MECInOT: Emulador de escenarios de Industrial Internet of Things y Multi-Access Edge Computing para su análisis de seguridad

Sergio Ruiz Villafranca
Instituto de Investigación en Informática de Albacete (i3a)
Universidad de Castilla-La Mancha
sergio.rvillafranca@uclm.es

José Roldán Gómez
Instituto de Investigación en Informática de Albacete (i3a)
Universidad de Castilla-La Mancha
Jose.Roldan@uclm.es

José Luis Martínez
Instituto de Investigación en Informática de Albacete (i3a)
Universidad de Castilla-La Mancha
joseluis.martinez@uclm.es

José Miguel Villalón Millán
Instituto de Investigación en Informática de Albacete (i3a)
Universidad de Castilla-La Mancha
josemiguel.villalon@uclm.es

***Resumen*—En los últimos años se está produciendo una gran revolución en el modo de gestionar los entornos industriales. El uso de nuevos dispositivos de la Internet de las Cosas (IoT) en estos entornos está produciendo lo que se conoce con el nombre de industria 4.0. Estos dispositivos IoT se caracterizan por su simplicidad y por su poca capacidad computacional. En estos escenarios, es de especial importancia solventar los nuevos problemas de seguridad que pueden aparecer, especialmente en aquellas infraestructuras consideradas críticas. Para facilitar esta tarea se presenta MECInOT, un emulador que permite a los investigadores generar diferentes escenarios de red sin la necesidad del gasto asociado al equipo industrial. Este emulador es completamente flexible, gracias a la virtualización de dispositivos, ajustándose a las necesidades que puedan surgir en este tipo de escenarios. A partir del emulador propuesto, se pueden desplegar y analizar medidas de seguridad para medir y evaluar su impacto.**

***Index Terms*—Industrial Internet of Things, Industria 4.0, Ciberseguridad, openLEON, Docker, MEC**

Tipo de contribución: *Investigación en ciberseguridad*

I. Introducción

Con el paso del tiempo se han ido produciendo sucesivas revoluciones industriales, con el objetivo de mejorar la productividad y viabilidad de las fábricas. Últimamente, el principal foco reside en la optimización de los recursos materiales y humanos gracias a los avances tecnológicos que están apareciendo. En este entorno, en la actualidad nos encontramos en la cuarta revolución industrial, también llamada industria 4.0 o IIoT (*Industrial Internet of Things*), que contempla ya los avances y las implementaciones de tecnologías IT, (*Information Technologies*). En esta cuarta revolución se considera la inclusión de las nuevas tecnologías como inteligencia artificial, *big data*, *cloud computing*, *edge computing*, virtualización, sistemas ciberfísicos, sensores IoT (*Internet of Things*), junto con los protocolos OT (*Operational Technologies)* que tradicionalmente se han establecido en este ámbito [1]. Incluso, ya existen autores [2] que están proponiendo soluciones a los problemas derivados de la implementación de la industria 4.0, abocando por la denominación de la industria 5.0. En esta futura industria se busca la personalización y el enfoque de las ventajas y funcionalidades de estas tecnologías en función de ámbito de producción de cada compañía.

El término de IIoT [3] nace al añadir dispositivos M2M (*machine to machine*) a Internet, incluyendo el uso de las tecnologías IoT en entornos industriales. Esto provoca que, gracias a la información que aportan estos dispositivos a las nuevas industrias, se genere una gran cantidad de datos que serán transportados por la red. Tal y como se menciona en [4], se espera que el 50 % de los dispositivos conectados a Internet en 2023 serán de este tipo. Este hecho hará que más de 14.7 mil millones de estos dispositivos estén conectados a Internet.

Por otro lado, una de las tecnologías que han revolucionado el mundo de la informática en los últimos años ha sido el *Cloud Computing* [5]. Su uso ha servido a muchas empresas como una opción económica de obtener una capacidad de cómputo moldeable en función de las necesidades que puedan surgir en momentos concretos. Sin embargo, con el incremento del número de dispositivos conectados, se está comenzando a apreciar una reducción en el rendimiento de la red de los principales proveedores de Cloud (Amazon Web Services, Azure Cloud, Google Cloud). Esto esta provocando que aparezcan retardos en las comunicaciones, haciendo que ciertas funcionalidades con restricciones temporales severas tengan problemas a la hora de cumplir con sus prestaciones, produciéndose latencias superiores a las deseadas [6]. La solución propuesta para esta situación ha sido la definición e implementación de *Edge & Fog Computing* [5], lo que nos permite acercar el procesamiento de la nube a las redes empresariales. Esto a su vez, abre la oportunidad de la aparición de nuevas aplicaciones que eran inviables en *Cloud Computing*, como todas aquellas asociadas a aplicaciones en tiempo real y con restricciones temporales muy exigentes, como la conducción autónoma o a la gestión y manejo de la información recibida por los drones no tripulados. Por otra parte, *Edge Computing* también ofrece la posibilidad de mejorar la gestión de un tráfico heterogéneo y distribuido, como el que se produce con el IIoT [7].

A pesar de las grandes aportaciones y ventajas que ofrecen

estas tecnologías, esta nueva situación provoca la aparición de nuevas vías de entrada y vulnerabilidades que pueden ser aprovechadas por los atacantes. Antes de la aparición del IIoT, las industrias se encontraban más aisladas de Internet, por lo que el acceso malintencionado a sus infraestructuras críticas era más difícil. Por otro lado, si la adopción e implementación de estas nuevas tecnologías no se realiza siguiendo una serie de buenas prácticas y con especial enfoque en evitar la aparición de riesgos en los sistemas, se pueden dar incidencias de seguridad sin que el personal de la empresa este preparado para poder afrontar esta situación [3]. En este punto, existe una fuerte demanda de herramientas que permitan emular en entornos controlados los procesos industriales, en una linea a lo que introduce el gemelo digital para entornos industriales. A pesar de las limitaciones que existen en la virtualizacion de entornos industriales, estas herramientas pueden servir para la mejora de los protocolos de red usados en dichos entornos.

Con esta premisa se presenta MECInOT, un emulador que está basado en openLEON [8], y que se enfoca en el despliegue de diferentes escenarios industriales caracterizados por el uso simulado de dispositivos IIoT. Con este trabajo se busca proporcionar una herramienta de virtualizacion de la capa de red IIoT. También, se quiere otorgar la capacidad de modificar e introducir nuevos dispositivos y servicios al emulador de los ya existentes en él. Incluso, la incorporación de la comunicación con *hardware* real, con el fin de dar soporte al paradigma de gemelo digital [9]. Además, se incluye un conjunto de herramientas en MECInOT, para realizar pruebas de seguridad en los entornos diseñados. Con el fin, de dar soporte a las investigaciones realizadas en el ámbito de la ciberseguridad en los entornos IIoT y MEC, y comprobar el impacto de los diferentes modelos de amenaza presentes en ellos.

II. Fundamentos Técnicos

En este apartado profundizaremos en algunas partes teóricas de las arquitecturas que desplegará el emulador, y en posibles riesgos y problemas que encontramos en los entornos IIoT.

II-A. Multi-Access Edge Computing (MEC)

MEC es una evolución del concepto de *Edge Computing*, que fue estandarizado por el *European Telecommunications Standards Institute* (ETSI), y que busca mejorar el rendimiento de las comunicaciones entre el entorno de *Cloud Computing* y las tecnologías relacionadas con IoT e IIoT. Uno de los puntos más destacados de MEC es el uso de la virtualización para permitir la computación de diferentes aplicaciones y la gestión del tráfico heterogéneo que tiene que soportar [10]. Como ya hacía *Edge Computing*, al acercar la computación que tradicionalmente se realizaba en los entorno *Cloud*, podemos reducir la latencia de las comunicaciones y evitar posibles saturaciones en la red. Esto se traduce en una mejor experiencia para los usuarios finales de las aplicaciones, permitiendo el desarrollo de aplicaciones que en base a las limitaciones de las tecnologías que había hasta ahora era inviable. Además, el uso de las nuevas redes móviles 5G y las futuras 6G [6], permitirá reducir aún más las latencias entre los dispositivos, lo cual posibilitará que las aplicaciones en tiempo real puedan desplegarse en este tipo de arquitecturas.

Para poder implementar MEC, debemos de hacer uso de varias tecnologías de virtualización sobre las que se asienta este nuevo paradigma [11]. Estas tecnologías son:

- ***Network Function Virtualization* (NFV) [12].** Como se ha mencionado previamente, el uso del IIoT va a producir que una gran cantidad de dispositivos IoT estén conectados en entornos industriales. Muchos de estos dispositivos van a hacer uso de protocolos de comunicación propietarios, ya que son diseñados para entornos muy concretos. Esto obligaría a utilizar dispositivos de red específicos para poder gestionar este tráfico. Gracias a NFV, se puede gestionar todo este tráfico heterogéneo de manera automatizada, reduciendo los costes que conlleva el uso de dispositivos especializados, respecto a dispositivos generales. En esencia, NFV se trata de un software que puede ser instalado en cualquier dispositivo físico que permita la comunicación con él. ETSI especifica algunas aplicaciones que se podrían abordar con NFV, como las funciones de conectividad *Dynamic Host Configuration Protocol* (DHCP) o *Network Address Translation* (NAT), funciones de seguridad con *firewalls*, y gestión del tráfico procedente de redes móviles.
- ***Software Defined Networking* (SDN).** Tradicionalmente, en las redes de comunicaciones la capa de datos se correspondía con la parte de la red que se ocupaba de gestionar el tráfico generado por los usuarios. Por otro lado, la capa de control se ocupaba de la gestión de aquellas operaciones de enrutamiento. Ambas capas trabajaban conjuntamente en los propios dispositivos de red, provocando que estos dispositivos se ralentizaran. Con el incremento del número de dispositivos conectados en cada red, se comprobó que esto era una opción no escalable y poco flexible. En este punto se comenzó a utilizar *scripts* para la reconfiguración de las redes, produciéndose muchos errores de configuración[13]. Con SDN se busca solventar estos problemas separando ambas capas. El objetivo principal es no predefinir de manera tan clara la red, y poder monitorizar y gestionar la red sin la necesidad de regirse bajo unos dispositivos de red concretos o propietarios. Para ello, se usa una arquitectura estructurada en base a la definición de la red, y el reparto de los recursos vía software gracias a la virtualización de redes [14].
- ***Network Slicing* [10].** Esta técnica consiste en definir diferentes redes virtuales o capas en función del criterio que se considere, y a cada una de ellas se destinará una cantidad de recursos diferentes según sus necesidades. Esto permite, generar múltiples redes personalizables partiendo de la red física desplegada, aportando flexibilidad a la hora de repartir los recursos de red según la demanda de cada capa.
- ***Service Function Chaining* (SFC).** El cambio de las redes tradicionales a las redes definidas por software y virtualizadas supone un esfuerzo para los equipos IT de las empresas [11]. Por ello, el objetivo de SFC es facilitar esta transición de manera dinámica. Realmente, SFC es un concepto muy parecido a NFV. La principal diferencia entre ambos radica en que SFC soluciona los problemas que pueden surgir de proveer un servicio que

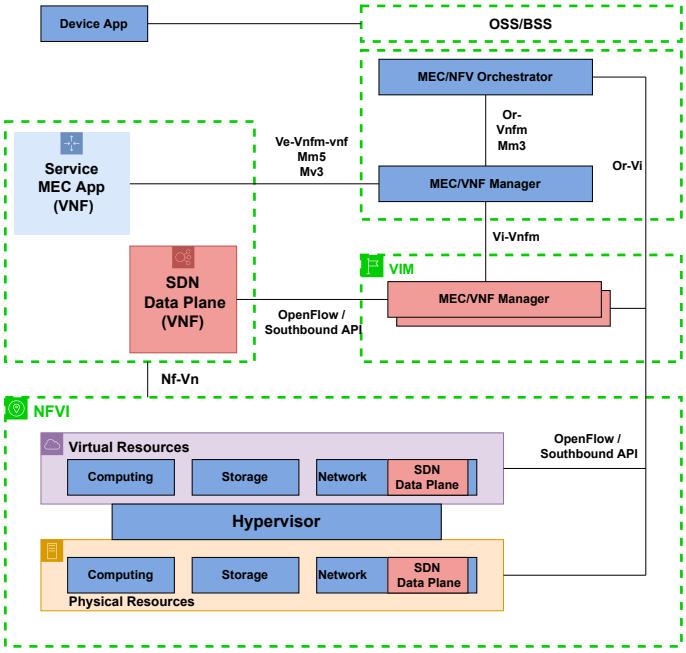

Figura 1. Arquitectura de referencia de MEC [10]

se encuentra dentro de una cadena de servicios en un dispositivo. Por tanto, SFC se ocupa de gestionar que servicios se están ejecutando en el dispositivo de red y cuáles pueden ser usados [14].

En la Figura 1 se puede observar un ejemplo de implementación de una arquitectura MEC con los servicios básicos de NFV y SDN que hemos visto. En ella se puede observar la comunicación entre el controlador de SDN y el orquestador NFV.

III. Estado del Arte

Existen algunos trabajos interesantes orientados a la emulación en entornos IIoT. En esta sección se realiza un breve estudio de los más destacados.

El emulador IIoT Testbed [15] permite la emulación de aplicaciones IIoT basadas en el *middleware Data Distribution Service* (DDS), permitiendo a los usuarios la gestión de procesos, en los cuales se pueden modificar múltiples parámetros, como por ejemplo *Quality of Service* (QoS). La gestión y creación de nuevos procesos IIoT se realiza a través de una interfaz gráfica web, facilitando su uso. Sin embargo, se trata de un emulador muy limitado en cuanto a funcionalidad y flexibilidad, ya que se encuentra centrado en la gestión de los datos de los procesos. Por esto, no es posible usarlo en nuestro entorno al no permitir introducir nuevos procesos y nuevas aplicaciones con diferentes protocolos.

Fogify [16] establece un entorno de desarrollo y despliegue de aplicaciones basadas en *Fog Computing* o MEC. Este emulador permite la creación y definición de redes, el despliegue de múltiples nodos *Edge* y la implementación de aplicaciones IoT. Además, también permite el análisis de prestaciones de los nodos desplegados y de la propia red, para comprobar diferentes fallos o rendimientos de las aplicaciones. Por contra, para poder usar Fogify se exige un alto dominio tanto de la herramienta, como de la base sobre la que se fundamenta su despliegue: *Docker Swarm*. Por otro lado, esta herramienta pueda quedar sin soporte en el corto plazo. Por último, se encuentran dificultades a la hora de desarrollar aplicaciones basadas en protocolos OT, imposibilitando la creación de un entorno IIoT heterogéneo.

Fogbed [17] utiliza conjuntamente las herramientas Containernet [18] y Maxinet [19], que derivan de Mininet, para permitir a los usuarios diseñar y desplegar las topologías que deseen. Al basarse en el lenguaje de programación Python, con un simple *script* es posible desplegar una topología cuyos nodos finales serán contenedores, lo que permite una gran flexibilidad a la hora de desplegar diferentes aplicaciones y servicios. El principal problema que encontramos con Fogbed, es que no se incluye la conectividad móvil que es tan característica en los entornos MEC. Además, al usar la versión por defecto de Containernet no se permite la creación de redundancia en la topología MEC. Esto puede generar posibles fallos de conectividad a la hora de incrementar la complejidad de los despliegues y aplicaciones.

El último emulador de este tipo de redes encontrado es openLEON [8]. Este emulador solventa los problemas que se mencionan de Fogbed, al implementar un controlador que permite el uso de *spanning-tree* en la topología, permitiendo la redundancia en la topología. Además, implementa en la topología desplegada con Containernet el emulador srsLTE [20] que junto con el *hardware* apropiado permite la conectividad LTE con toda la topología. Por esto, se considera openLEON como uno de los emuladores más apropiados para ser usado en el despliegue de entornos IIoT. Sin embargo, openLEON está centrado en el despliegue de la parte MEC, y no cuenta con servicios ni aplicaciones IIoT. Esto hace que no se puedan realizar experimentaciones y pruebas de seguridad en entornos de estas características. Por tanto, en este trabajo se considera usar openLEON como base de nuestro emulador para el despliegue de la topología MEC. El emulador propuesto, MECInOT, integra esta parte, junto con el despliegue de las topologías industriales con sus correspondientes dispositivos y servicios. Esto supone un aumento de la funcionalidad que no podemos encontrar en el resto de trabajos mencionados.

IV. Descripción de MECInOT

A lo largo de esta sección se detallan los aspectos de diseño y de metodología que se han tenido en cuenta a la hora de elaborar este emulador.

IV-A. Diseño

A la hora de plantear el diseño sobre el que se desarrolla MECInOT se han considerado varios puntos que deben ser abordados:

- **Emulación realista de entornos físicos.** Permitiendo realizar pruebas y obtener datos de una manera análoga a realizarlas sobre una topología real, con la ventaja de no necesitar los dispositivos reales para llevarlas a cabo.
- **Flexibilidad a la hora de elaborar diferentes escenarios en función de las necesidades de las investigaciones.** Gracias al uso de la virtualización de dispositivos mediante contenedores, somos capaces de modificar el número de dispositivos, su funcionamiento, o incluso la realización de múltiples subredes destinadas a una función específica.

- **Facilidad en la inclusión de dispositivos reales en los escenarios de prueba.** En función de la naturaleza de la experimentación, es posible que sea interesante realizar una evaluación de prestaciones en las que se incluyan dispositivos reales. En esta evaluación se pueden obtener métricas como colisiones, retrasos en los paquetes, rendimientos, etc. Es por ello, que gracias a la virtualización sobre la que se basa el emulador, es posible incluir dentro de las redes que despliega dispositivos físicos de red, como *IoT gateway*.
- **Enfoque para pruebas en ciberseguridad.** El diseño de este emulador se ha centrado en posibilitar la realización de pruebas de ciberseguridad en entornos IIoT. Para ello, se proporciona una serie de *scripts* y un contenedor atacante que permite a los usuarios, de forma sencilla, lanzar diferentes tipos de ataques, y comprobar su impacto en el escenario a estudio.

IV-B. Metodología de despliegue del emulador

En un escenario real de industria 4.0, es necesario el despliegue de cada dispositivo real en la red industrial o empresarial, y proporcionar una configuración de red que cubre las necesidades especificas del escenario. En el caso del despliegue de MECInOT, el primer paso es la virtualización de la máquina encargada de la creación de la red interna industrial, y de la máquina que se ocupará del despliegue de la arquitectura MEC. A continuación, se debe definir la comunicación entre la red interna industrial, y la topología MEC. Para que ésta sea posible, tanto la máquina real, como las máquinas virtuales que gestionan la red IIoT y la arquitectura MEC deben tener una dirección IP perteneciente a la misma subred, configurando el adaptador de red de las máquinas virtuales en modo puente. Además, es necesario la realización de una configuración de *routing* dentro de la máquina virtual que gestiona la red industrial, que nos permita comunicar ambas subredes.

A continuación, se deben establecer las direcciones de subred utilizadas para la red IIoT y la que contiene la arquitectura MEC. En la configuración del emulador se ha establecido la dirección 172.19.0.0/16 para la red industrial. En ella se encontrarán los dispositivos IIoT virtualizados en contenedores con la herramienta Docker-Compose que nos facilitará el rápido despliegue y modificación de los escenarios. En el caso de la red usada para el despliegue de la arquitectura MEC, se ha dejado la que ofrece openLEON en su configuración base, que es la dirección privada 10.0.0.0/12. Se comprueba que existe conexión entre los *hosts* de la red IIoT, y los de la red MEC.

El último paso consistirá en el despliegue de una arquitectura IIoT sobre la que se desea realizar las pruebas. Esta arquitectura, estará compuesta por un conjunto de dispositivos IIoT, ya implementados en los escenarios de prueba que proporciona el emulador, y por las topologías de redes industriales que serán descritas en el siguiente apartado.

IV-C. Topología industrial

Con la finalidad de poder emular una red de una industria 4.0, en MECInOT se ha implementado una red empresarial única haciendo uso de la interfaz que crea Docker-Compose para levantar los diferentes contenedores. Por otro lado, también se permite definir diferentes subredes para cada uno de los protocolos de comunicación que podemos encontrar en un escenario industrial, y que se van a definir a continuación:

Internet of Things. En el emulador se han incluido aplicaciones con los protocolos más utilizados en el ámbito del IoT: MQTT (*MQ Telemetry Transport*), CoAP (*Constrained Application Protocol*) y AMQP (*Advanced Message Queuing Protocol*) [21], [22].

Operational Technologies. Los protocolos que se han agregado al emulador son aquellas versiones industriales que emplean el protocolo de transporte TCP: Modbus/TCP, que es el estándar industrial más utilizado, S7 que es un protocolo de comunicación propietario usado por los PLC (*Programmable Logic Controller*) de Siemens, y OPC UA (*Open Platform Communications United Architecture*), el cual está considerado como uno de los protocolos que permiten una convergencia entre protocolos IT y OT [22].

Information Technologies. En este grupo de protocolos ubicamos aquellos que son los más utilizados por los usuarios de la red de Internet, como pueden ser el el HTTP (*Hypertext Transfer Protocol*) [23].

Como se ha comento previamente, nuestro emulador está diseñado para una evaluación de seguridad de redes IIoT. Es por ello, que debemos incluir un conjunto de herramientas que permitan de forma sencilla ejecutar ataques en entornos industriales, y evaluar su impacto. En este punto se ha incluido un nodo que utiliza como base la distribución Kali Linux para usarla en el análisis de seguridad.

De cara a evaluar el impacto de un posible ataque a la infraestructura o a los datos de ésta, se ha considera incluir los siguientes tipos de ataques:

- **Ataque de manipulación de paquetes.** Para este caso, el nodo atacante debe realizar un ataque *Man in The Middle* para capturar los paquetes a modificar. Se modificarán los datos incluidos en los paquetes transmitidos en la red IIoT. El resto de campos que componen los paquetes no son modificados.
- **Ataque de fuerza bruta.** Se ha implementado un *script* que lleva a cabo un ataque de diccionario sobre un login de un servidor web.
- ***Payload* en trama HTTP.** Con el objetivo de tratar de aprovecharnos de una vulnerabilidad tipo *Shellshock* [24] de un servidor web, se ha automatizado con un *script* el lanzamiento de una trama HTTP modificada con un *payload* en el campo *User-Agent*.
- **Escaneo de red.** Se ha preparado un *script* con el que se puede realizar de manera automatizada y sucesiva un escaneo sobre los dispositivos de la red. Este *script* de ataque también permite el lanzamiento de diferentes tipos de escaneos en una misma ejecución.
- **Ataques de denegación de servicio.** Se han implementado métodos de denegación de servicio tradicionales como los *ping of the dead* [25], y los basados en inundación o saturación de los puertos con el lanzamiento masivo de tramas TCP. Además, se han adaptados estos ataques en función del protocolo objetivo. En concreto, para el protocolo AMQP, se realiza la inclusión de dispositivos falsos con el fin de saturar la cola de mensajes, y que

un dispositivo legitimo no llegue a establecer comunicación. Por otro lado, aprovechando que CoAP se trata de un protocolo que utiliza UDP como protocolo de capa de transporte, se ha implementado un ataque de amplificación de UDP, utilizando uno de los métodos del servidor CoAP para denegar el servicio a algunos de los dispositivos de la red. Este último ataque consiste en utilizar un paquete UDP muy ligero, y usurpar la IP del dispositivos que se quiera atacar. De este modo podemos aprovecharnos del funcionamiento del protocolo UDP para que envíe un paquete mayor del enviado por el atacante.

- **Ataques de escaneo de dispositivos.** Este ataque trata de aprovecharse de una configuración básica de un *broker* de MQTT para obtener información adicional del mismo. Esta tarea se encuentra automatizada con un *script*, con lo que es posible obtener información de los dispositivos y los *topics* de comunicación.

IV-D. Topologia MEC

Como ya se ha comentado anteriormente, la topología MEC es desplegada gracias al emulador openLEON. En él, se puede diferenciar una red de centro de datos, y otra con conectividad móvil:

- **Centro de datos.** openLEON implementa esta parte de la topología con el emulador Containernet [18], que es una expansión del emulador Mininet [26], permitiendo la creación de topologías cuyos *hosts* están implementados mediante contenedores. Por otro lado, el esquema de la topología está basado en un centro de datos de *3-Tier*, lo cual hace que nos encontremos una topología de red jerárquica de 3 niveles. En dicha topología aparecen 2 *core switches*, y 2 *switches* de agregación por cada uno de ellos. Además, la arquitectura se compone de un total de 64 hosts conectados entre 8 switches que se encuentran en el nivel *Top of Rack*. Esta estructura ofrece redundancia de conexiones entre los distintos elementos con el fin de evitar puntos de fallo único en la red. Por ello, es de interés la implementación del protocolo *spanning-tree* en los dispositivos de red. Debido a que el controlador SDN por defecto de Mininet no lo soporta, los desarrolladores de openLEON decidieron utilizar el controlador RYU, que es un módulo del lenguaje de programación Python que soporta el protocolo OpenFlow para su implementación en la topología.
- **Comunicación móvil.** La parte que distingue a openLEON de otros emuladores estudiados es el soporte en las comunicaciones móviles como LTE. OpenLEON implementa esta parte a través del emulador srsLTE [20], y necesita para su funcionamiento de una serie de dispositivos hardware reales, como antenas y estaciones LTE. Con este módulo se permite la comunicación entre los dispositivos conectado vía LTE con los nodos del *Edge*.

En la Figura 2 se encuentran representadas todas las partes que conforman el emulador. Se puede observar las diferentes funcionalidades que aporta cada una de ellas al conjunto general de MECInOT.

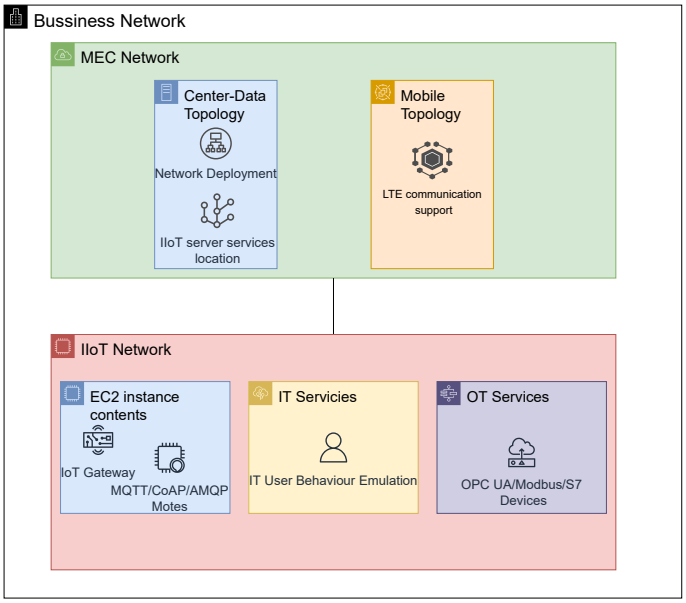

Figura 2. Diagrama lógico del emulador MECInOT.

V. Evaluación del escenario

En esta sección se muestran algunos ejemplos de escenarios que pueden ser desplegados en MECInOT, explicando su funcionamiento. Por otro lado, se evalúa el consumo de recursos de la máquina donde corre el emulador sobre los escenarios estudiados. Por último, se comprueba la viabilidad del emulador para la realización de pruebas orientadas a la ciberseguridad en entorno IIoT desplegado.

V-A. Hardware utilizado

Para la ejecución del emulador propuesto se ha utilizado un portátil con una CPU Intel i7-10875H a 2.30GHz con 32GB de RAM, teniendo instalado Windows 10 20H2. Por otro lado se ha seleccionado el hipervisor VirtualBox 6.1.16r. Además, se utilizará como hardware adicional una Raspberry Pi Zero 2 W que hará la función de *IoT gateway* en aquellos escenarios que sea necesarios.

V-B. Escenario industrial OT

En este escenario encontraremos exclusivamente los protocolos industriales que ya han sido mencionados anteriormente. A continuación se describe el reparto de los nodos entre las topologías y se describirá su funcionamiento.

El escenario de estudio del **protocolo OPC UA** está compuesto por un nodo cliente que se encuentra en la red industrial, y un nodo servidor alojado en la topología MEC. El funcionamiento básico de la comunicación entre ambos nodos será la lectura de una cadena de caracteres aleatoria que genera el servidor en un tiempo comprendido entre 1 a 9 segundos.

Para el caso del **protocolo S7** se despliegan dos nodos clientes, uno de lectura y otro de escritura, que se localizarán en la red industrial, y el nodo servidor alojado en la topología MEC. El funcionamiento es muy similar al del protocolo OPC UA. El cliente escritor modifica el valor de una cadena alojada en el servidor, y esta es leída por el segundo cliente.

Para el caso del **protocolo ModBus/TCP** se despliegan dos clientes localizados en la red industrial, con las mismas funciones que ya se han mencionado para el protocolo S7, y un servidor alojado en la topología MEC. En este escenario uno de los clientes escribe una lista de valores con valor verdadero o falso de manera alterna en el servidor, y cada cierto tiempo el segundo cliente lee estos valores almacenados.

Además, se despliega un nodo atacante en la red junto con el resto de dispositivos. Este nodo, descrito en la sección previa, cuenta con herramientas de escaneo y de aprovechamiento de vulnerabilidades. Además posee una serie de *scripts* enfocados para la realización de ataques.

V-C. Escenario fábrica 4.0

Se trata de una expansión del escenario anterior en el que se introduce la comunicación entre dispositivos IoT y de usuarios IT. Esto permite emular el compartimiento que se puede encontrar en una fábrica 4.0 y que se encuentra perfectamente conectada a diferentes servicios de red. En el caso de los protocolos IoT se diferenciará su comportamiento, pues su tráfico va dirigido a un *IoT gateway* localizado en la red empresarial. Al igual que en el escenario anterior, también se ha introducido un nodo que emula a un posible atacante en la red.

Con el **protocolo MQTT**, se ha tratado emular que la compañía tiene una serie de sensores de temperatura en diferentes puntos de la red industrial y que emitirá la información de temperatura en grados Celsius cada 3 segundos. La función del *IoT gateway* es simplemente implementar el *MQTT Broker* para establecer la comunicación con un subscriptor que se encuentra en uno de los nodos *Edge* de la topología MEC.

En el caso del **protocolo CoAP**, disponemos de un cliente localizado en la red industrial que escribe mensajes en el *IoT gateway* y que serán leídos por otro cliente alojado en la topología MEC.

Para el **protocolo AMQP**, se despliega un cliente que publica en la cola RabbitMQP alojada en el *IoT gateway* un número aleatorio entre el 33 y el 126 y que será recibido por un cliente alojado en la topología MEC.

Por parte de los **protocolos IT**, se han incluido una serie de servidores HTTP y HTTPS, siendo uno de ellos un login habitual de usuarios y un servidor *streaming* de contenido multimedia, tratando de simular la obtención de imágenes de un vídeo. Por parte de los clientes, que se encuentran exclusivamente en la red industrial, se han creado unos *scripts* que periódicamente consultarán y realizarán peticiones a los distintos servidores.

Un esquema lógico de la comunicación entre las partes de la topología descrita se puede ver en la Figura 3. Donde se puede diferenciar como se destina cada tipo de tráfico por los diferentes elementos que se han incluido en este escenario.

V-D. Estudio de rendimiento

Una vez descritos algunos de los escenarios que se pueden desplegar con el emulador, se va a realizar un estudio del consumo de recursos que conlleva el despliegue del escenario más complejo, que corresponde al escenario de una fábrica 4.0, debido a la mayor cantidad de contenedores que se deben desplegar y la cantidad de servicios y aplicaciones que se utilizan. La métrica que vamos a utilizar es el porcentaje de

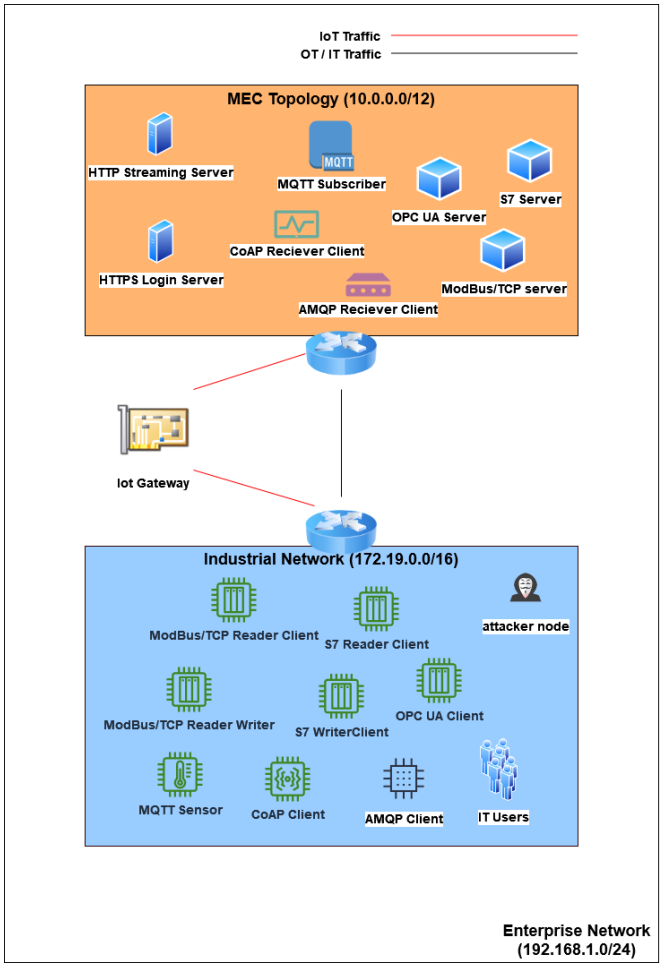

Figura 3. Esquema de red del escenario de una fábrica 4.0.

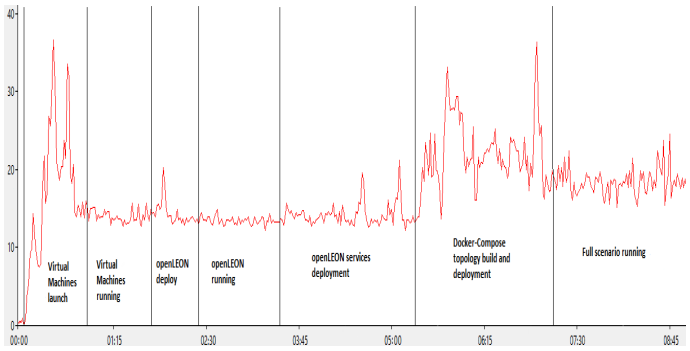

Figura 4. Evolución del consumo de CPU con el despliegue del escenario completo.

uso total del procesador de la máquina anfitriona con Windows 10 y utilizaremos la herramienta de Monitor de Rendimiento que proporciona el mismo sistema operativo para la obtención de los datos.

En la figura 4, se puede observar la evolución del consumo de CPU a lo largo de los 10 minutos que dura la ejecución del escenario. Podemos observar desde el coste que supone el levantamiento de las máquinas virtuales que contienen las dos partes del emulador hasta el impacto que tiene la ejecución en el sistema. También se aprecian los aumentos de carga según se despliegan las diferentes partes de los emuladores, destacando el pico de consumo cuando se hace la construcción

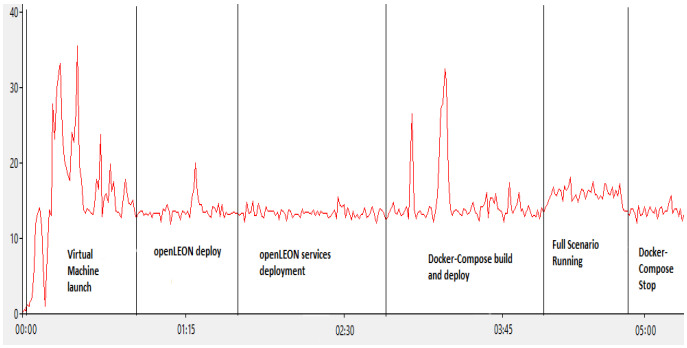

Figura 5. Evolución del consumo de CPU con el despliegue del escenario solo con protocolos OT.

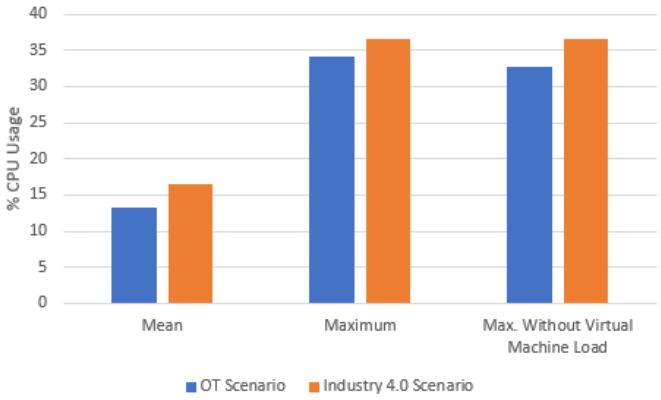

Figura 6. Gráfica comparativa del uso de CPU entre ambos escenarios.

de las imágenes de los contenedores y su despliegue.

Para comprobar la flexibilidad y como escalaría el consumo de recursos en función de un escenario con mayor complejidad, se realiza una comparativa con el despliegue del escenario que únicamente tiene en cuenta las aplicaciones relacionadas con los protocolos OT. Lo que implica que se despliega un escenario más simple que el que se utilizó para el primer experimento. La Figura 5 muestra la evolución de la ejecución de este único escenario durante una ejecución de 5 minutos, en la que también se muestra el impacto del lanzamiento de ataques a través del nodo atacante y que se ha reflejado en dicha figura.

Tal y como se muestra en la Figura 6, podemos comprobar cómo según la complejidad del escenario los recursos necesarios son mayores. Sin embargo, teniendo en cuenta el uso medio de CPU en ambos casos, podemos ver que la diferencia es únicamente del alrededor del 2 %, por lo que sería posible seguir incrementando el número de dispositivos virtualizados y el número de aplicaciones, hasta llegar a alcanzar altas tasas de utilización del procesador de la máquina. Otro apunte que podemos observar es que el pico de mayor utilización del procesador se produce durante el despliegue de las máquinas virtuales. Por la parte, en el escenario de industria 4.0 se observa que el mayor pico de consumo no viene dado por el despliegue de las maquinas virtuales, sino que viene asociado por la construcción y despliegue de las imágenes y sus correspondientes contenedores. Este experimento demuestra la viabilidad de desplegar los diferentes escenarios en un sistema con unos recursos limitados.

```
s7_client_writer_1       | [*] Message send: pxCbpppfKvyNaoTqbIsD
s7_client_writer_1       | [*] Message send: rGAIJVhgSZbnikwoLTlJ
s7_client_writer_1       | [*] Message send: VrDqAiCARQCdxAJrHray
s7_client_writer_1       | [*] Message send: aQjhdJixzSRZeKBHDSsL
s7_client_writer_1       | [*] Message send: wMeCoFEtJfMaVupHZxVv
```

Figura 7. Información enviada por el cliente escritor.

```
s7_client_reader_1       | 53435566844169441025
s7_client_reader_1       | 77410953794301182351
s7_client_reader_1       | 55736648159120147726
s7_client_reader_1       | 20972574760450636806
s7_client_reader_1       | 23851989352500968793
```

Figura 8. Datos recibidos por el cliente lector.

V-E. Análisis de Seguridad

Tras analizar el rendimiento del emulador, se va a realizar una prueba de concepto enfocada a la realización de un ciberataques, en concreto de un intento de modificación de paquetes del escenario de OT.

Primero se debe de acceder al nodo atacante del escenario y utilizar la herramienta *arpspoof* para poder capturar el tráfico que envíe o reciba el nodo que se desea, en esta prueba nuestro objetivo va a ser el cliente lector del protocolo S7.

Posteriormente, debemos de lanzar el *script* de manipulación de paquetes, que se ocupará de modificar los datos de los paquetes que lee el cliente del servidor. En la Figura 7 y Figura 8 se muestra como el cliente escritores envía cadenas de caracteres, mientras que el lector recibe del servidor números enteros, esto implica que nuestro ataque de manipulación está surgiendo efecto y estamos obteniendo el comportamiento esperado.

VI. Conclusiones

Este artículo se presenta el emulador MECInOT para de redes IIoT con soporte MEC; se presenta su funcionalidad, estructura y su potencial uso en la emulación de entornos industriales y, sobre todo, para el análisis de seguridad de este tipo de infraestructuras. Se han presentado las partes y potenciales usos de la herramienta propuesta y, se ha evaluado desde un punto de vista de aplicación y como herramienta para realizar auditorías de seguridad. Se ha demostrado que el emulador es escalable y permite emular topologías realmente complejas en ordenadores personales, gracias al uso de la tecnología de contenedores y virtualización. Finalmente, se ha presentado como se pueden desplegar ataques o intrusiones en el emulador y, se abre la puerta a la inclusión e investigación de técnicas de detección de intrusiones o amenazas y de nuevas aplicaciones.

VII. Trabajo Futuro

En este apartado se comentará algunos aspectos a mejorar del emulador y algún campo donde se podría aprovechar su uso.

- **Optimización en la imagen de los contenedores.** Reducción del tamaño de los contenedores para reducir el tiempo y consumo empleado en el despliegue de la topología.
- **Unificación del emulador en una única máquina virtual.** Actualmente el emulador utiliza dos máquinas virtuales interconectadas a través de la red doméstica,

conllevando un uso de recursos que se podrían ahorrar si se llegase a utilizar únicamente una máquina virtual.

- **Mejorar la administración y gestión de los contenedores.** A medida que los escenarios se vayan ampliando y aumentando en complejidad, la estabilidad de los mismos puede ser comprometida si no se usa una plataforma de gestión de los dispositivos virtualizados. Por ello, se puede realizar una migración del uso de Docker Compose a un *cluster* de Kubernetes a la hora del despliegue del escenario. Esto a su vez, nos permite automatizar algunos pasos en los que se ve implicado el usuario con Docker-Compose, como puede ser el reinicio y levantamiento de un contenedor en caso de fallo en la red o el despliegue de nuevos contenedores en determinadas situaciones.
- **Implementación de nuevas funcionalidades en la red.** Al tratarse de un emulador de despliegues de redes orientado a la ciberseguridad, puede ser de utilidad aplicarlo para la implementación de *firewalls* o IDS (*Intrusion Detection System*) para probar diferentes formas de reducir, anular o detectar los posibles daños que puede provocar un atacante en cada situación, aprovechándonos de los ataques ya implementados en el emulador.
- **Introducción de nuevas aplicaciones.** Gracias a la topología desplegada con openLEON, se podrían incluir aplicaciones basadas en las tecnologías emergentes orientadas a mejorar la seguridad y privacidad en MEC. Por ejemplo, los contratos inteligentes basados en *blockchain*.
- **Inclusión de ataques específicos.** Los ataques incluidos en MECInOT están diseñados para suplir los ataques genéricos, que se encuentran en los entornos IT con un enfoque a dispositivos OT. Sin embargo, es necesario incluir las herramientas necesarias para poder explotar las vulnerabilidades, que se pueden encontrar en los protocolos de red OT y en dichos dispositivos. Además, también se debe de enfocar dichos ataques al aprovechamiento de las vulnerabilidades encontradas en los entornos vitualizados y en el paradigma MEC [27].

AGRADECIMIENTOS

Este trabajo ha sido realizado bajo la financiación europea del Fondo Social Europeo Plus (FSE+) con el contrato predoctoral en la Universidad de Castilla La-Mancha en el proyecto con identificador PI001482, por el Ministerio de Ciencia, Innovación y Universidades y los Fondos FEDER de la Unión Europea, referencia RTI2018-098156-B-C52 y la beca FPU 17/02007. Finalmente, también ha sido financiado por la JCCM [proyecto SB-PLY/17/180501/ 000353] y [proyecto SBPLY/21/180501/000195].

REFERENCIAS

[1] D. Ivanov, C. Tang, A. Dolgui, D. Battini, and A. Das, "Researchers' perspectives on industry 4.0: multi-disciplinary analysis and opportunities for operations management," *International Journal of Production Research*, pp. 1–24, 08 2020.

[2] P. K. R. Maddikunta, Q.-V. Pham, P. B, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[3] L. L. Dhirani, E. Armstrong, and T. Newe, "Industrial iot, cyber threats, and standards landscape: Evaluation and roadmap," *Sensors*, vol. 21, no. 11, p. 3901, 2021.

[4] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2018–2023," *Cisco White Paper*, pp. 1–36, 2020.

[5] A. T. Atieh, "The next generation cloud technologies: A review on distributed cloud, fog and edge computing and their opportunities and challenges," *ResearchBerg Review of Science and Technology*, vol. 1, no. 1, p. 1–15, Oct. 2021.

[6] T. K. Rodrigues, J. Liu, and N. Kato, "Application of cybertwin for offloading in mobile multiaccess edge computing for 6g networks," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 231–16 242, 2021.

[7] D. Borsatti, G. Davoli, W. Cerroni, and C. Raffaelli, "Enabling industrial iot as a service with multi-access edge computing," *IEEE Communications Magazine*, vol. 59, no. 8, pp. 21–27, 2021.

[8] C. Fiandrino, A. Pizarro, P. Mateo, C. Andrés Ramiro, N. Ludant, and J. Widmer, "openleon: An end-to-end emulation platform from the edge data center to the mobile user," *Computer Communications*, vol. 148, pp. 17–26, 12 2019.

[9] C. Alcaraz and J. Lopez, "Digital twin: A comprehensive survey of security threats," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2022.

[10] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.

[11] M. Liyanage, P. Porambage, and A. Y. Ding, "Five driving forces of multi-access edge computing," *arXiv preprint arXiv:1810.00827*, 2018.

[12] J. Liu, Q. Li, R. Cao, W. Tang, and G. Qiu, "Mininet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 166, pp. 255–267, 2020.

[13] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[14] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[15] R. S. Auliva, R.-K. Sheu, D. Liang, and W.-J. Wang, "Iiot testbed: A dds-based emulation tool for industrial iot applications," in *2018 International Conference on System Science and Engineering (ICSSE)*, 2018, pp. 1–4.

[16] S. Moysis, G. Zacharias, T. Demetris, P. George, and D. Marios D., "Fogify: A fog computing emulation framework," in *Proceedings of the 5th ACM/IEEE Symposium on Edge Computing*, ser. SEC '20. New York, NY, USA: Association for Computing Machinery, 2020.

[17] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso, "Fogbed: A rapid-prototyping emulation environment for fog computing," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[18] M. Peuster, H. Karl, and S. van Rossem, "Medicine: Rapid prototyping of production-ready network services in multi-pop environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 148–153.

[19] P. Wette, M. Dräxler, and A. Schwabe, "Maxinet: Distributed emulation of software-defined networks," *2014 IFIP Networking Conference*, pp. 1–9, 2014.

[20] I. Gomez-Miguelez, A. Garcia-Saavedra, P. Sutton, P. Serrano, C. Cano, and D. Leith, "srslte: an open-source platform for lte evolution and experimentation," 10 2016, pp. 25–32.

[21] B. Mishra and A. Kertesz, "The use of mqtt in m2m and iot systems: A survey," *IEEE Access*, vol. 8, pp. 201 071–201 086, 2020.

[22] D. Silva, L. I. Carvalho, J. Soares, and R. C. Sofia, "A performance analysis of internet of things networking protocols: Evaluating mqtt, coap, opc ua," *Applied Sciences*, vol. 11, no. 11, p. 4879, 2021.

[23] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd multimedia systems conference*, 2012, pp. 89–94.

[24] C. Mary, "Shellshock attack on linux systems-bash," *International Research Journal of Engineering and Technology*, vol. 2, no. 8, pp. 1322–1325, 2015.

[25] R. Ahmad and I. Alsmadi, "Machine learning approaches to iot security: A systematic literature review," *Internet of Things*, vol. 14, p. 100365, 2021.

[26] K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as software defined networking testing platform," in *International Conference on Communication, Computing & Systems (ICCCS)*, 2014, pp. 139–42.

[27] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.