P. and Bradbury, H. (2001). *Handbook of action research: Participative inquiry and practice*. sage.

Reis, M. d. and Yang, Z. (2011). Approximate likelihood calculation on a phylogeny for bayesian estimation of divergence times. *Molecular Biology and Evolution*, 28(7):2161–2172.

Reis, W. P. N. d., Silva, G. J. d., Junior, O. M., and Vivaldini, K. C. T. (2021). An extended analysis on tuning the parameters of adaptive monte carlo localization ros package in an automated guided vehicle. *The International Journal of Advanced Manufacturing Technology*, 117:1975–1995.

Rmus, M., McDougle, S. D., and Collins, A. G. (2021). The role of executive function in shaping reinforcement learning. *Current Opinion in Behavioral Sciences*, 38:66–73.

Robert, C. P., Casella, G., and Casella, G. (2010). *Introducing monte carlo methods with r*, volume 18. Springer.

Sadhu, P. K., Yanambaka, V. P., and Abdelgawad, A. (2022). Internet of things: Security and solutions survey. *Sensors*, 22(19):7433.

Saito, S., Wenzhuo, Y., and Shanmugamani, R. (2018). *Python reinforcement learning projects: eight hands-on projects exploring reinforcement learning algorithms using TensorFlow*. Packt Publishing Ltd.

Salama, R., Altrjman, C., and Al-Turjman, F. (2023). An overview of the internet of things (iot) and machine to machine (m2m) communications. *NEU Journal for Artificial Intelligence and Internet of Things*, 2(3).

Salim, M. M., Elsayed, H. A., and Abdalzaher, M. S. (2023). A survey on essential challenges in relay-aided d2d communication for next-generation cellular networks. *Journal of Network and Computer Applications*, 216:103657.

Savage, L. J. (1972). *The foundations of statistics*. Courier Corporation.

Schiassi, E., De Florio, M., D'Ambrosio, A., Mortari, D., and Furfaro, R. (2021). Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models. *Mathematics*, 9(17):2069.

Shafiq, M., Gu, Z., Cheikhrouhou, O., Alhakami, W., and Hamam, H. (2022). The rise of "internet of things": review and open research issues related to detection and prevention of iot-based security attacks. *Wireless Communications and Mobile Computing*, 2022:1–12.

Shafiq, M., Tian, Z., Sun, Y., Du, X., and Guizani, M. (2020). Selection of effective machine learning algorithm and bot-iot attacks traffic identification for internet of things in smart city. *Future Generation Computer Systems*, 107:433–442.

Shafique, K., Khawaja, B. A., Sabir, F., Qazi, S., and Mustaqim, M. (2020). Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios. *IEEE Access*, 8:23022–23040.

Shah, T. and Venkatesan, S. (2018). Authentication of iot device and iot server using secure vaults. In *2018 17th IEEE international conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*, pages 819–824. IEEE.

Shah, V. (2020). Reinforcement learning for autonomous software agents: Recent advances and applications. *Revista Espanola de Documentacion Cientifica*, 14(1):56–71.

Shakeri, M., Sadeghi-Niaraki, A., Choi, S.-M., and Islam, S. R. (2020). Performance analysis of iot-based health and environment wsn deployment. *Sensors*, 20(20):5923.

Shakya, A. K., Pillai, G., and Chakrabarty, S. (2023). Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, page 120495.

Shandilya, S. K., Upadhyay, S., Kumar, A., and Nagar, A. K. (2022). Ai-assisted computer network operations testbed for nature-inspired cyber security based adaptive defense simulation and analysis. *Future Generation Computer Systems*, 127:297–308.

Sharma, R., Prakash, S., and Roy, P. (2020). Methodology, applications, and challenges of wsn-iot. In *2020 international conference on electrical and electronics engineering (ICE3)*, pages 502–507. IEEE.

Shaukat, K., Alam, T. M., Hameed, I. A., Khan, W. A., Abbas, N., and Luo, S. (2021). A review on security challenges in internet of things (iot). In *2021 26th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE.

Shen, Q. and Shen, Y. (2024). Endpoint security reinforcement via integrated zero-trust systems: A collaborative approach. *Computers & Security*, 136:103537.

Shetty, S. H., Shetty, S., Singh, C., and Rao, A. (2022). Supervised machine learning: algorithms and applications. *Fundamentals and methods of machine and deep learning: algorithms, tools and applications*, pages 1–16.

Shrivastava, A., Bhardwaj, A., and Hasteer, N. (2020). Iot in automobile sector: State of the art. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 254–259. IEEE.

Shukla, R. K. (2021). A review on network monitoring in software-defined networking. *Asian Journal of Multidimensional Research*, 10(11):334–340.

Sikimić, M., Amović, M., Vujović, V., Suknović, B., and Manjak, D. (2020). An overview of wireless technologies for iot network. In *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6. IEEE.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Singh, A. P., Luhach, A. K., Gao, X.-Z., Kumar, S., and Roy, D. S. (2020). Evolution of wireless sensor network design from technology centric to user centric: an architectural perspective. *International Journal of Distributed Sensor Networks*, 16(8):1550147720949138.

Singh, C. and Jain, A. K. (2024). A comprehensive survey on ddos attacks detection & mitigation in sdn-iot network. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, page 100543.

Singh, J. and Singh, J. (2021). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture*, 112:101861.

Sobin, C. (2020). A survey on architecture, protocols and challenges in iot. *Wireless Personal Communications*, 112(3):1383–1429.

Souri, A., Hussien, A., Hoseyninezhad, M., and Norouzi, M. (2022). A systematic review of iot communication strategies for an efficient smart environment. *Transactions on Emerging Telecommunications Technologies*, 33(3):e3736.

Sovacool, B. K. and Del Rio, D. D. F. (2020). Smart home technologies in europe: A critical review of concepts, benefits, risks and policies. *Renewable and sustainable energy reviews*, 120:109663.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Taddy, M. A., Lee, H. K., and Sansó, B. (2009). Fast inference for statistical inverse problems. *Inverse Problems*, 25(8):085001.

Talukder, S. and Talukder, Z. (2020). A survey on malware detection and analysis tools. *International Journal of Network Security & Its Applications (IJNSA) Vol*, 12.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337.

Tandale, K. D. and Pawar, S. N. (2020). Different types of phishing attacks and detection techniques: A review. In *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, pages 295–299. IEEE.

Tang, L., Zhou, Y., Wang, L., Purkayastha, S., Zhang, L., He, J., Wang, F., and Song, P. X.-K. (2020). A review of multi-compartment infectious disease models. *International Statistical Review*, 88(2):462–513.

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., and Asadpour, M. (2020). Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data*, 7:1–47.

Tao, W., Zhao, L., Wang, G., and Liang, R. (2021). Review of the internet of things communication technologies in smart agriculture and challenges. *Computers and Electronics in Agriculture*, 189:106352.

Tarantola, A. (2005). *Inverse problem theory and methods for model parameter estimation*. SIAM.

Tarekegn, A. N., Giacobini, M., and Michalak, K. (2021). A review of methods for imbalanced multi-label classification. *Pattern Recognition*, 118:107965.

Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G. D., and Barajas-Solano, D. (2020). Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems. *Water Resources Research*, 56(5):e2019WR026731.

Tayyab, U.-e.-H., Khan, F. B., Durad, M. H., Khan, A., and Lee, Y. S. (2022). A survey of the recent trends in deep learning based malware detection. *Journal of Cybersecurity and Privacy*, 2(4):800–829.

Tightiz, L. and Yang, H. (2020). A comprehensive review on iot protocols' features in smart grid communication. *Energies*, 13(11):2762.

Torres-Carrión, H., Solano-Chamba, C., Narváez-Guillen, C., and Cueva-Hurtado, M. (2022). Iot security issues in the context of edge computing: A systematic review of literature. In *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–7. IEEE.

Uchenna, C. C., Jamil, N., Ismail, R., Yan, L. K., and Mohamed, M. A. (2021). Malware threat analysis techniques and approaches for iot applications: A review. *Bulletin of Electrical Engineering and Informatics*, 10(3):1558–1571.

Uehara, M., Shi, C., and Kallus, N. (2022). A review of off-policy evaluation in reinforcement learning. *arXiv preprint arXiv:2212.06355*.

Uprety, A. and Rawat, D. B. (2020). Reinforcement learning for iot security: A comprehensive survey. *IEEE Internet of Things Journal*, 8(11):8693–8706.

Veloz, J., Alcivar, A., Salvatierra, G., and Silva, C. (2017). Ethical hacking, una metodología para descubrir fallas de seguridad en sistemas informáticos mediante la herramienta kali-linux. *Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones*, 1(1).

Vercio, L. L., Amador, K., Bannister, J. J., Crites, S., Gutierrez, A., MacDonald, M. E., Moore, J., Mouches, P., Rajashekar, D., Schimert, S., et al. (2020). Supervised machine learning tools: a tutorial for clinicians. *Journal of Neural Engineering*, 17(6):062001.

Victoire, T. A., Vasuki, M., Karunamurthy, A., Soundarya, D., and Sarumathi, S. (2023). A survey on cyber security threats and its impact on society. *International Journal of Research in Engineering, Science and Management*, 6(6):146–152.

Vidyasagar, M. (2020). Recent advances in reinforcement learning. In *2020 American Control Conference (ACC)*, pages 4751–4756. IEEE.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Vishwakarma, R. and Jain, A. K. (2020). A survey of ddos attacking techniques and defence mechanisms in the iot network. *Telecommunication Systems*, 73(1):3–25.

Vouros, G. A. (2022). Explainable deep reinforcement learning: state of the art and challenges. *ACM Computing Surveys*, 55(5):1–39.

Wang, H.-n., Liu, N., Zhang, Y.-y., Feng, D.-w., Huang, F., Li, D.-s., and Zhang, Y.-m. (2020). Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 21(12):1726–1744.

Wang, S., Balarezo, J. F., Chavez, K. G., Al-Hourani, A., Kandeepan, S., Asghar, M. R., and Russello, G. (2022). Detecting flooding ddos attacks in software defined networks using supervised learning techniques. *Engineering Science and Technology, an International Journal*, 35:101176.

Wang, X. and Su, H. (2020). Completely model-free rl-based consensus of continuous-time multi-agent systems. *Applied Mathematics and Computation*, 382:125312.

Wei, C.-Y., Jahromi, M. J., Luo, H., Sharma, H., and Jain, R. (2020a). Model-free reinforcement learning in infinite-horizon average-reward markov decision processes. In *International conference on machine learning*, pages 10170–10180. PMLR.

Wei, S., Bao, Y., and Li, H. (2020b). Optimal policy for structure maintenance: A deep reinforcement learning framework. *Structural Safety*, 83:101906.

Wickramasinghe, I. and Kalutarage, H. (2021). Naive bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation. *Soft Computing*, 25(3):2277–2293.

Winarno, A. and Affandi, M. (2022). Design and construction of smart house prototype based internet of things (iot) using esp8266. *BEST: Journal of Applied Electrical, Science, & Technology*, 4(1):11–14.

Wong, W. and Juwono, F. H. (2022). Estimating effective reproduction number for sir compartmental model: A stochastic evolutionary approach. *Journal of Social Computing*, 3(2):182–189.

Wongvorachan, T., He, S., and Bulut, O. (2023). A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information*, 14(1):54.

Wu, H., Han, H., Wang, X., and Sun, S. (2020). Research on artificial intelligence enhancing internet of things security: A survey. *IEEE Access*, 8:153826–153848.

Xie, J., Shao, Z., Li, Y., Guan, Y., and Tan, J. (2019). Deep reinforcement learning with optimized reward functions for robotic trajectory planning. *IEEE Access*, 7:105669–105679.

Yaacoub, J.-P. A., Noura, H. N., Salman, O., and Chehab, A. (2023). Ethical hacking for iot: Security issues, challenges, solutions and recommendations. *Internet of Things and Cyber-Physical Systems*, 3:280–308.

Yakupov, D. (2022). Overview and comparison of protocols internet of things: Mqtt and amqp. *International Journal of Open Information Technologies*, 10(9):90–98.

Yan, S., Ren, J., Wang, W., Sun, L., Zhang, W., and Yu, Q. (2022). A survey of adversarial attack and defense methods for malware classification in cyber security. *IEEE Communications Surveys & Tutorials*, 25(1):467–496.

Yang, K., Zhang, J., Xu, Y., and Chao, J. (2020). Ddos attacks detection with autoencoder. In *NOMS 2020-2020 IEEE/IFIP network operations and management symposium*, pages 1–9. IEEE.

Yang, L., Ng, B., Seah, W. K., Groves, L., and Singh, D. (2021). A survey on network forwarding in software-defined networking. *Journal of Network and Computer Applications*, 176:102947.

Yashoda, M. and Shivashetty, V. (2022). Bi-crs: Bio-inspired cluster-based routing scheme for d2d communication in iot. In *Proceedings of International Conference on Recent Trends in Computing: ICRTC 2021*, pages 187–199. Springer.

Yuan, Y., Yu, Z. L., Gu, Z., Yeboah, Y., Wei, W., Deng, X., Li, J., and Li, Y. (2019). A novel multi-step q-learning method to improve data efficiency for deep reinforcement learning. *Knowledge-Based Systems*, 175:107–117.

Zaman, G., Kang, Y. H., and Jung, I. H. (2008). Stability analysis and optimal vaccination of an sir epidemic model. *BioSystems*, 93(3):240–249.

Zang, M., Zheng, C., Dittmann, L., and Zilberman, N. (2023). Towards continuous threat defense: In-network traffic analysis for iot gateways. *IEEE Internet of Things Journal*.

Zang, W., Zhang, P., Zhou, C., and Guo, L. (2015). Locating multiple sources in social networks under the sir model: A divide-and-conquer approach. *Journal of Computational Science*, 10:278–287.

Zaza, A. M., Kharroub, S. K., and Abualsaud, K. (2020). Lightweight iot malware detection solution using cnn classification. In *2020 IEEE 3rd 5G World Forum (5GWF)*, pages 212–217. IEEE.

Zhaikhan, A., Kishk, M. A., ElSawy, H., and Alouini, M.-S. (2020). Safeguarding the iot from malware epidemics: A percolation theory approach. *IEEE Internet of Things Journal*, 8(7):6039–6052.

Zhang, X., Breitinger, F., Luechinger, E., and O'Shaughnessy, S. (2021). Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations. *Forensic Science International: Digital Investigation*, 39:301285.

Zhang, Y. and Zavlanos, M. M. (2024). Cooperative multiagent reinforcement learning with partial observations. *IEEE Transactions on Automatic Control*, 69(2):968–981.

Zhang, Z. and Wang, D. (2024). Adaptive individual q-learning—a multiagent reinforcement learning method for coordination optimization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12.

Zhao, S., Peng, Y., Zhang, Y., and Wang, H. (2022). Parameter estimation of power electronic converters with physics-informed machine learning. *IEEE Transactions on Power Electronics*, 37(10):11567–11578.

Zhou, Z., Liu, G., and Zhou, M. (2023). A robust mean-field actor-critic reinforcement learning against adversarial perturbations on agent states. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12.

Zhu, H., Rashidinejad, P., and Jiao, J. (2024). Importance weighted actor-critic for optimal conservative offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Zhu, J., Wu, F., and Zhao, J. (2021). An overview of the action space for deep reinforcement learning. In *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, pages 1–10.

Zikria, Y. B., Ali, R., Afzal, M. K., and Kim, S. W. (2021). Next-generation internet of things (iot): Opportunities, challenges, and solutions. *Sensors*, 21(4):1174.