ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

ANÁLISIS COMPARATIVO A NIVEL TRANSACCIONAL DE BROKERS MQTT (MOSQUITTO, MOSCA Y EMQ) CON RESPECTO A LA DISPONIBILIDAD EN INFRAESTRUCTURAS IoT ANTE ATAQUES DDoS.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SOFTWARE – MENCION SEGURIDAD

ING. AMAGUAYA RAMOS RICHARD XAVIER
richard.amaguaya@epn.edu.ec

Director: MSc. Gustavo David Salazar Chacón
gustavo.salazar@epn.edu.ec

Codirector: PhD. Jenny Gabriela Torres Olmedo
jenny.torres@epn.edu.ec

Quito-Ecuador 2020

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación “Análisis comparativo a nivel transaccional de Brokers MQTT (Mosquitto, Mosca y EMQ) con respecto a la disponibilidad en infraestructuras IoT ante ataques DDoS”, desarrollado por Amaguaya Ramos Richard Xavier, estudiante de la Maestría de Ingeniería de Software mención Seguridad, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Gustavo David Salazar Chacón

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Amaguaya Ramos Richard Xavier, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Amaguaya Ramos Richard Xavier

DEDICATORIA

El presente trabajo de titulación lo dedico a mi familia ya que gracias a ellos he podido dar este paso muy importante para mi vida profesional.

AGRADECIMIENTO

Agradezco a mis padres por haber confiado en mí para seguir preparándome profesionalmente dentro de esta prestigiosa institución como lo es la Escuela Politécnica Nacional, y a todos los docentes que cada día de clase han sabido impartir conocimientos y compartir vivencias propias que me sirven de guía para aplicar dentro del ámbito laboral.

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

LISTA DE FÓRMULAS

RESUMEN

En el presente proyecto de titulación se presenta el desarrollo de infraestructuras IoT (Internet of things) como pruebas de concepto evaluando tres diferentes Brokers MQTT (Mosquitto, Mosca y EMQ) ante ataques de DDoS. El objetivo del proyecto es brindar información sobre cuál de ellos es óptimo con respecto a la disponibilidad tomando en consideración cuatro métricas de calidad (KPI's-Key performance indicator): 1) MTTD-Mean Time to Detect, 2) MTTR-Mean Time to Resolve, 3) MTTF-Mean Time to Failure, y 4) MTBF-Mean Time Between Failures.

Para el desarrollo de todo el proyecto se aplicó dos metodologías, la primera corresponde a la metodología de desarrollo XP y la segunda al método comparativo, lo cual nos permitió cumplir satisfactoriamente cada uno de nuestros objetivos planteados en el proyecto. Para esto, iniciamos con la identificación del problema central y recolección de datos para su posterior análisis.

De acuerdo con la revisión literaria se elaboró un caso práctico; durante la implementación de las infraestructuras IoT se evidenció ciertas diferencias con cada uno de los Brokers MQTT y durante las pruebas con dos herramientas utilizadas para ataques DDoS Metasploit (SYNFLOOD) y LOIC (Low Orbit Ion Cannon), se recolectó datos esenciales para la posterior interpretación de resultados, en la que se pudo determinar qué bróker es óptimo. Adicionalmente se elaboró recomendaciones antes de iniciar un proyecto con este tipo de tecnologías emergentes.

Palabras clave: Broker, MQTT, KPIS, DDoS, Disponibilidad.

ABSTRACT

In this project, the development of IoT (Internet of things) infrastructures is presented as proofs of concept evaluating three different MQTT Brokers (Mosquitto, Mosca and EMQ) against DDoS attacks. The objective of the project is to provide information on which of them is optimal with respect to availability, considering four quality metrics (KPI's-Key performance indicator): 1) MTTD-Mean Time to Detect, 2) MTTR-Mean Time to Resolve, 3) MTTF-Mean Time to Failure, and 4) MTBF-Mean Time Between Failures.

For the development of the project, two methodologies were applied. The first methodology corresponds to the XP development methodology and the second corresponds to the comparative method, which allowed us to satisfactorily fulfill each objective set forth in the project. For this, we start with the identification of the central problem and data collection for subsequent analysis. In accordance with the literature review, a practical case was prepared. During the implementation of the IoT infrastructures, certain differences were evident with each of the MQTT Brokers and during the tests with two tools used for DDoS attacks: Metasploit (SYNFLOOD) and LOIC (Low Orbit Ion Cannon), essential data was collected for subsequent interpretation of results. As conclusion, it was possible to determine which broker is optimal. Additionally, recommendations were prepared before starting a project with this type of emerging technologies.

Keywords: Broker, MQTT, KPIS, DDoS, Availability.

1. INTRODUCCIÓN

Actualmente la "Industria 4.0" es tan irreversible y profunda como sus predecesoras (máquinas de vapor, energía eléctrica, sistemas informáticos), Sin embargo, la principal diferencia es que, evolucionará y se adaptará a una mayor velocidad[1], dando paso a nuevas tecnologías para el uso de las personas, industria e ingeniería.

El Internet de las cosas (IoT, por sus siglas en inglés) como tecnología emergente ha sido un avance que ha permitido mejorar productos y servicios, siendo estos implementados en hogares inteligentes, vehículos autónomos, equipos controlados a distancia, y en diversas áreas como salud, transporte e ingeniería [2]. Esta tecnología ha ido evolucionando y mejorando tanto en su infraestructura como en su funcionalidad, seguridad y protección en el transporte de datos[3].

El presente proyecto de titulación tiene un enfoque en el ámbito de la seguridad, específicamente en cuanto a disponibilidad de infraestructuras IoT a nivel transaccional. Se analizarán los tres principales brokers MQTT que están en auge: Mosquitto, Mosca y EMQ, los mismos que permiten la utilización del protocolo MQTT en la versión 3.1.1 que es utilizado para mensajería dentro de este tipo de infraestructuras.

MQTT es el foco central de la comunicación entre publicadores y subscriptores para el envío de mensajes entre dispositivos y aplicativos conectados; este permite mecanismos de enrutamiento uno a uno, uno a varios y varios a varios.

En cuanto a los brokers, Mosquitto es ampliamente utilizado en microprocesadores con MQTT debido a su facilidad de implementación y configuración dentro de diferentes ambientes utilizando pocos recursos, al igual que Mosca que está basada en JavaScript permitiendo como objetivo principal la comunicación de publicación y suscripción [4]. Por otro lado, está EMQ, un bróker escrito en Erlang OTP de fácil escalabilidad, y que utiliza pocos recursos para su implantación.

Estos tres brokers MQTT son "open source" lo que implica que se debe tomar en consideración que la información, ayuda o soporte técnico es mucho más limitada.

Con la finalidad de corroborar que estos brokers poseen los parámetros de seguridad necesarios para soportar diversos ataques, serán sometidos a un ataque de denegación de servicios (DDoS), este tipo de ataques causan que un servicio sea inaccesible al momento de realizar una petición hacia dispositivos conectados en este caso a una infraestructura IoT, esto nos permitirá evaluar su comportamiento con respecto a la disponibilidad tomando en consideración los KPI's de seguridad (MTTD, MTTR, MTTF y MTBF).

.

2. OBJETIVOS

2.1 Objetivo general

Comparar el comportamiento de brokers MQTT (Mosquitto, Mosca y EMQ) con respecto a la disponibilidad en infraestructuras IoT ante ataques DDoS.

2.2 Objetivos específicos

- Identificar las características principales y la funcionalidad básica de los brokers MQTT-Mosquitto, Mosca y EMQ.
- Implementar tres infraestructuras IoT como pruebas de concepto del uso de brokers MQTT-Mosquitto, Mosca y EMQ.
- Medir los KPI's MTTD, MTTR, MTTF y MTBF con respecto a la disponibilidad en las tres infraestructuras sometidas a ataques DDoS.

3. MARCO TEÓRICO

3.1. Internet de las Cosas (IoT)

El Internet de las cosas o IoT, por sus siglas en inglés, puede ser definido como una red de dispositivos que interactúan entre sí, tal como se muestra en la Figura 1, es decir dispositivo a dispositivo (D2D, por sus siglas en inglés) mediante el Internet[5], además IoT es considerado como un marco de protocolos, tecnologías, procesos y flujos de datos, que requieren ciclos de desarrollo avanzados y visibilidad y monitoreo total para tener éxito[6]. Estos dispositivos no necesariamente necesitan de la interacción de un ser humano para su correcto funcionamiento.

Actualmente IoT es una tecnología que a pesar de conectar diversos dispositivos para un control y monitoreo ya sea en áreas de salud, ingeniería, etc., posee ciertas incertidumbres sobre afecciones físicas, medioambientales, económicas y de seguridad[7]. El poseer este control de las cosas a través del Internet genera que el ser humano para este año 2020 adquiera muchos más dispositivos, ascendiendo a más de 50 mil millones de estos en funcionamiento[8] generando un incremento considerablemente alto en datos u información generados por los mismos que debe ser almacenados en diferentes servidores físicos o en la nube.

Este incremento en el tráfico de datos y almacenamiento genera que puedan existir diversas vulnerabilidades que afecten al triángulo de confidencialidad, integridad y disponibilidad (CIA-triad) comprometiendo así la integridad de las personas.

Figura 1 – IoT (Internet of things)

3.2. CIA-Triad

La triada CIA (confidencialidad, integridad y disponibilidad) son parámetros de seguridad para el desarrollo de un sistema, en este caso enfocado a infraestructuras IoT (Figura 2).
Para este tipo de proyecto de desarrollo se toma como enfoque principal la disponibilidad ya que esta permitirá conocer qué afecciones puede causar si estas llegan a fallar durante el envío de datos.

3.2.1 Disponibilidad

La disponibilidad [7] es la denegación de uso no autorizada: un intruso puede evitar que un usuario autorizado se refiera o modifique información, aunque el intruso no pueda hacerlo, ni modifique la información por sí mismo.

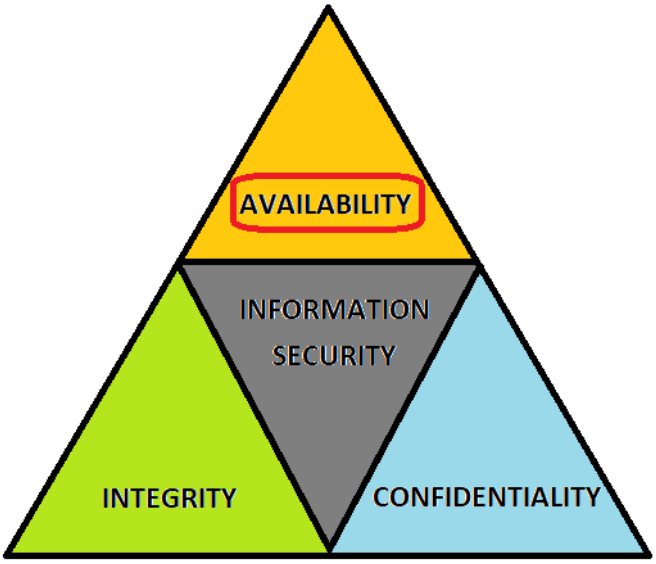

Figura 2 – CIA-Triad- Enfocando la Disponibilidad

Dentro del área de sistemas y tecnología un sistema se considera utilizable cuando es efectivo y eficiente, y sus usuarios son generalmente satisfechos con su desempeño de tareas específicas dentro de un determinado entorno[9].
Además, al garantizar que un sistema esté disponible siempre que los usuarios lo necesiten, permite la capacidad de evitar errores peligrosos que pueden llevar a afecciones más graves dentro del mismo.

3.3. Denegación de Servicio Distribuido (DDos)

Dentro de las tecnologías emergentes como IoT los ataques de Denegación de Servicio Distribuido (DDoS) son posibles debido a que en la web existen herramientas de libre

acceso que permiten la realización de estos[10]. Con el avance de los años los piratas informáticos han ido encontrado la forma de potenciar este tipo de ataques, tal es el caso de la creación de botnets.

Un ataque DDoS es realizado desde varios nodos y posee como objetivo principal atacar a un servidor (Figura 3), en este caso a una infraestructura IoT, consumiendo los recursos y ancho de banda hasta que esta colapse y no permita el ingreso de más peticiones por parte de dispositivos conectados y de usuarios legítimos, ocasionando que estos datos en transmisión se pierdan[11].

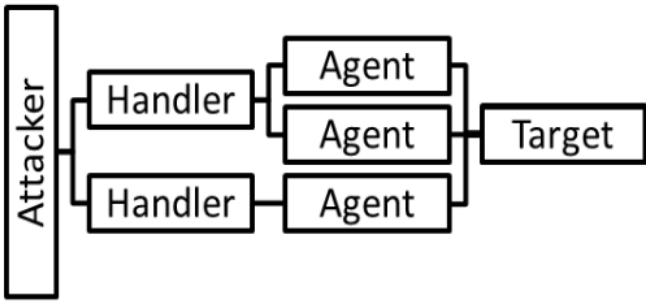

Figura 3 – Ataques DDoS[11]

Actualmente, las seguridades implementadas en diversos dispositivos son adaptables a una amplia gama de entornos de red siendo lo más precisos posibles[10].

Estas medidas optadas pueden en ciertos casos ocasionar falsos positivos que pueden dar lugar a denegar servicios a usuarios legítimos, y en otros casos a falsos negativos los cuales dan lugar a ataques no percibidos.

La evolución de los ataques DDoS han permitido que se pueda vulnerar herramientas de seguridad implementadas en pequeñas, medianas o grandes empresas, conllevando a que existan varios tipos de ataques DDoS tales como:

- Inundación UDP (UDP-Flood)

  Consiste en que un atacante inunda diferentes paquetes UDP en diferentes puertos (Figura 4), ocasionando que el host compruebe el puerto de escucha y responda con paquetes ICMP de destino inalcanzable.[11]

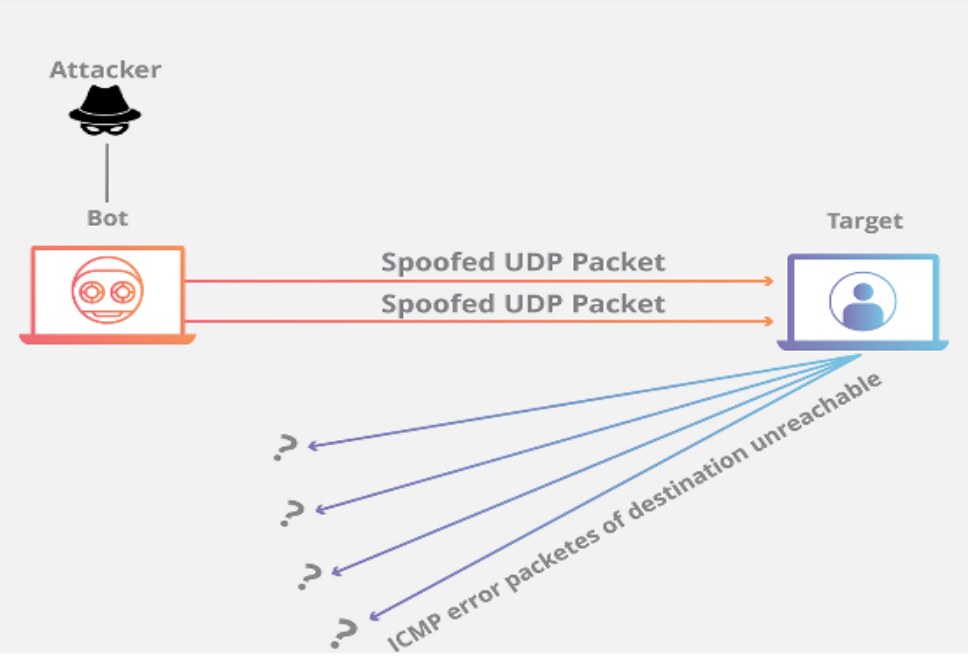

Figura 4 – UDP-FLOOD[12]

- Inundación ICMP/PING (ICMP/PING-Flood)

  Tiene como objetivo principal inundar el host de destino con solicitudes ICMP que envían paquetes de forma rápida sin esperar una respuesta (Figura 5), ocasionando que el ancho de banda colapse[11].

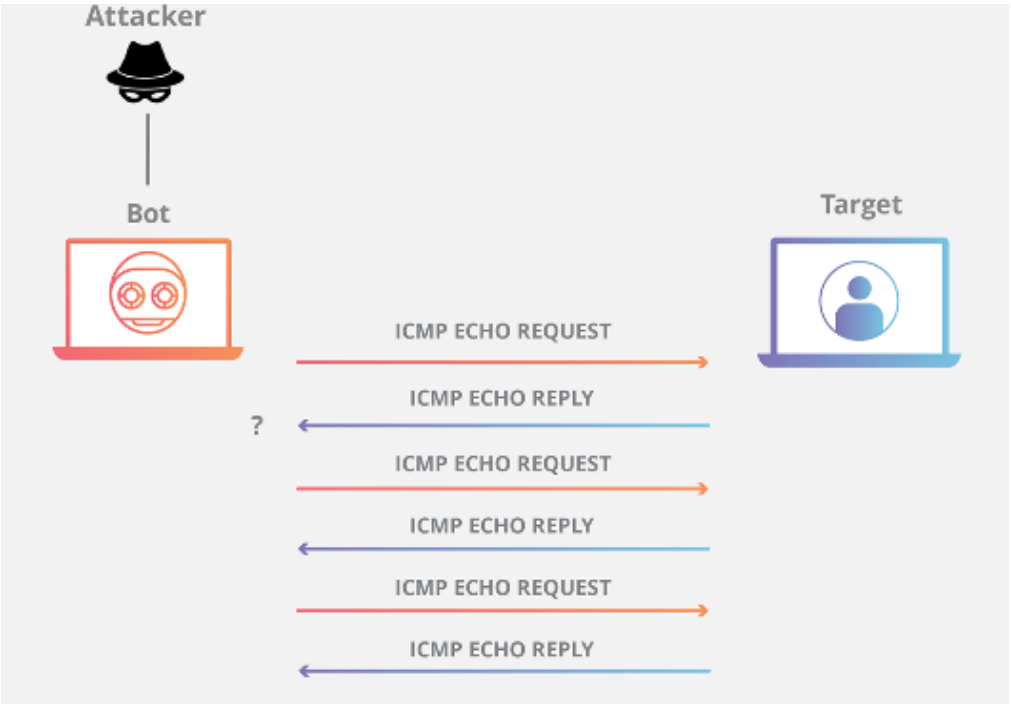

Figura 5 – ICMP/PING-FLOOD[12]

- Inundación SYN (SYN-Flood)

  Es esencial para ataques para el protocolo MQTT ya que este tipo de ataques busca una vulnerabilidad en las conexiones TCP (Figura 6). Posee como objetivo principal inundar con solicitudes SYN pero sin obtener respuestas SYN-ACK del host ocasionando que este siga esperando el reconocimiento de cada una de las solicitudes dando como resultado la caída del mismo[11].

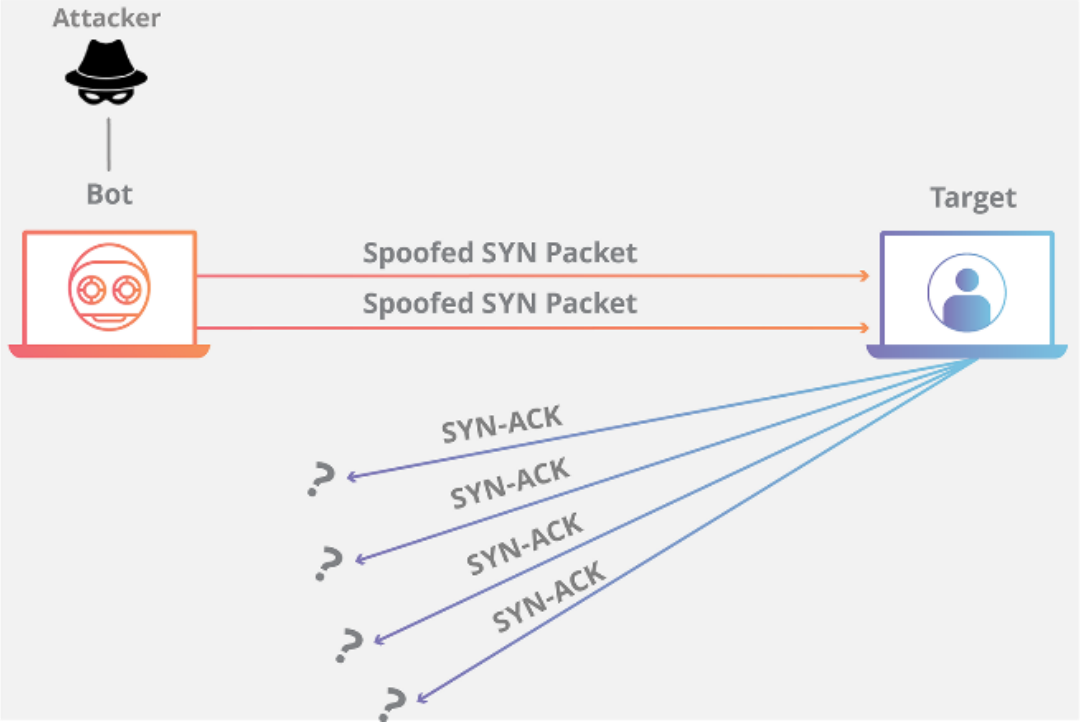

Figura 6 – SYN-FLOOD[12]

- Ping de la Muerte (Ping of death)

  Es un tipo de ataque DDoS que tiene como objetivo principal enviar múltiples pings maliciosos hacia un ordenador sobrepasando la longitud máxima de un paquete IP que es de 65.535 bytes, ocasionando que los buffers de memoria asignados para cada paquete se desborde y este colapse tal como se muestra en la Figura 7 [11].

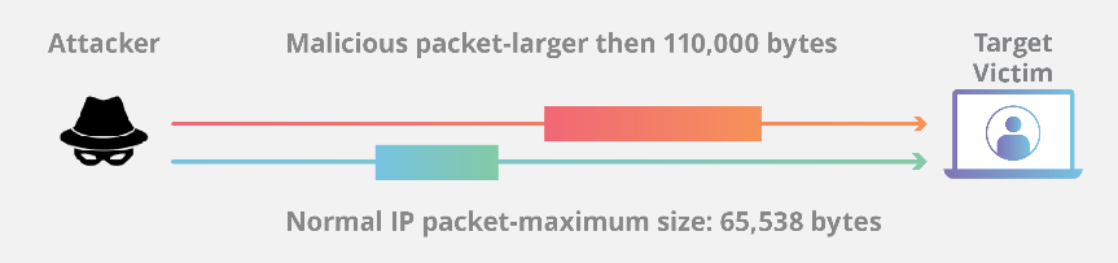

Figura 7 – PING OF DEATH[12]

- DDoS de día cero (Zero-days DDoS)

  Son ataques nuevos que explotan vulnerabilidades que no son conocidas por la gente o el fabricante del producto o servicio (Figura 8), permitiendo así ejecutar código malicioso que ocasiona el colapso de este[11].

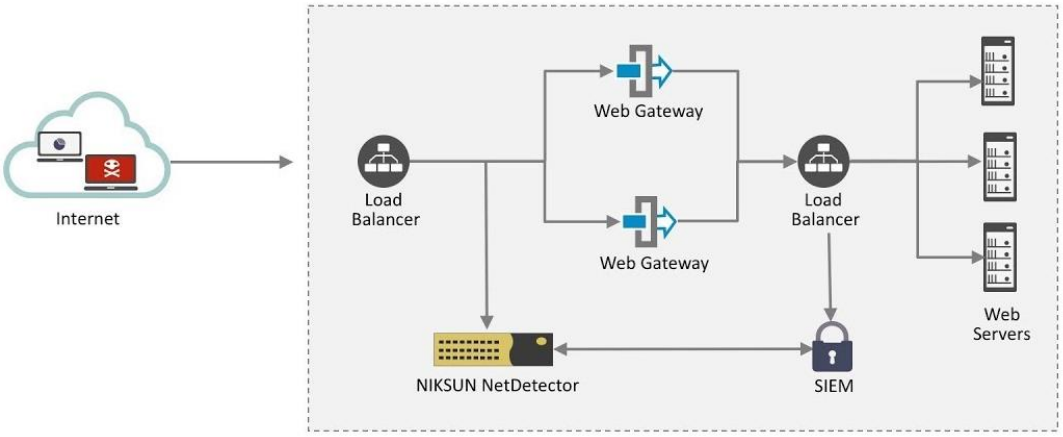

Figura 8 – ZERO-DAYS DDoS

3.4. LOIC

Low Orbit Ion Cannon es una herramienta diseñada para de DDoS. Está desarrollado en base a lenguaje C++ y una biblioteca Qt4 (Framework multiplataforma orienta a objetos). Es de libre acceso y de fácil uso e implementación. Tiene como objetivo principal realizar el envío de varios paquetes TCP, UDP y ICMP hacia una red en este caso un broker MQTT consumiendo recursos como tiempo de CPU, almacenamiento y ancho de banda de este [13].

LOIC es una herramienta peligrosa ya que posee ciertas características principales que pueden causar daño tales como:

- Ataques mediante TCP, UDP y ICMP.
- Permite la selección de cantidad de hilos.
- Permite la selección de velocidad para el ataque.
- Permite elegir el ataque mediante URL o IP.

3.5. Bróker MQTT

El protocolo de transporte de telemetría de cola de mensajes o MQTT por sus siglas en inglés, fue ideado por IBM con el propósito de brindar una comunicación D2D dentro de infraestructuras IoT ya que este requiere poco ancho de banda para su funcionamiento dentro de dispositivos con poca CPU y RAM (Figura 9). Es ideal para sensores y microprocesadores [14].

MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom — ahora Eurotech — en 1999 [15][16] como una forma rentable y confiable de conectar los dispositivos de monitoreo utilizados en las industrias del petróleo y el gas a servidores empresariales remotos.

Este protocolo está basado en la topología estrella, la cual posee como nodo central un bróker que es el encargado de gestionar la red de conexiones con los dispositivos y transmitir los mensajes.

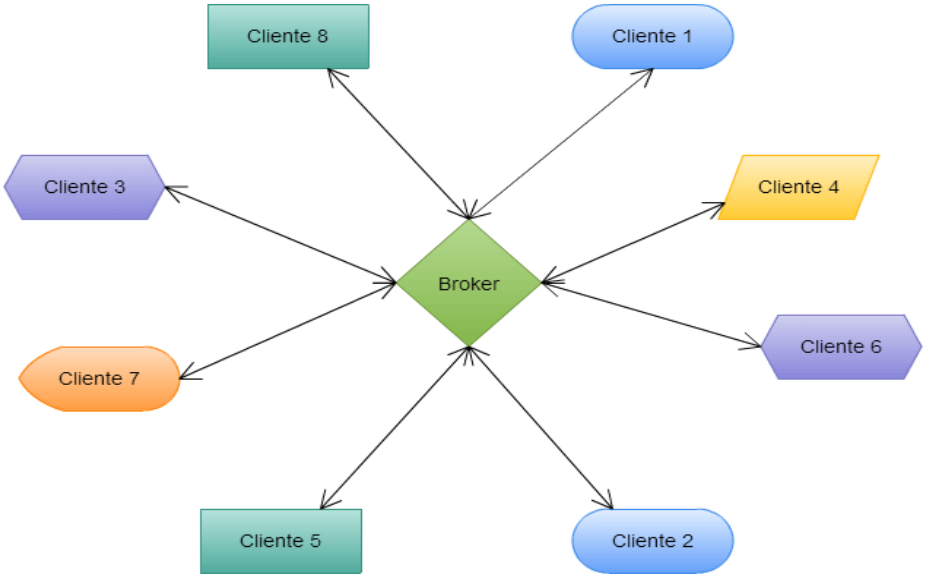

Figura 9 – Topología estrella MQTT

Este tipo de topología del protocolo para mantener siempre la conexión con cada uno de los canales entre los dispositivos realizan dos actividades:

- PINGREQ. - Es un paquete enviado por el cliente (Figura 10).
- PINGRESP. - Es el paquete de respuesta enviado por el bróker (Figura 10).

Estas comunicaciones realizadas están basadas en topics o temas dentro de cuales se encuentran emisores y receptores, además se debe tomar en consideración que estos receptores y emisores deben pertenecer al mismo topic para poder comunicarse[17].

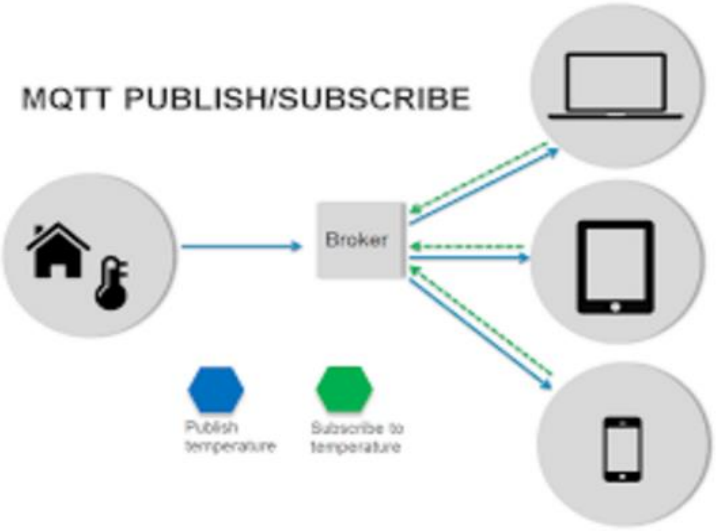

Figura 10 – MQTT Emisores/Receptores

3.5.1. Eclipse Mosquitto

Es un bróker MQTT de código abierto que permite la implementación del protocolo MQTT (Figura 11). Además es un intermediario de mensajes liviano de fácil uso para dispositivos de baja potencia como Arduino, o Raspberry pi[18].

Figura 11 – Eclipse Mosquitto

Características:

- Admite MQTTv3.1, v3.1.1 y v5.0
- Utiliza pocos recursos aptos para sensores de baja potencia, dispositivos móviles, y microcontroladores.
- Posee un repositorio de vulnerabilidades detectadas a tomar en consideración
- De fácil uso e implementación disponible para multiplataformas.

3.5.2. Mosca

Es un bróker MQTT utilizado para desarrollo de infraestructuras IoT basado en el protocolo MQTT. Mosca está basado en una librería de JavaScript [19] de fácil uso e implementación (Figura 12).

Sus características principales son:

- MQTTv 3.1 y v3.1.1
- QoS 0 y QoS 1
- Almacena paquetes fuera de línea para QoS1
- Rápido
- Se puede usar dentro de aplicaciones node.js

Figura 12 – Mosca

Es un bróker, que, a pesar de su eficiencia, posee ciertas limitaciones como en la funcionalidad de escalabilidad, ya que necesita de otro agente para poder cumplir esa característica. Entre los agentes se pueden mencionar:

- Redis
- MongoDB
- RabbitMQ
- ZeroMQ

3.5.3. EMQ

Es un agente para mensajería MQTT distribuido y altamente escalable. Está escrito en Erlang/OTP, de fácil uso e implementación en dispositivos x86/ARM ya sea con recursos limitados hasta en nubes privadas, públicas e híbridas (Figura 13) [20].

Características:

- Soporta MQTTv3.1, v3.1.1 y v5.0
- Garantiza una integración fluida con clientes y herramientas MQTT

- Permite escribir complementos para admitir protocolos en la capa TCP y UDP
- Permite un millón de conexiones concurrentes.
- Garantiza una latencia en milisegundos
- Varios nodos pueden trabajar en conjunto garantizando confiabilidad y disponibilidad

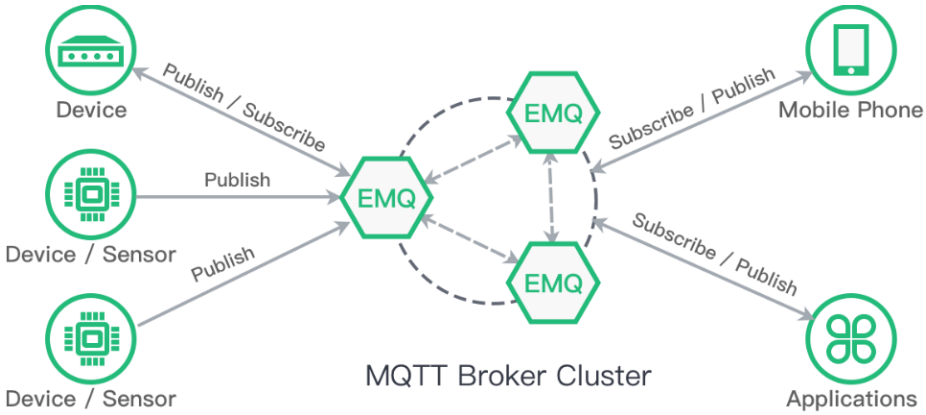

Figura 13 – EMQ[20]

3.6. Node-Red

Es una herramienta basada en JavaScript desarrollado por IBM que permite tener combinaciones de hardware IoT, API y otros servicios en línea de manera inteligente [21]. Es de fácil uso y acceso, posee un panel que funciona de dos maneras: arrastrar, soltar y conectar nodos, o importar código JavaScript Estos nodos proporcionan funcionalidades tales como monitoreo, lectura y escritura. Además posee una flexibilidad permitiendo conectar nodos de entrada, salida y procesamiento, controlar cosas o enviar alertas (Figura 14) [22].

Esta herramienta posee tres componentes básicos que permiten la creación rápida y configuración en tiempo real de prototipos de aplicaciones IoT:

1. Panel de Nodos
2. Panel de Flujo
3. Panel de depuración e información

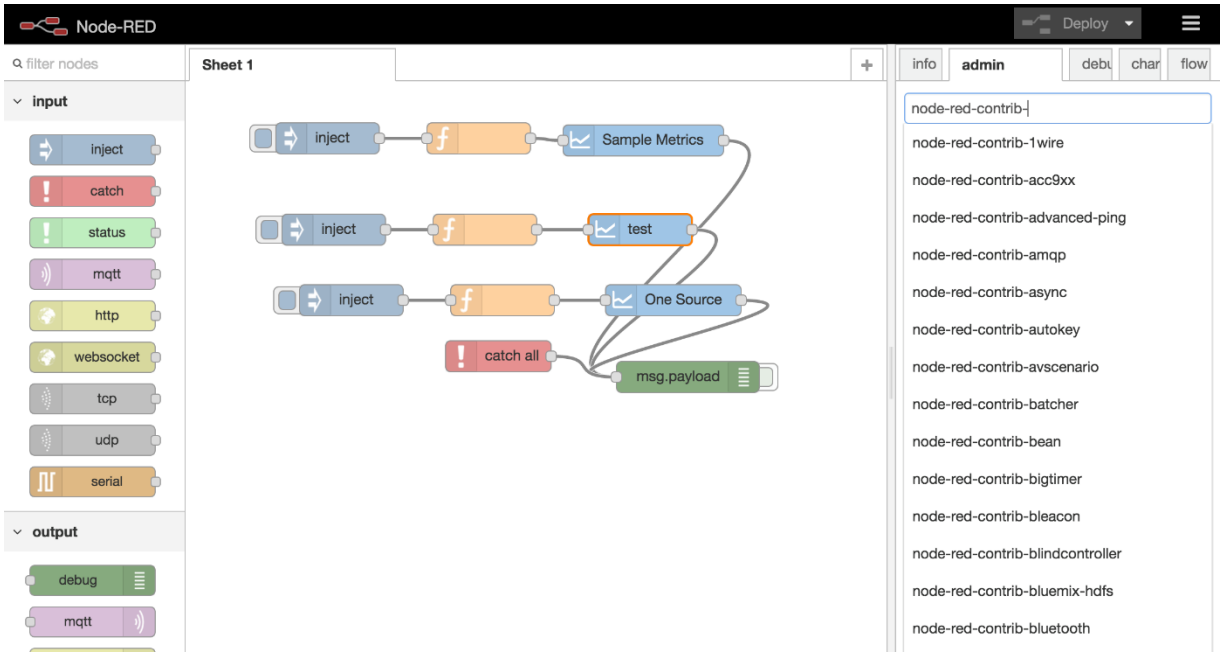

Figura 14 – *Node-red*

3.7. Medidores de Desempeño - KPI

Los medidores de desempeño o KPI por sus siglas en inglés (*Key Performance Indicator*) nos permiten medir un proceso o evaluar un objetivo previamente definido, en este caso medir el desempeño de las infraestructuras IoT con respecto a la disponibilidad.

Los KPIS MTTD, MTTR, MTTF y MTBF son fundamentales para obtener resultados que nos ayuden a una toma de decisiones correcta sobre el bróker MQTT óptimo con respeto a los demás.

3.7.1. Tiempo Medio para Detectar - MTTD

El tiempo medio para detectar o MTTD por sus siglas en inglés (*Mean Time to Detect*), es una medida para conocer cuánto tiempo se demora en este caso el bróker en darse cuenta de que es víctima de un incidente de seguridad. Un tiempo corto puede pasar desapercibido como un incidente de seguridad, al ser un tiempo mayor esto posee una relevancia de que sea un posible ataque[23].

$$MTTD=\text{Total Time Incidents} / \text{Total number of incidents} \quad (1)$$

3.7.2. Tiempo Medio para Resolver - MTTR

El tiempo medio para resolver o MTTR por sus siglas en inglés (*Mean Time to Resolve*), es una medida que permite determinar cuánto tiempo le toma al bróker en resolver la incidencia después de que esta haya acontecido o sea descubierta. Si el tiempo es alto

esto indica que los recursos actuales no son suficientes para poder mitigar la amenaza identificada[23].

MTTR= Downtime / Number of Failures (2)

3.7.3. Tiempo Medio para Fallar - MTTF

El tiempo medio para fallar o MTTF por sus siglas en inglés (*Mean Time to Failure*), es una medida que permite conocer la vida útil del elemento antes de fallar, ya sea por producto de una incidencia o que directamente el bróker deje de funcionar[23].

MTTF= Total Hour of Operation / Total Number of Units (3)

3.7.4. Tiempo Medio entre Fallos - MTBF

El tiempo medio entre fallos o MTBF por sus siglas en inglés *(Mean Time between Failures*), es una medida que permite determinar cada que tiempo un bróker es susceptible a sufrir una incidencia o fallo dentro de su funcionamiento[23].

MTBF= Operating Time / Number of Failures (4)

4. METODOLOGÍA

El presente proyecto de titulación amerita de un análisis exhaustivo de teoría, pruebas y comprobaciones para poder dar alcance a los objetivos planteados. Para esto es necesario utilizar dos metodologías ya que una será específicamente para el desarrollo del caso práctico: metodología de desarrollo ágil XP; y la otra corresponde al Método Comparativo. Estas metodologías ayudan a encontrar cada una de las características principales, ventajas y desventajas, funcionalidades, especificaciones, requerimientos, entre otras, de cada uno de los brokers MQTT Mosquitto, Mosca y EMQ respectivamente, además de dividirlas en diferentes actividades para su desarrollo.

4.1. Criterios de selección de metodologías

En la actualidad SCRUM y XP son las metodologías de desarrollo ágil más utilizadas dentro de empresas para el desarrollo de proyectos de software[24].

Se elige a XP como metodología principal para el presente proyecto de titulación debido a:

- Se puede realizar cambios durante las iteraciones además de poder reemplazar requerimientos equivalentes entre sí.
- Permite la utilización de buenas prácticas XP tales como *test-driven development, pair programing*, diseño simple, entre otras.
- Se centra en la producción de un producto de mayor calidad que la administración del proyecto.
- Las iteraciones son muy cortas durando entre 1 a 2 semanas.
- Trabaja en un estricto orden de prioridad de requerimientos.

En base a temas investigativos[25] se elige el método comparativo como metodología de investigación frente a otros debido a:

- Ser un método versátil y efectivo en muestras pequeñas.
- Observar dos o más casos, objetos o en este caso tecnologías con el fin o propósito de comprender eventos desconocidos.
- Puede diseñarse de forma activa, participativa, inductiva y de descubrimiento, situando actividades que ponen en práctica las técnicas dinámicas de observación y evaluación [26].
- Obtener resultados que conlleven a la solución de un problema o al entendimiento de este, e incluso brindar posibles soluciones a futuras investigaciones [26].

4.2. Extreme programing - xp

XP es una metodología ágil orientada a obtener el éxito en el desarrollo de un proyecto de software. Es fundamental ya que permite una realimentación continua, simplicidad en las soluciones planteadas y en este caso nos ayuda ya que los requerimientos son cambiantes[27].

La metodología XP durante el desarrollo de las infraestructuras IoT permite tener un control ordenado y sistemático sobre el diseño y pruebas que se van realizando, además es óptima para trabajar en conjunto con el Método comparativo ya que permite dividir la complejidad del proyecto en cuatro fases: 1) Exploración, 2) Planificación, 3) Iteraciones y; 4) Producción. Cada una de estas fases cumplen con el ciclo de vida dinámico que cuenta con cuatro etapas: Análisis, Diseño, Desarrollo y Pruebas (Figura 15); de forma que se brinde una solución práctica y eficiente sobre cada una de las infraestructuras con los brokers correspondientes[28].

- **Fase de Exploración**

  Esta fase posee como objetivo principal familiarizarse con las herramientas, tecnologías y prácticas que se van a utilizar durante el desarrollo del proyecto investigativo, además permite la recolección inicial de requerimientos iniciales y elaborar un prototipo que demuestre que la arquitectura utilizada es viable o válida para el mismo[27].

- **Fase de Planificación**

  En esta fase se establece los requerimientos esenciales que se necesitan para el desarrollo completo del proyecto, además de la elaboración de un cronograma de actividades en base al tiempo disponible y al alcance de este con fechas planteadas para la elaboración de cada etapa[28].

- **Fase de iteraciones**

  La fase de iteraciones consiste en dar cumplimento con cada una de las actividades planteadas en el cronograma, estas estarán divididas en varias iteraciones con una duración de dos a tres semanas, al final de cada iteración se revisará requerimientos no cumplidos, pruebas no superadas entre otras[27].

- **Fase de Producción**

  En esta etapa se realiza una revisión completa a los requerimientos planteados para comprobar en este caso las infraestructuras IoT estén optimas y adecuadas para someterlas ante ataques DDoS.

Cabe mencionar que cada una de estas fases da cumplimiento con cada una de las etapas del ciclo de vida dinámico para que el producto final sea exitoso y satisfaga las necesidades principales

Figura 15 – Ciclo de Vida

En base a los requerimientos y medidas tomadas para el presente proyecto se posee las siguientes ventajas y desventajas de la metodología XP

Tabla 1 – Ventajas y Desventajas Metodología Xp

Ventajas	Desventajas
• Desarrollo organizado	• Se aplica a proyectos de poca duración.
• Tasa de errores muy baja	• Costos en correcciones en caso de fallos es elevado
• Permite la realización de pruebas continuas.	• Está sujeto a cambios constantes
• Permite ahorrar tiempo durante el desarrollo del proyecto	

4.3. Método comparativo

Es un método ordenado y sistemático que comprueba o compara dos o más actividades o aplicaciones, en este caso el desarrollo e implementación de tres infraestructuras IoT como pruebas de conceptos con cada uno de los brokers MQTT (Mosquitto, Mosca y EMQ).[29]

Este método, dentro del presente proyecto, permite generar nuevo conocimiento a través de partes conocidas como los brokers Mosquitto, Mosca y EMQ que son muy utilizados dentro del transporte de mensajería en IoT, generando a partir de ello

ventajas, desventajas, características principales, funcionalidades entre otras y determinar en base a los KPI planteados que bróker es óptimo para la implementación.

Entre las técnicas a aplicarse por parte del método son:

- Identificación del problema
- Criterios de selección
- Análisis de brokers MQTT
- Explicación e interpretación de resultados

5. IMPLEMENTACIÓN DE CASO PRÁCTICO

5.1. Identificación del problema

Actualmente, el uso e implantación de infraestructuras IoT con MQTT ha ido en incremento para la cual han surgido diversos brokers de conexión, control y monitoreo de dispositivos conectados a la red, por tal motivo es necesario conocer si estas nuevas herramientas tecnológicas brindan la seguridad respectiva para el tránsito o transporte de datos o información, obteniendo como problemática principal del proyecto investigativo el conocer qué broker MQTT, entre Mosquitto, Mosca y EMQ, es más eficiente y óptimo con respecto a la disponibilidad en detectar, solucionar o evitar ataques DDoS que atenten a su infraestructura IoT.

5.2. Criterios de selección

En base a los datos recolectados y analizados, se dividió los criterios de selección en tres partes clave:

- Características y funcionalidades principales, facilidades de uso e implementación de cada uno de los brokers MQTT.
- Recursos necesarios para desarrollo e implementación de las infraestructuras como pruebas de concepto.
- KPI's de seguridad.

Características y funcionalidades principales

Se elaboró la siguiente tabla comparativa en base a la información analizada:

Tabla 2.- Comparativa Mosquitto, EMQ, Mosca

MOSQUITTO	EMQ X	MOSCA
Permite una escalabilidad amplia en diversos dispositivos gracias a ser liviano.	Es un agente MQTT escalable para dispositivos con limitaciones como poca memoria entre otras.	Es un corredor de Node.js con MQTT o puede ser considerado como un intermediario entre Node.js y MQTT.
Son considerados Herramientas de "Cola de mensajes		
Se basa de una arquitectura de poseer un núcleo y subproceso único que permite admitir dispositivos integrados	Soporta MQTT-SN, CoAP, STOMP, SockJS	No es apto para servicios a gran escala

e implementados en recursos limitados		
No es apto para servicios a gran escala	Apto para servicios de gran escala, 1 millón de conexiones a un solo servidor	Puede ser utilizado de forma independiente o dentro de otra aplicación con Node.js
Se ejecuta en cualquier microcontrolador de baja potencia debido a ser liviano y compacto.	Alta concurrencia con una baja latencia	Es un servidor muy simple que proporciona una API basada en eventos para crea una lógica MQTT.
Son Open Source admitiendo las versiones de MQTT v3.1 y v3.1.1, solo Mosquitto y EMQ soporta v5.0		
Provee líneas de comando simple de publicación y suscripción	Los nodos pueden trabajar juntos como un cluster para garantizar confiabilidad y disponibilidad	Provee líneas de comando simple de publicación y suscripción
Soporta QoS 0,1 y 2		Soporta QoS 0 y 1
Poseen hilos de seguridad ante ataques		
Los dos broker son compatibles entre sí, EMQ admite otros protocolos de comunicación y acceso.		

En base a las facilidades de uso e implementación se elaboró la siguiente tabla tomando en consideración las experiencias de usuario al momento de trabajar con ellas:

Tabla 3.- Facilidades de uso e implementación de brokers MQTT

CRITERIOS DE SELECCIÓN	MOSQUITTO	MOSCA	EMQ X
	Instalación e implementación		
Links o comandos de descarga	**X**	**X**	**X**
Guía de instalación	**X**	**X**	**X**
Guía de configuración	**X**	**X**	**X**
Soporte	**X**	**Limitado**	**X**
CRITERIOS	**Facilidades de Uso**		
Comandos de configuración del broker	**X**	**X**	**X**
Archivo de fácil configuración	**X**	**X**	**X**
Panel de status del broker	**X**	**X**	**X**
Panel de Monitoreo	**-**	**-**	**X**

Recursos Necesarios

Durante el desarrollo del proyecto fue necesario la utilización de varios recursos, tanto intelectuales como físicos:

Tabla 4.- Recursos Utilizados

HUMANO	HARDWARE		SOFTWARE	
	MÁQUINA FÍSICA	MÁQUINA VIRTUAL	MÁQUINA FÍSICA	MÁQUINA VIRTUAL
Desarrollador	Core i7 6700k	Procesador 8	Windows 10	Kali Linux
Tutor	RAM 16GB	RAM 15GB	VMWARE 15	Metasploit
Codirector	SSD 1TB	Almacenamiento 20 GB		LOIC
				Ubuntu 18.04
				Node-red
				Mosquitto
				Mosca
				EMQ
				Wireshark

KPI's de seguridad

Los KPI's fundamentales para este proyecto, en el que se involucra la disponibilidad de los brokers MQTT son cuatro: MTTD, MTTR, MTTF y MTBF, los cuales permitirán una toma de decisiones correcta sobre la eficiencia y eficacia de los mismos ante ataques DDoS, tal como se muestra en la Figura 16.

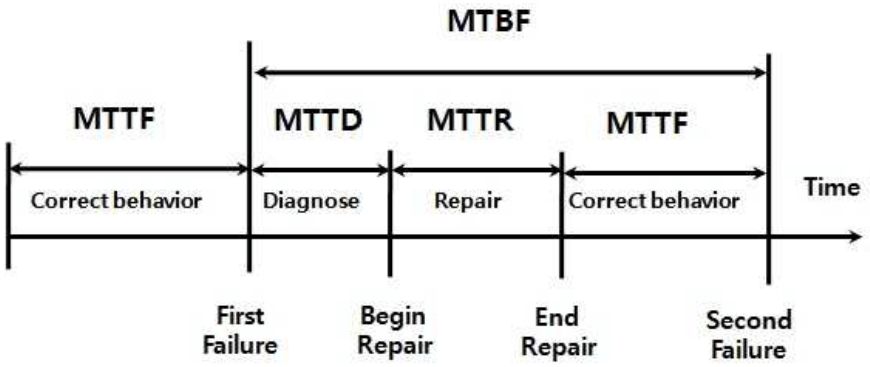

Figura 16 – KPI's de seguridad

5.3. Análisis de brokers MQTT- caso práctico

Se diseña y desarrolla tres infraestructuras IoT como pruebas de concepto simulando el uso de dos tipos de sensores (Temperatura y luz).

En cada una de las máquinas virtuales a utilizarse se ha asignado las mismas características de hardware como procesador, RAM y almacenamiento para tener un análisis de resultados en las mismas condiciones.

En este caso es necesario la utilización de siete máquinas virtuales, tres destinadas para la simulación de las infraestructuras con Mosquitto, Mosca y EMQ y cuatro destinadas para máquinas maliciosas.

Se ha planteado un escenario simulando el uso de infraestructuras IoT dentro de una empresa pequeña, en la que es necesario controlar la temperatura de cuartos fríos para un data center, neveras para el control de mercadería y control de encendido de luces de las mismas habitaciones y generar un control de consumo de energía tal como lo indica la Figura17.

Por último, se establece un caso de atacantes que buscan realizar un daño a la empresa mediante ataques DDoS hacia el broker MQTT como lo indica la Figura 18.

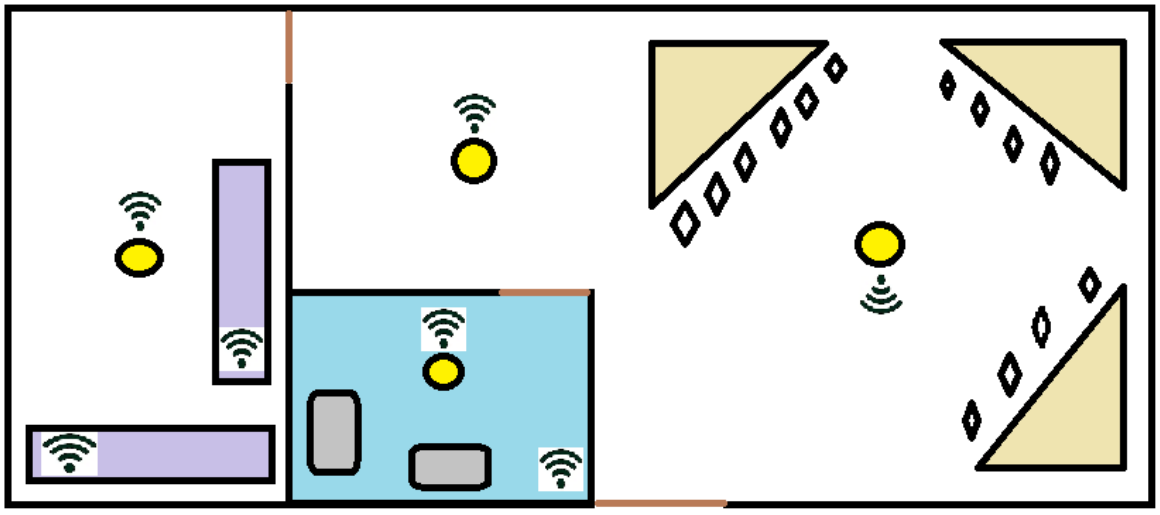

Figura 17 – Escenario caso práctico

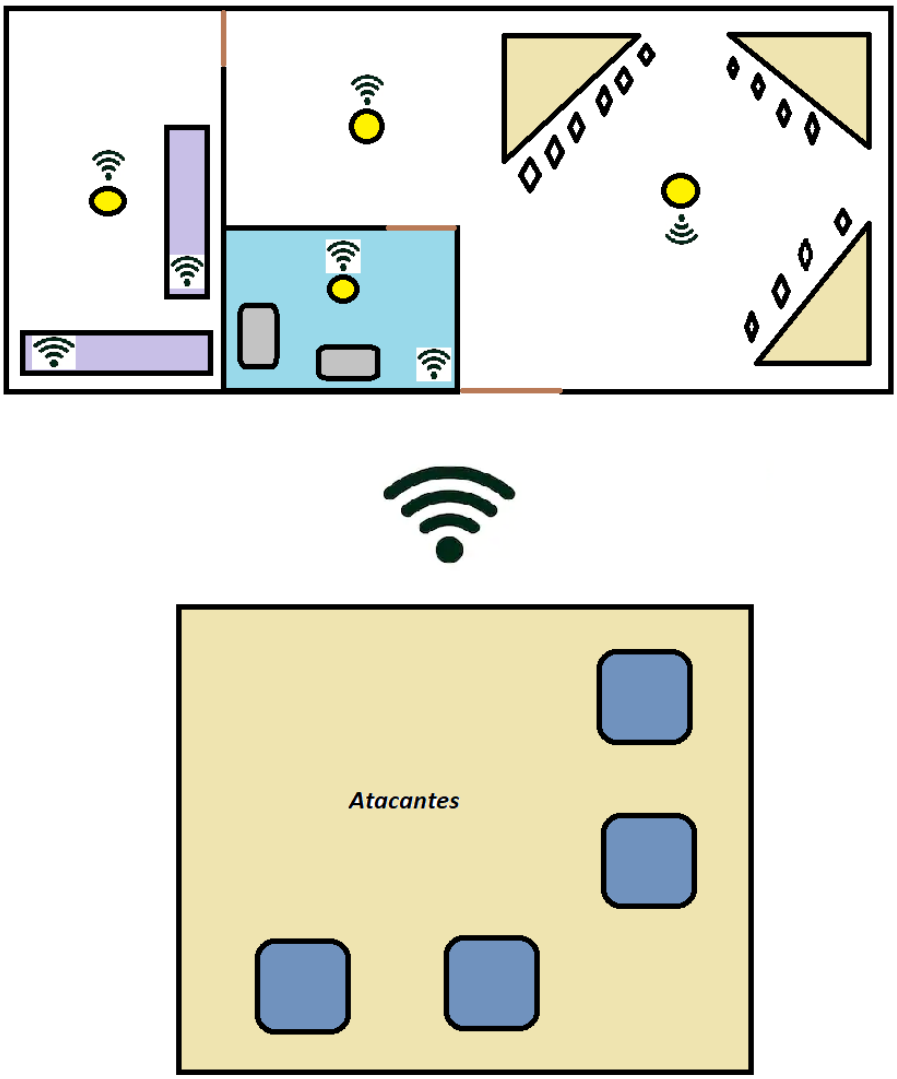

Figura 18 – Esquema caso ataque DDoS

5.3.1. Instalación y configuración de brokers MQTT, LOIC

5.3.1.1. Mosquitto

La herramienta Mosquito posee dos formas de instalación, la primera descargar un .zip con la misma para luego descomprimirla e instalar, y la segunda mediante comandos tal como se muestra a continuación.

```
“sudo apt update”
“sudo apt upgrade”
“sudo apt install mosquitto”
```

Para la comprobación del correcto funcionamiento del broker Mosquitto, observar la figura 19, esta información nos permitirá monitorear si el broker llega a fallar durante las pruebas.

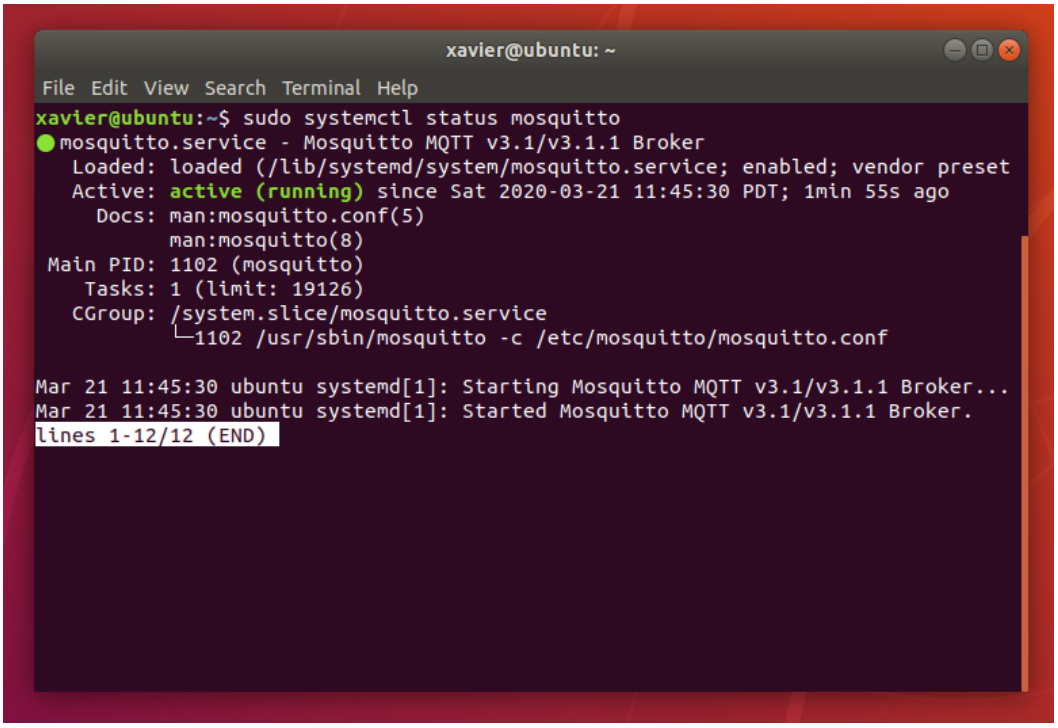

Figura 19 – Estatus broker Mosquitto

El archivo de configuración correspondiente está en el path “/etc/mosquitto/mosquitto.conf”:

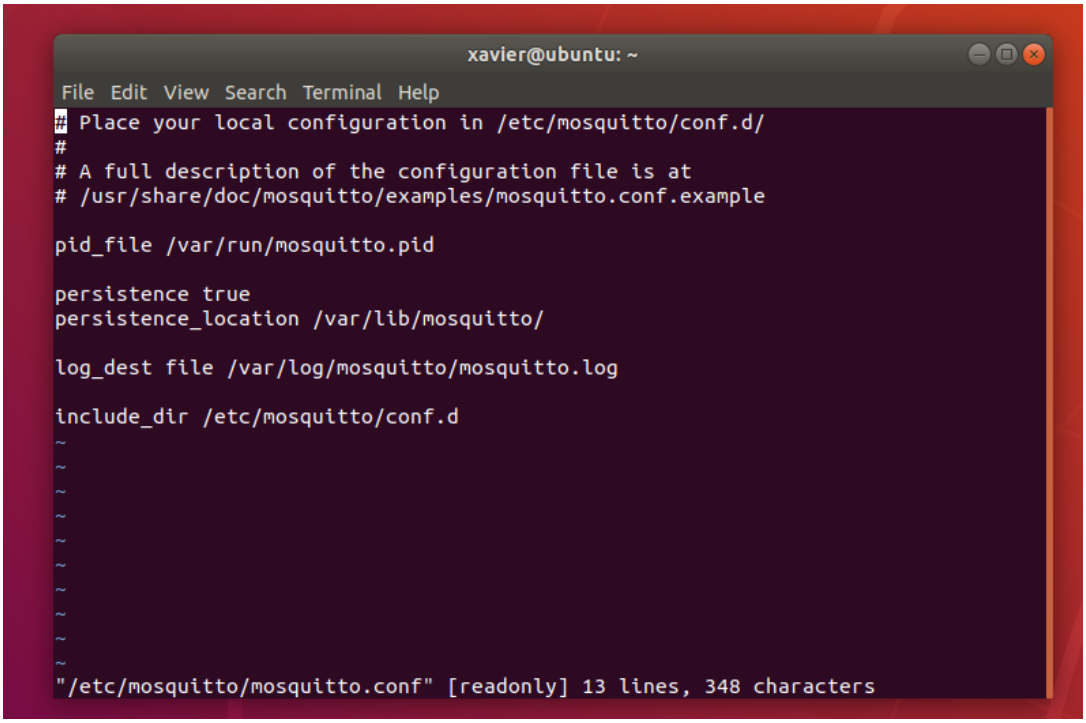

Figura 20 –Mosquitto.conf

Archivo de logs: “/var/log/mosquitto/mosquitto.log”:

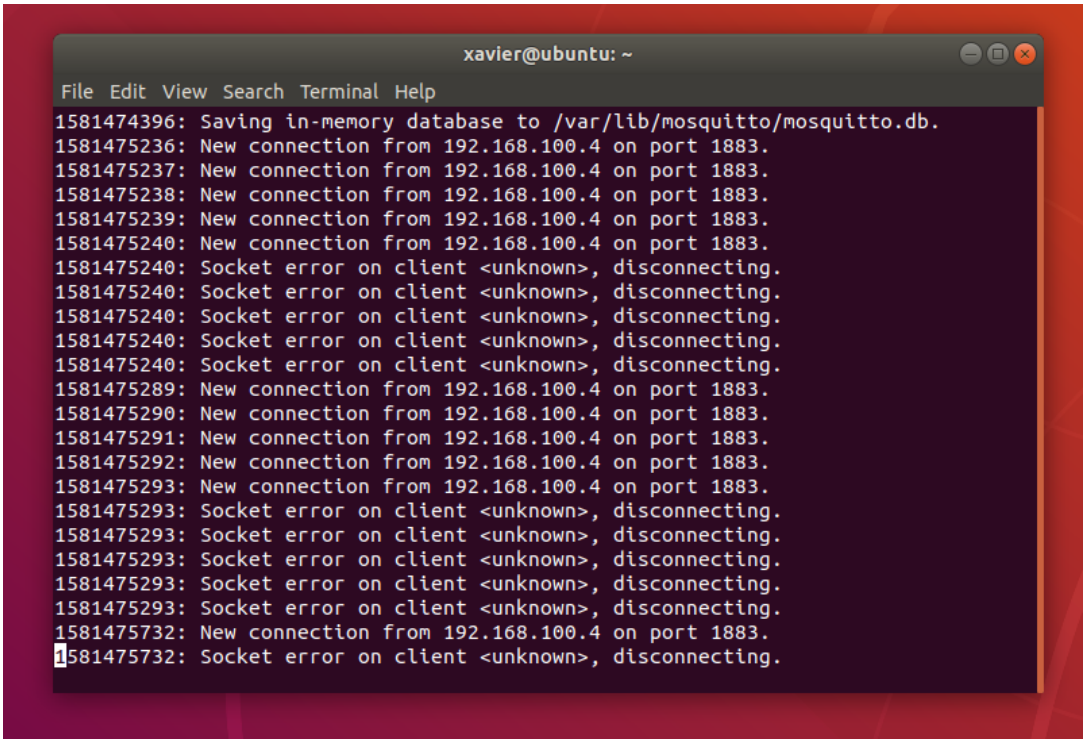

Figura 21 –Mosquitto.log

5.3.1.2. Mosca

Mosca posee dos formas de instalación: una a través de *Node-red* y otra a través de líneas de comandos, en este caso se instaló por medio de *Node-red*, para lo cual debemos dirigirnos a la opción *manage pallets*, luego en *install* y buscar Mosca:

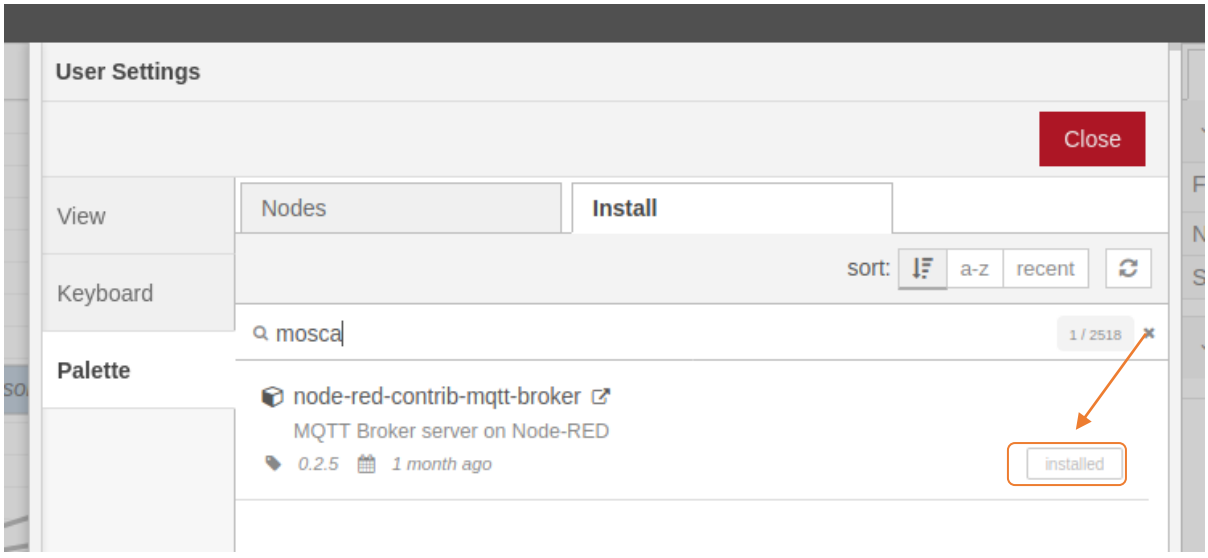

Figura 22 – Instalación Mosca

Por medio del comando “*mosca status -v*” podemos observar porque puerto está corriendo el servidor si está inicializado:

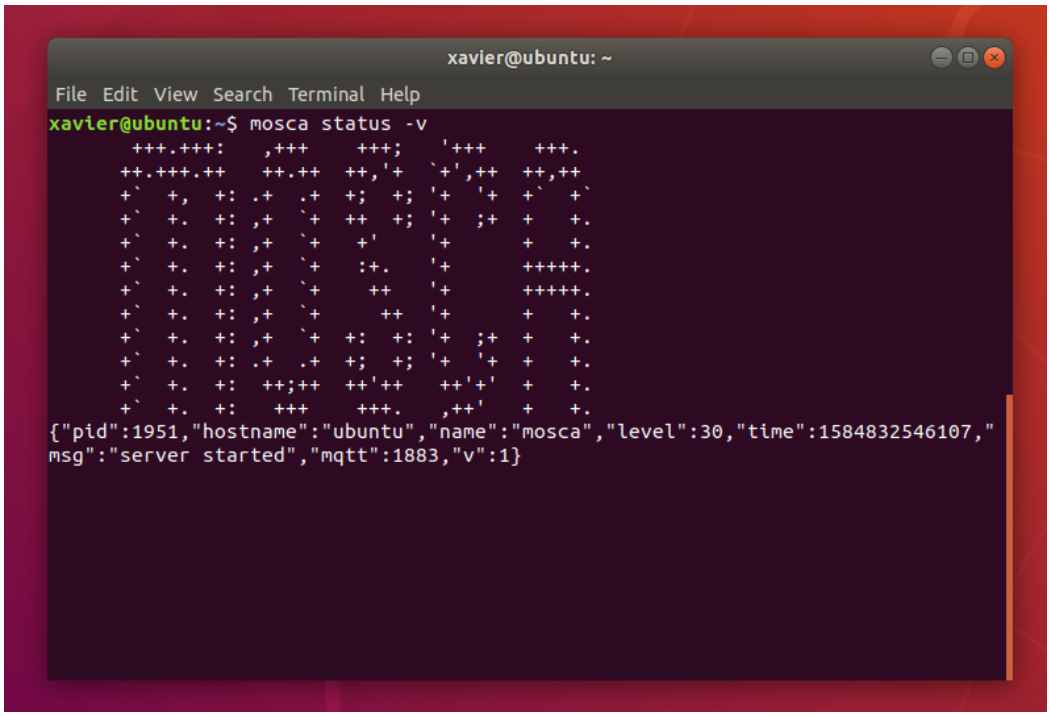

Figura 23 – Estatus Mosca

5.3.1.3. EMQ

EMQ al igual que Mosquitto posee dos formas de instalación: por descarga de un archivo .zip o .deb o por medio de líneas de comando, en este caso se instaló por medio de líneas de comando:

“sudo apt-cache Madison emqx”
“sudo apt install emqx=3.1.0”

Para iniciar y corroborar el estatus de nuestro broker se utiliza las siguientes líneas de comando “emqx start”, “emqx_ctl status”.

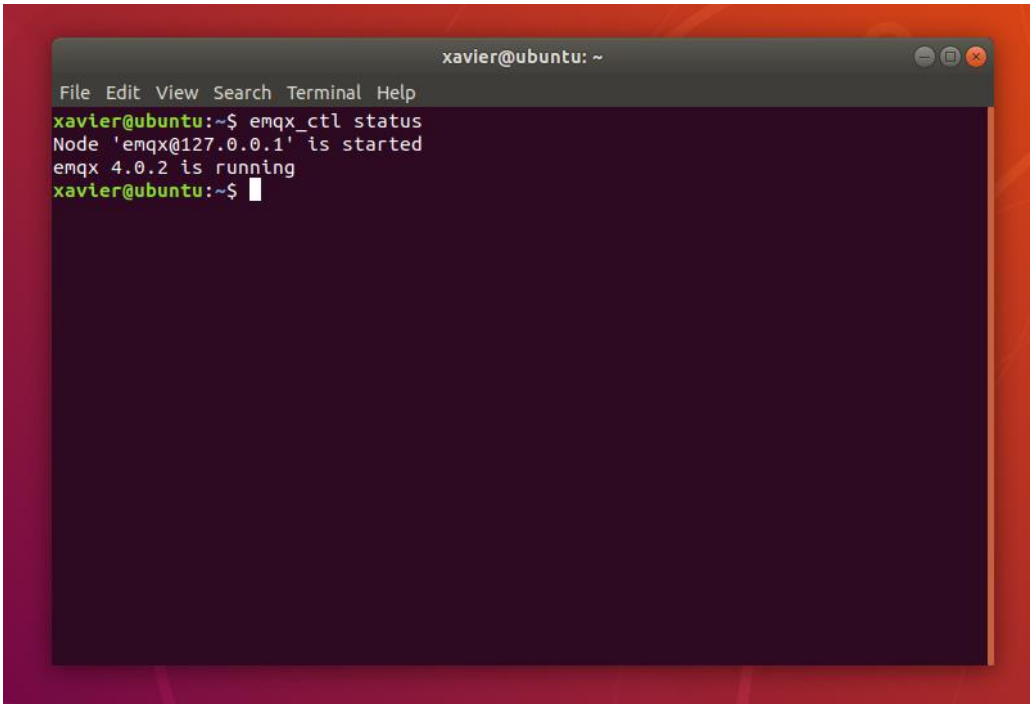

Figura 24 – Estatus EMQ

Para configuración del broker EMQ, se tiene un panel para un control ordenado de las conexiones dadas entre otras:

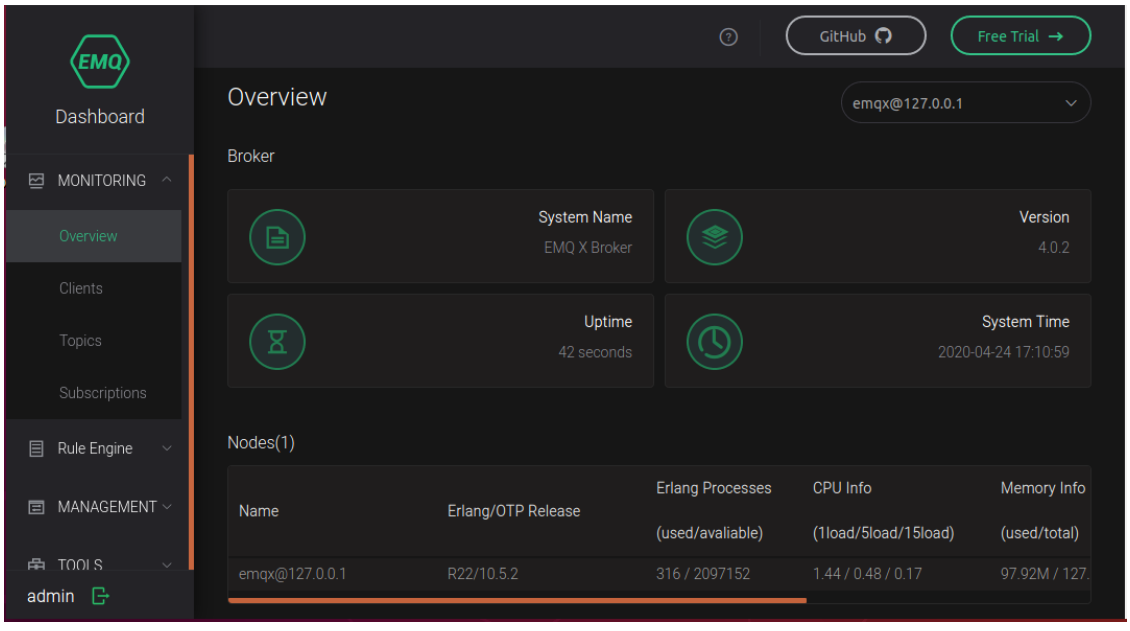

Figura 25 – *Dashboard* EMQ

5.3.2. Instalación herramientas de ataque DDoS

Para la realización de los ataques maliciosos de denegación de servicios, es necesario la instalación de 2 herramientas, en este caso LOIC y Metasploit:

LOIC es una herramienta que se debe descargar con cautela teniendo siempre nuestro antivirus en perfecto funcionamiento ya que no son oficiales y son de libre uso y estas pueden venir con virus escondidos y causar daño a nuestro equipo.

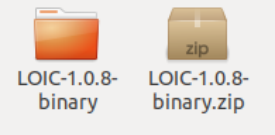

Figura 26 – LOIC.zip

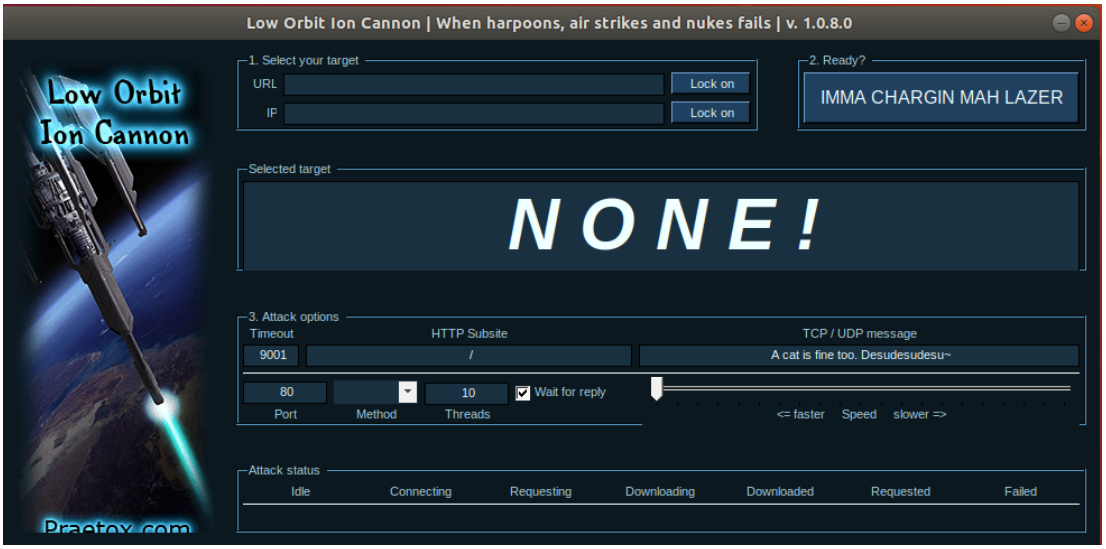

Figura 27 – LOIC.exe

La inicialización de Metasploit en Kali Linux para los ataques de tipo *SYN FLOOD*

Figura 28 – Kali Linux con Metasploit

5.4. Configuración del ataque con las herramientas Metasploit y LOIC

Para la realización de las infraestructuras IoT como pruebas de concepto se utilizará Node-Red en su versión 1.0.3

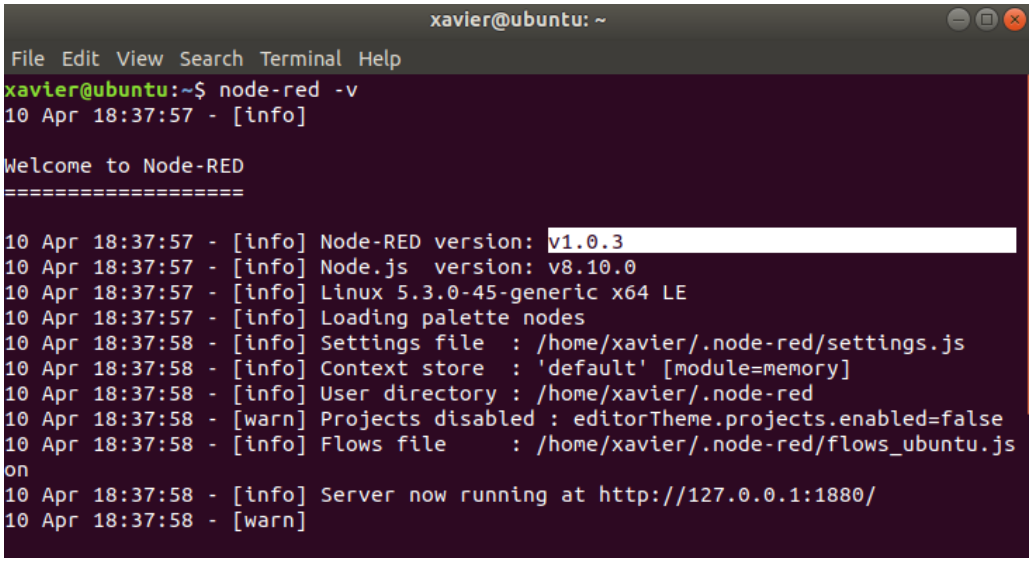

Figura 29 – *Node-Red* v1.0.3.

El esquema de caso práctico ya diseñado se lo realizó con cada uno nuestros brokers (Figura 30,31,32) ya instalados y configurados anteriormente.

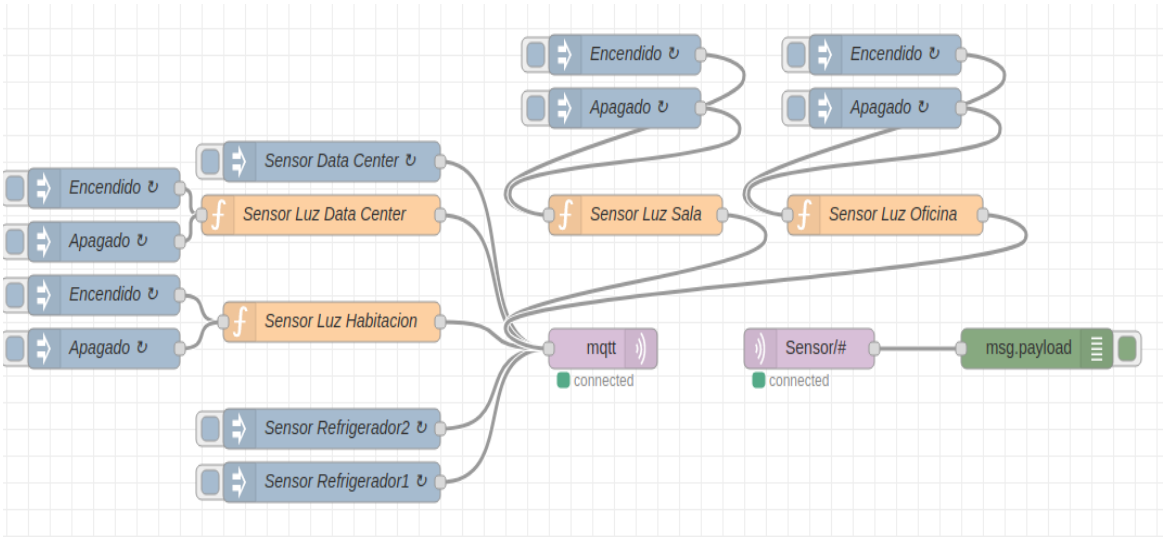

Figura 30 – Prueba Sensores broker Mosquitto con *Node-Red*.

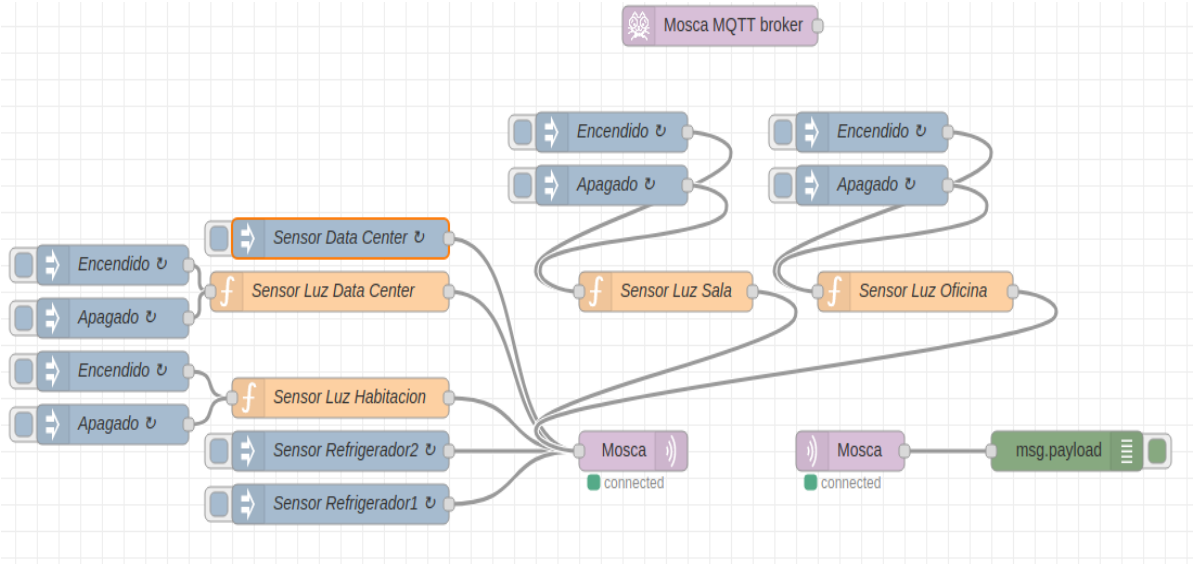

Figura 31 – Prueba Sensores broker Mosca con *Node-Red*

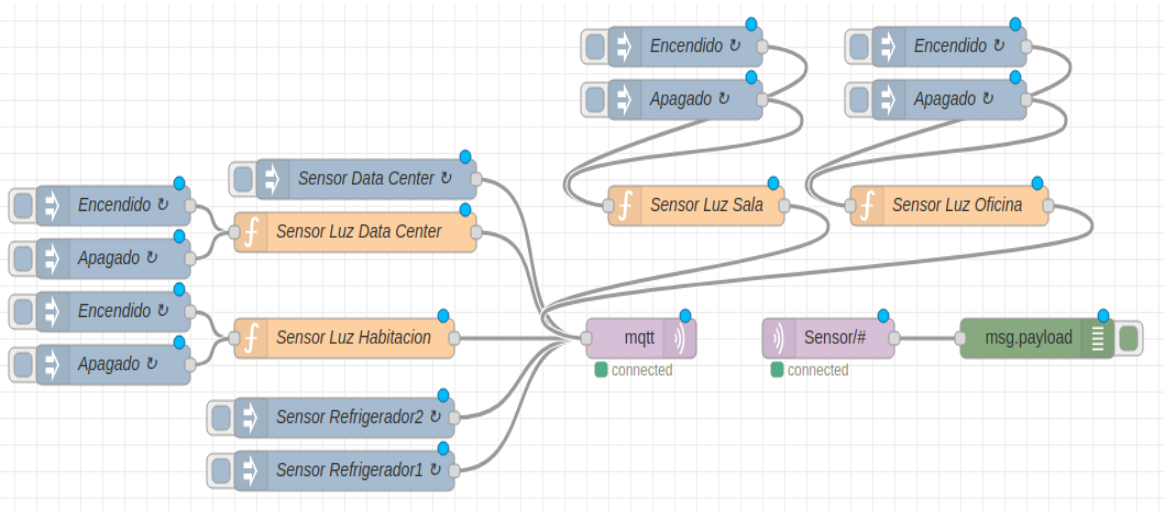

Figura 32 – Prueba Sensores broker EMQ con *Node-Red*

Con los esquemas ya diseñados y comprobado su funcionamiento, se procede a la configuración tanto del ataque con la herramienta Metasploit y LOIC.

Para la configuración de ataque en Metasploit para ataques de tipo SYN-FLOOD se utilizará la siguiente línea de comando “use auxiliary/dos/tcp/synflood”:

```
set RHOST “IP víctima”
set RPORT “Puerto MQTT”
set TIMEOUT ---
run
```

```
       =[ metasploit v5.0.73-dev-                         ]
+ -- --=[ 1962 exploits - 1095 auxiliary - 336 post        ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops             ]
+ -- --=[ 7 evasion                                       ]

msf5 > use auxiliary/dos/tcp/synflood
msf5 auxiliary(dos/tcp/synflood) > set RHOSTS 192.168.100.7
RHOSTS => 192.168.100.7
msf5 auxiliary(dos/tcp/synflood) > set RPORT 1883
RPORT => 1883
msf5 auxiliary(dos/tcp/synflood) > set TIMEOUT 25
TIMEOUT => 25
msf5 auxiliary(dos/tcp/synflood) > run
```

Figura 33 – Ataque DDoS SYN-Flood

Para la realización del ataque con la Herramienta LOIC es necesario solo la IP de las máquinas víctimas, en este caso de cada uno de los brokers.

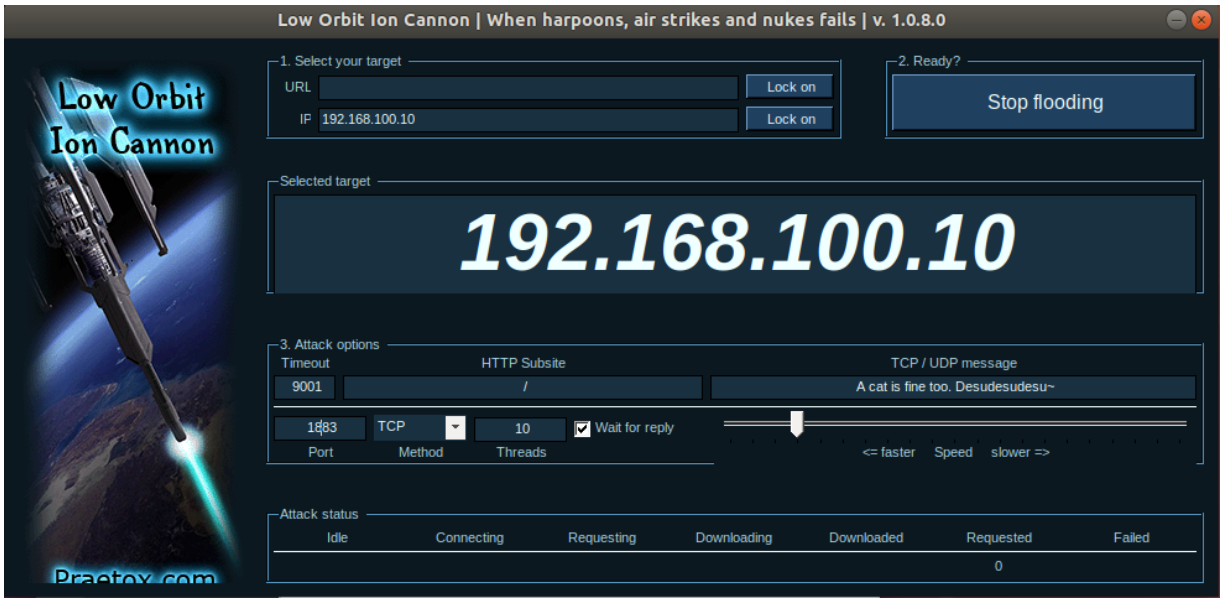

Figura 34 – Ataque DDoS LOIC

6. RESULTADOS

Resultados de la herramienta LOIC

Durante la realización de los ataques por medio de las cuatro máquinas maliciosas hacia cada uno de los brokers MQTT por un tiempo de seis horas que fueron sometidos con la herramienta LOIC no se pudo observar algún tipo de incidencia que pueda afectar al rendimiento de cada uno de ellos con respecto a la disponibilidad. Se ha evaluado el riesgo en caso de existir una eventual incidencia en cada uno de los brokers medido en un semáforo por lo que se obtiene los siguientes resultados:

Bajo Riesgo	
Medio Riesgo	
Alto Riesgo	

Semáforo de Riesgo

Tabla 5.- Comparativa KPIS Mosquitto, EMQ, Mosca, con LOIC

	MOSQUITTO	MOSCA	EMQ
MTTD	0 horas	0 horas	0 horas
MTTR	0 segundos	0 segundos	0 segundos
MTTF	-	-	-
MTBF	0 segundos	0 segundos	0 segundos

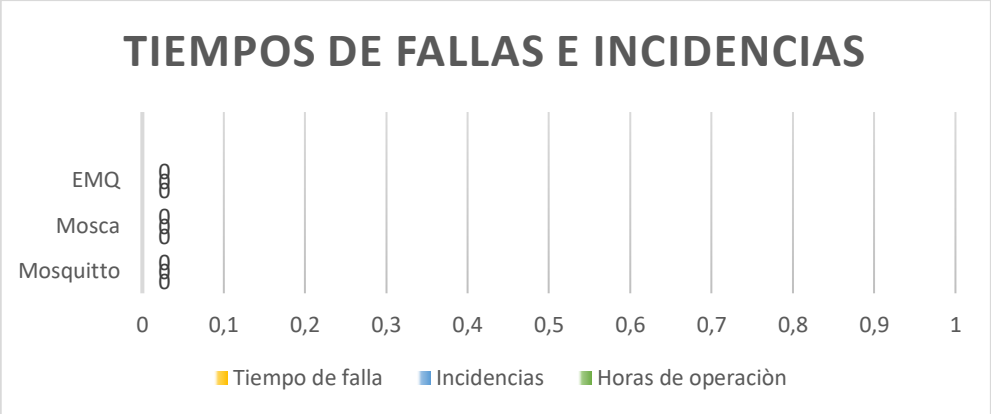

Figura 35 – Gráfico de barras de tiempo de caídas de Mosquitto, Mosca y EMQ con LOIC

En este caso los brokers se pueden decir que están 100% disponibles y que no existe ningún tipo de inconvenientes para la realización de peticiones hacia el Bróker.

Resultados de la herramienta METASPLOIT

En nuestro segundo caso de ataques maliciosos se utilizó Metasploit para el tipo de ataque SYN-FLOOD, en lo que se pudo evidenciar lo siguiente:

- **Broker Mosquitto. –** Obtuvo la primera incidencia de pérdida de conexión con esta herramienta a los cincuenta y un minutos con cuarenta y dos segundos de haber iniciado el ataque.

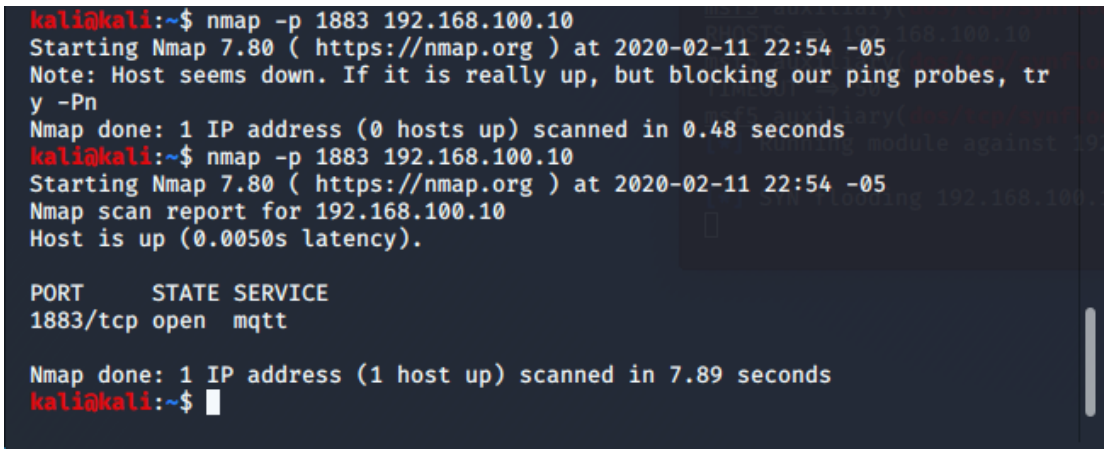

Figura 36 – Comprobación de caída del bróker Mosquitto mediante nmap

Se comprobó mediante nmap hacia la IP con el puerto de salida del broker MQTT Mosquitto 1883 en el cual se pudo evidenciar que al momento de caer pierde completamente la conexión y un segundo después esta vuelve a responder.

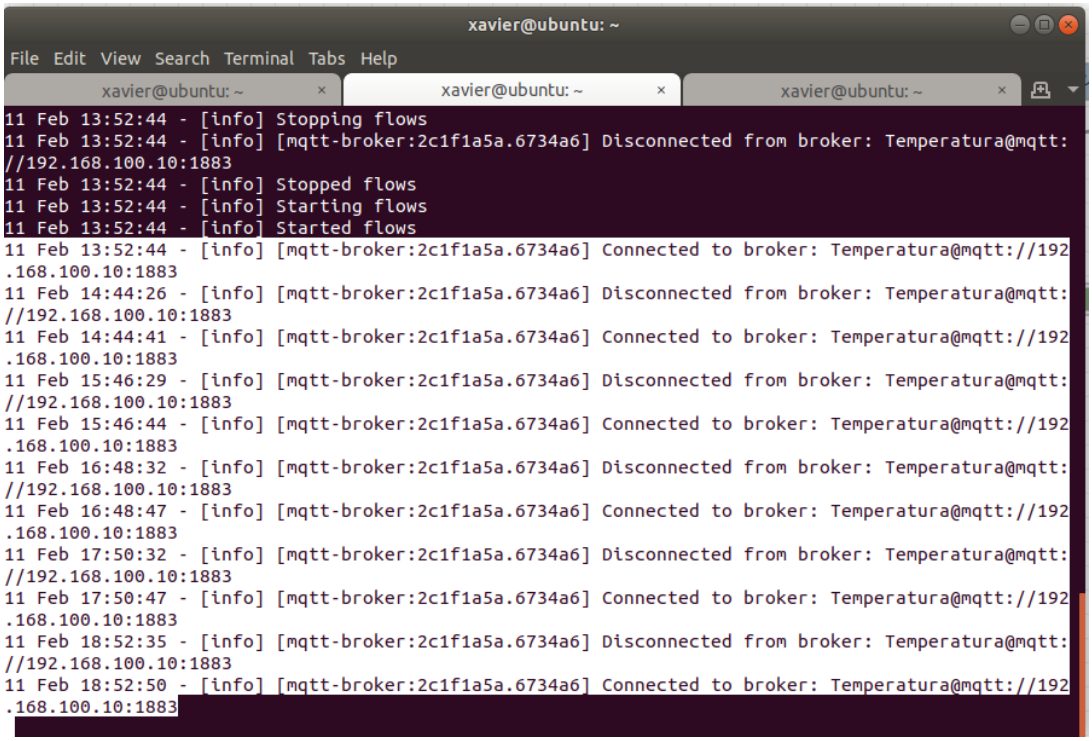

Figura 37 – Desconexión de Bróker Mosquitto en *Node-red*

En *wireshark* se puede evidenciar que toda petición o envío de información desde la infraestructura IoT hacia el broker MQTT Mosquitto cuando esta falla, la información que está en transmisión se pierde y no se recupera cuando esta se levanta, como se puede apreciar en el Anexo 5.

- **Bróker Mosca. -** Obtuvo su primera incidencia a los cuarenta y nueve minutos con seis segundos, pero en este caso se pudo evidenciar que en dos incidencias posteriores los tiempos de caída del bróker fueron de más de 30 segundos con lo cual la pérdida de información es considerablemente alta tomando en cuenta que *Node-red* le toma 15 segundos por defecto en restablecer la conexión en caso de una pérdida con este, en *wireshark* de igual manera se pudo evidenciar que se pierden los datos que hayan estado en transmisión y no se recuperan tal como se puede apreciar en el Anexo 6.

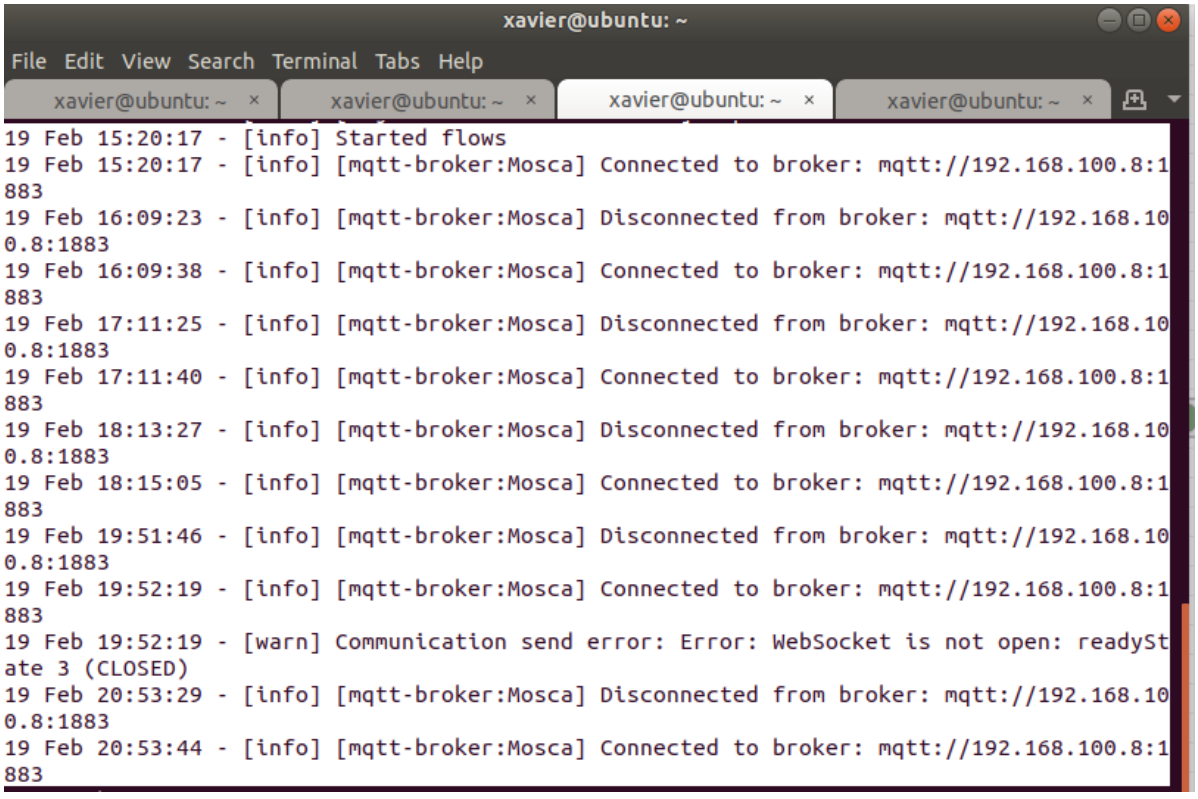

Figura 38 – Desconexión de Bróker MOSCA en *Node-red*

- **Broker EMQ. -** Obtuvo su primera incidencia a los cincuenta y cuatro minutos con veinte y uno segundos, además se pudo notar que al igual que Mosquitto la desconexión y conexión del bróker es menor a un segundo.

Al igual que los demás brokers analizados los datos en transmisión en ese momento no se recuperan como lo podemos ver en *wireshark* (Anexo 7), como punto principal es que el número de incidencias detectadas fue menor que en los demás casos prácticos.

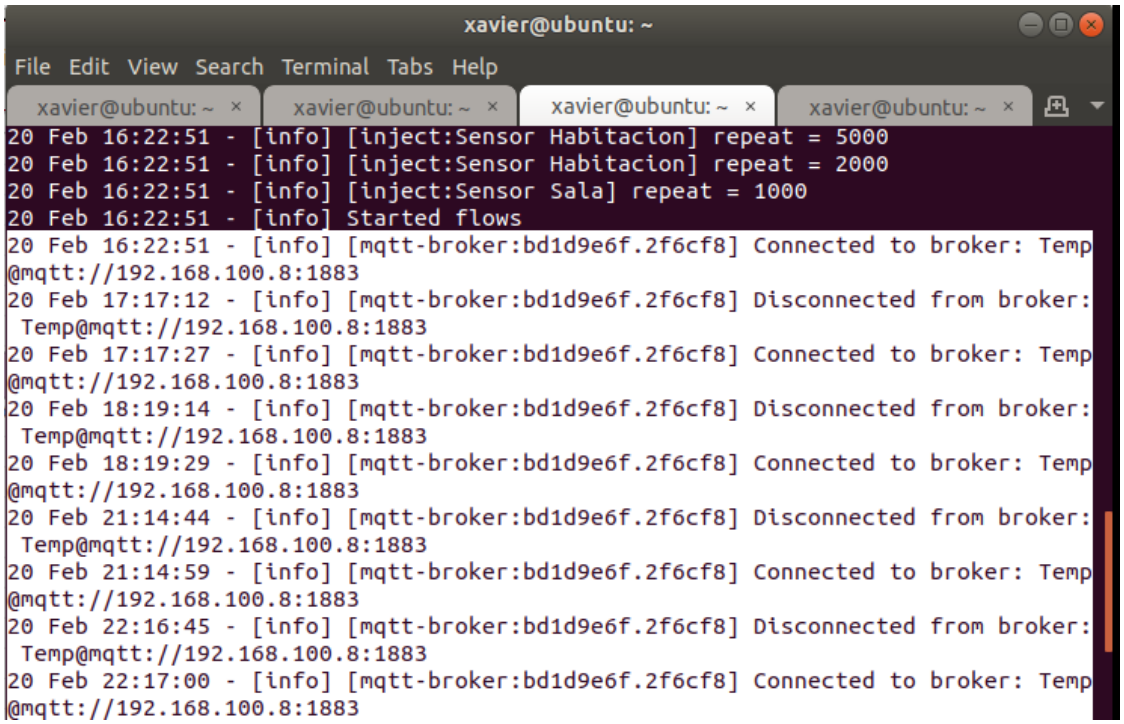

Figura 39 – Desconexión de Bróker EMQ en *Node-red*

Como se muestra en las Figuras 37,38 y 39, son el número total de fallas presentadas por cada uno de los brokers analizados. De igual manera se evaluado el riesgo mediante el usos de un semáforo dando como resultado los siguientes datos expresados en la siguiente tabla:

Bajo Riesgo	
Medio Riesgo	
Alto Riesgo	

Semáforo de Riesgo

Tabla 6.- Comparativa KPIS Mosquitto, EMQ, Mosca con Metasploit

	MOSQUITTO	**MOSCA**	**EMQ**
MTTD	5,58,45/5=1,11,45	5,57,4/5=1,11,25	5,59,0/4=1,29,45
MTTR	80segundos/5= 16 segundos	176segundos/5= 35,2 segundos	60 segundos/4= 15 segundos
MTTF	-	-	-

MTBF	6/5 = 1,2 Horas	6/5 = 1,2 Horas	6/4 = 1,5 Horas

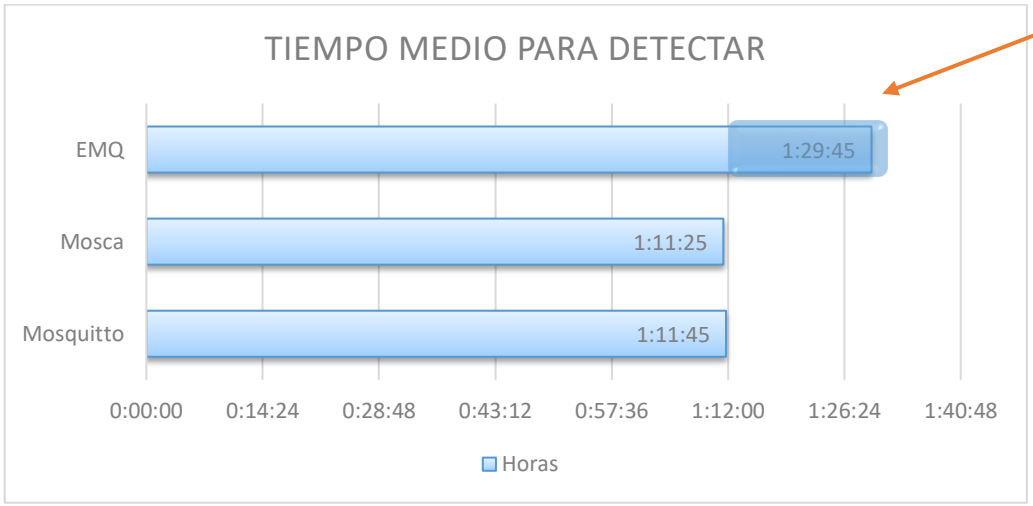

Figura 40 – MTTD

Siendo EMQ el broker con mayor tiempo promedio en demorarse en detectar una incidencia, le sigue Mosquitto que posee una ligera diferencia de 20 segundos con Mosca.

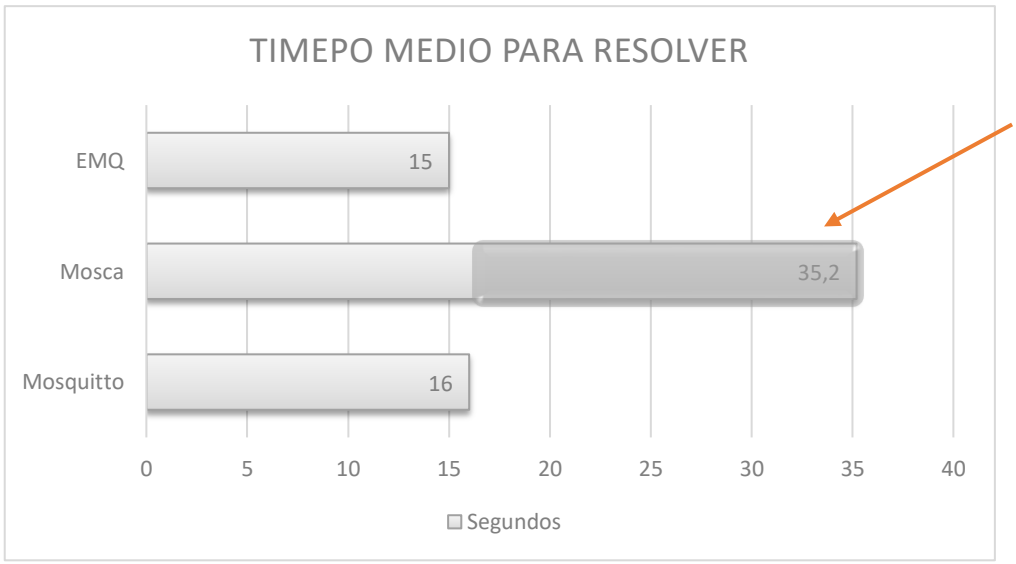

Figura 41 – MTTR

Mosca presenta el mayor tiempo para resolver una incidencia y tanto EMQ como Mosquitto están ligeramente iguales con 1 segundo de diferencia.

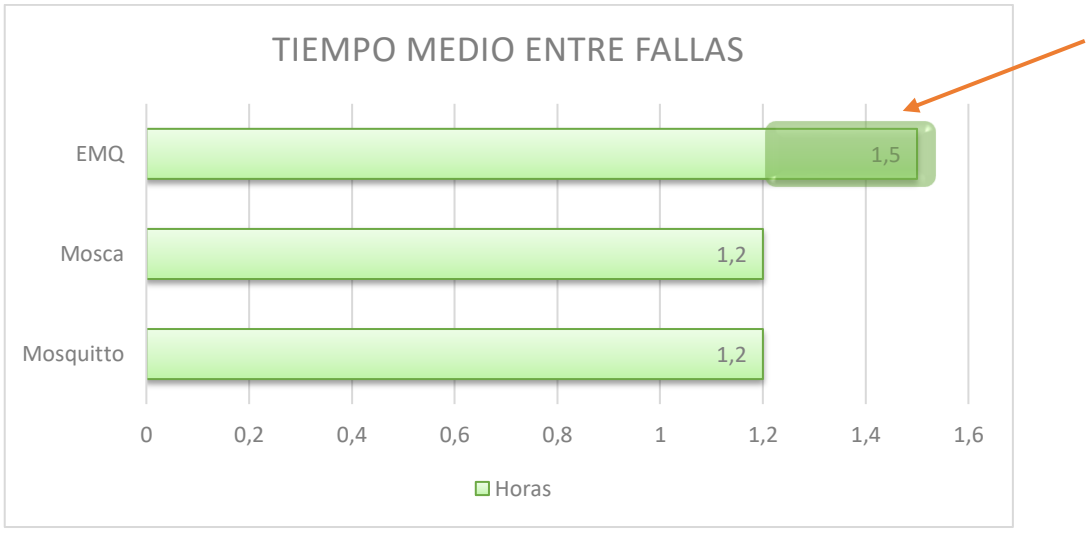

Figura 42 – MTBF

EMQ posee un tiempo medio entre fallas superior a Mosca y Mosquitto debido a que presenta un menor número de incidencias.

En este caso se puede decir que los brokers no son 100% disponibles debido a las incidencias presentadas en cada uno de los KPI's y cada una de las herramientas posee un porcentaje de disponibilidad relativamente alto por encima del 99%:

Se procedió hacer una regla de tres para calcular el porcentaje de disponibilidad para las seis horas expuestas ante ataques DDoS:

6 horas = 21600 segundos

X*100/21600=? (5)

- Mosquitto estuvo activo por 5:58:45 horas

(21525*100) / 21600 = 99,65 %

- Mosca estuvo activo por 5:57:4 horas

(21424*100) / 21600 = 99,19%

- EMQ estuvo activo por 5:59:0 horas

(21540*100) / 21600 = 99,72%

En base al desarrollo del proyecto de titulación se obtuvo los siguientes resultados:

Analizando los datos recolectados sobre cada uno de los brokers Mosquitto, Mosca y EMQ se pudo obtener la siguiente información relevante: Las tres herramientas son muy idénticas y cumplen al cien por ciento su objetivo principal que es la comunicación con el protocolo MQTT dentro de infraestructuras IoT.

- En el caso de Mosquito y EMQ son los que mejor se adaptan a trabajos con microprocesadores a pequeña y media escala, son flexibles y permiten un trabajo simplificado e intuitivo.
- EMQ es el broker más completo que se encuentra de forma teórica, ya que este permite un trabajo a gran escala, además acepta la incorporación o un trabajo en conjunto con Mosquitto sin ningún inconveniente.
- En el caso de Mosca es el broker que en base a los datos aún es muy limitada tanto en escalabilidad como en flexibilidad, tal vez en versiones futuras se vea mejoras en esta herramienta.

Adicionalmente, se tomó en consideración la usabilidad que en la actualidad poseen estas herramientas:

- Mosquitto es uno de los brokers más utilizados para infraestructuras IoT de pequeña y mediana escala gracias a su robustez y flexibilidad de trabajo en microprocesadores.
- Mosca es un bróker muy liviano que permite de igual forma instalarse de forma rápida en microprocesadores, pero es de bajo uso ya que su escalabilidad, información y soporte es limitada.
- EMQ es un broker muy completo teniendo como ventajas principales una conectividad todo en uno, robustez, escalabilidad, flexibilidad, control y monitoreo, como resultado de esto ha generado un gran impacto en el uso de esta herramienta.
- Como característica principal de los tres brokers analizados son muy seguros teóricamente y eso es lo que ofertan o promueven en cada una de sus fuentes oficiales y comunidades.

Durante el caso práctico se pudo evidenciar que en los dos escenarios planteados se obtuvo diferentes resultados:

- Con la herramienta LOIC no se pudo evidenciar algún tipo de inconveniente que puedan afectar la disponibilidad de los brokers, trabajando así con normalidad cada uno de ellos durante las seis horas de ataque.
- Con la herramienta Metasploit se pudo evidenciar diversos incidentes en cada uno de ellos siendo más notorio el caso de MOSCA.
    - En Mosquitto se pudo evidenciar 5 incidencias que afectaron el funcionamiento de este, en este caso se obtuvo que el tiempo medio para resolver es de 16 segundos restando los 15 segundos por defecto de *Node-red* para restablecerse tenemos un segundo de desconexión o pérdida. A

pesar de los inconvenientes su disponibilidad fue de 99.65% durante las seis horas de ataque.

- En Mosca de igual forma se pudo evidenciar 5 incidencias que afectaron el funcionamiento de este, en este caso el tiempo medio para reparar es de 35.2 segundos restando los 15 segundos de *Node-red* se obtiene 20.2 segundos de pérdida lo que es un índice muy alto para tomar en consideración. A pesar de todo esto su disponibilidad durante las seis horas fue 99.19%.
- En EMQ se pudo evidenciar un número menor de incidencias siendo un total de 4, teniendo un tiempo medio para resolver de 15 segundos restando los 15 segundos de *Node-red* se puede decir que la desconexión no supera el segundo por lo que la velocidad de resolver el problema es muy eficiente por parte de la herramienta. A pesar de esto obtiene un 99.72% de disponibilidad en las seis horas de ataque.
- En las tres herramientas se pudo descartar el KPI MTTF ya que ninguno llego a fallar en su totalidad y no necesito la intervención de una persona para resolver el problema.

7. DISCUSIÓN

Teóricamente, cada bróker posee ciertas similitudes como en calidad de servicio, versión de MQTT, flexibilidad para la instalación, funcionalidad en microprocesadores, entre otras. Pero entre todas estas se destaca de mejor manera el broker EMQ ya que posee una gran ventaja para la implementación en grandes infraestructuras, altas velocidades de comunicación entre nodos y un *dashboard* de monitoreo de las conexiones. Mosquitto y Mosca son unos excelentes brokers MQTT que de igual manera son de fácil uso e implementación, pero no es apto para infraestructuras a gran escala.

En el escenario planteado para prueba de conceptos se tiene:

- **Mosca**. – Durante las pruebas realizadas el broker mosca obtuvo los peores resultados, tomando en consideración que una incidencia duro 1 minuto y 38 segundos en poder restablecerse, además su tiempo promedio para detectar una incidencia es de 1 hora 11 minutos y 25 segundos.
- **Mosquitto**. – Posee una gran solvencia y reacción para levantarse de forma inmediata en caso de una incidencia de ataques DDoS tomándole aproximadamente 1 segundo para restablecerse, además se pudo evidenciar que obtuvo el mismo número de incidencias que Mosca y su tiempo promedio para detectar una incidencia es de 1 hora 11 minutos y 45 segundos.
- **EMQ**. – Posee una óptima solvencia y capacidad para reaccionar sobre ataques DDoS tomándole menos de 1 segundo para que esta se restablezca. Además, se pudo evidenciar que obtuvo menores incidencias durante las pruebas realizadas y los tiempos promedios para detectar una incidencia es de 1 hora con 29 minutos y 45 segundos.

Como aspecto muy importante se constató que ninguno de los tres brokers MQTT ha llegado a fallar de forma irreparable durante todas las pruebas realizadas, por tal motivo el KPI MTTF posee un valor nulo.

Otro aspecto importante es que, para el caso práctico realizado bajo el escenario planteado con la herramienta LOIC no se obtuvo alguna afectación en el funcionamiento con respecto a la disponibilidad de los tres broker.

Los ataques realizados por medio de la herramienta Metasploit con el ataque DDoS de tipo SYN-FLOD fueron de vital importancia para poder comprobar que en la actualidad las infraestructuras IoT basadas con el protocolo MQTT son seguras, pero no a un 100% y siempre se debe tomar en consideración que se debe aplicar todas las medidas de seguridad para evitar que sucedan incidencias que puedan afectar a las mismas.

Después de analizar todos los resultados arrojados en base a los KPI's de seguridad y las discusiones dadas se puede decir que el broker EMQ es óptimo y eficiente con respecto a los demás en base a la disponibilidad.

En base a los resultados y discusiones dadas se elabora las siguientes recomendaciones a tomar en consideración antes de elaborar una futura investigación que involucre este tipo de tecnología:

- Se recomienda que antes de utilizar un bróker MQTT dentro de infraestructuras IoT se debe evaluar bien el escenario donde se lo utilizará ya que existen diferentes herramientas que permiten el uso del protocolo MQTT que brindan diferentes características y funcionalidades que pueden ser aplicados en diferentes casos.
- Se recomienda que los ataques DDoS solo se realicen en lugares autorizados y dentro de ambientes controlados, ya sea para casos de estudio o investigación y no para causar daño a personas o empresas en las que se pueda generar perdidas representativas y que afecten la integridad de estas.
- Se recomienda que se debe tener un conocimiento básico sobre IoT para poder realizar diferentes escenarios, casos de estudio o simulaciones.
- Se recomienda conocer las características principales y funcionalidad básica del protocolo MQTT antes de su utilización, ya que esta puede ser un poco confusa y pueda causar inconvenientes durante su implementación.
- Se recomienda que en caso de realizar una simulación de ataques DDoS hacia un broker MQTT como el planteado en el proyecto de titulación, se posea una máquina física robusta, es decir que posea un buen procesador, memoria RAM de 16 Gb o superior y un buen sistema de refrigeración, ya que el tráfico de red que este tipo de ataque genera es muy alto y conlleva a que ocupe muchos recursos de procesamiento del CPU ocasionando su sobrecalentamiento que pueda dañar el equipo.
- Se recomienda para una futura investigación aumentar el número de publicadores, suscriptores y nodos conectados a una misma infraestructura IoT, simulando un escenario a gran escala en la que puedan existir un mayor número de peticiones por segundo y someterla ante ataques de DDoS para comprobar su funcionamiento en base a KPI's de seguridad.

8. CONCLUSIONES

En base al estudio bibliográfico realizado se analizó las características y funcionalidades principales de cada uno de los brokers MQTT Mosquitto, Mosca y EMQ, concluyendo que existen similitudes (son livianos ideales para uso en microprocesadores, etc.) entre cada una de ellas, destacándose EMQ, ya que brinda una mayor escalabilidad y flexibilidad para el manejo dentro de infraestructuras IoT. Cabe recalcar que todas las herramientas analizadas son muy buenas dentro de entornos pequeños y que pueden ser utilizados en proyectos de estudios, simulaciones o en casos reales como parqueaderos, casas inteligentes, etc.

Adicionalmente, durante la simulación de los ataques DDoS con las herramientas LOIC y Metasploit, hacia las tres infraestructuras IoT como prueba de concepto, se pudo determinar que estas no son 100% disponibles y se debe implementar todas las medidas de seguridad para mitigar o evitar que estos sucedan.

Las incidencias causadas por los ataques DDoS hacia los brokers MQTT (Mosquitto, Mosca y EMQ) ocasionan que la información que se encuentra en tránsito se pierda y esta no pueda ser recuperada al restablecer su conexión.

Con respecto a los KPI's MTTD, MTTR, MTTF y MTBF los tres brokers poseen resultados muy buenos con respecto a la disponibilidad; superando el 99%. Hay que tomar en consideración que la simulación realizada fue dentro de un entorno pequeño y en entornos más grandes u otros dominios de aplicación estos resultados pueden variar.

Finalmente, se concluye que el presente trabajo de titulación posee como aporte esencial brindar resultados sobre una estricta evaluación con respecto a la disponibilidad bajo ataques DDoS de brokers MQTT (Mosquitto, Mosca y EMQ), en base a las mediciones tomadas y que a su vez pueda ser utilizada en cualquier área de trabajo o investigación (salud, construcción, vivienda, ingeniería), se recomienda el uso la herramienta EMQ.

9. BIBLIOGRAFÍA

[1] G. D. Salazar, C. Venegas, M. Baca, I. Rodriguez, and L. Marrone, "Open Middleware proposal for IoT focused on Industry 4.0," *2018 IEEE 2nd Colomb. Conf. Robot. Autom. CCRA 2018*, 2018.

[2] G. Andrade-Salinas, G. Salazar-Chacon, and L. M. Vintimilla, "Integration of IoT Equipment as Transactional Endorsing Peers over a Hyperledger-Fabric Blockchain Network: Feasibility Study," *Commun. Comput. Inf. Sci.*, vol. 1193 CCIS, pp. 95–109, 2020.

[3] L. Tan and N. Wang, "Future Internet: The Internet of Things," *ICACTE 2010 - 2010 3rd Int. Conf. Adv. Comput. Theory Eng. Proc.*, vol. 5, pp. 376–380, 2010.

[4] A. Rahman, S. Roy, M. S. Kaiser, and M. S. Islam, "A lightweight multi-tier S-MQTT framework to secure communication between low-end IoT nodes," in *Proceedings of 2018 5th International Conference on Networking, Systems and Security, NSysS 2018*, 2019.

[5] P. P. Ray, "A survey on Internet of Things architectures," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 3. King Saud bin Abdulaziz University, pp. 291–319, 01-Jul-2018.

[6] G. D. Salazar, C. Hervas, E. Estevez, and L. Marrone, "High-level IoT governance model proposal for digitized ecosystems," *Proc. - 2019 Int. Conf. Inf. Syst. Softw. Technol. ICI2ST 2019*, vol. VI, no. 2, pp. 79–84, 2019.

[7] Kevin Asthon, "That ' Internet of Things ' Thing," *RFID J.*, p. 4986, 2010.

[8] M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, "Uninvited connections: A study of vulnerable devices on the internet of things (IoT)," *Proc. - 2014 IEEE Jt. Intell. Secur. Informatics Conf. JISIC 2014*, pp. 232–235, 2014.

[9] S. Deepika and P. Pandiaraja, "Ensuring CIA triad for user data using collaborative filtering mechanism," *2013 Int. Conf. Inf. Commun. Embed. Syst. ICICES 2013*, pp. 925–928, 2013.

[10] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," *Proc. - DARPA Inf. Surviv. Conf. Expo. DISCEX 2003*, vol. 1, pp. 303–314, 2003.

[11] K. Sonar and H. Upadhyay, "A Survey : DDOS Attack on Internet of Things," vol.

10, no. 11, pp. 58–63, 2014.

[12] CLOUDFLARE, “DDOS,” *Cloudflare, Inc.*, 2020. [Online]. Available: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/. [Accessed: 06-Jan-2020].

[13] B. Nagpal, P. Sharma, N. Chauhan, and A. Panesar, “DDoS tools: Classification, analysis and comparison,” *2015 Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2015*, no. February, pp. 342–346, 2015.

[14] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, “Secure MQTT for Internet of Things (IoT),” in *Proceedings - 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015*, 2015, pp. 746–751.

[15] M. Collina, G. E. Corazza, and A. Vanelli-Coralli, “Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST,” *IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC*, pp. 36–41, 2012.

[16] L. C.a *et al.*, “Securing smart maintenance services: Hardware-security and TLS for MQTT,” *Proceeding - 2015 IEEE Int. Conf. Ind. Informatics, INDIN 2015*, pp. 1243–1250, 2015.

[17] Y. Upadhyay, A. Borole, and D. Dileepan, “MQTT based secured home automation system,” *2016 Symp. Colossal Data Anal. Networking, CDAN 2016*, 2016.

[18] Eclipse Foundation, “Eclipse Mosquitto.” [Online]. Available: https://mosquitto.org/. [Accessed: 04-Dec-2019].

[19] M. Collina, “Mosca MQTT broker as a module.” [Online]. Available: https://www.mosca.io/. [Accessed: 02-Dec-2019].

[20] Technologies Co, “EMQ,” 2020. [Online]. Available: https://www.emqx.io/. [Accessed: 04-Jan-2020].

[21] A. Rajalakshmi and H. Shahnasser, “Internet of things using node-red and alexa,” *2017 17th Int. Symp. Commun. Inf. Technol. Isc. 2017*, vol. 2018-Janua, pp. 1–4, 2017.

[22] M. Lekić and G. Gardašević, “IoT sensor integration to Node-RED platform,” *2018 17th Int. Symp. INFOTEH-JAHORINA, INFOTEH 2018 - Proc.*, vol. 2018-Janua, no. March, pp. 1–5, 2018.

[23] P. Souza, W. Marques, R. Reis, and T. Ferreto, “Iagree: Infrastructure-agnostic Resilience Benchmark Tool for Cloud Native Platforms,” *CLOSER 2019 - Proc. 9th Int. Conf. Cloud Comput. Serv. Sci.*, no. Closer, pp. 396–403, 2019.

[24] S. S. Patel, *From the trenches*, vol. 69, no. 6. 2016.

[25] graciela Tono, "LA UTILIZACION DEL METODO COMPARATIVO EN ESTUDIOS CUALITATIVOS EN CIENCIA POLITICA Y CIENCIAS SOCIALES : diseño y desarrollo de una tesis doctoral," *Kairos Rev. temas Soc.*, vol. 27, pp. 1–12, 2011.

[26] Y. D. E. De, "Método comparativo."

[27] P. Letelier and C. Penadés, "Masyxp.Pdf," *Métodologías ágiles para el Desarro. Softw. Extrem. Program.*, 2017.

[28] J. F. González, "Introducción a las metodologías ágiles Otras formas de analizar y desarrollar," *Cataluña*, pp. 1–56, Creacion: 2013; Recuperado: 1 Febrero 2016, 2013.

[29] D. de L. C. Gómez and de la G. E. A. De León, "Carlos_Gomez_Diaz_Cap11_Metodocomparativo_Pdf.Pdf." 2014.

ANEXOS

Anexo I – Control de Tráfico con *WIRESHARK* EMQ

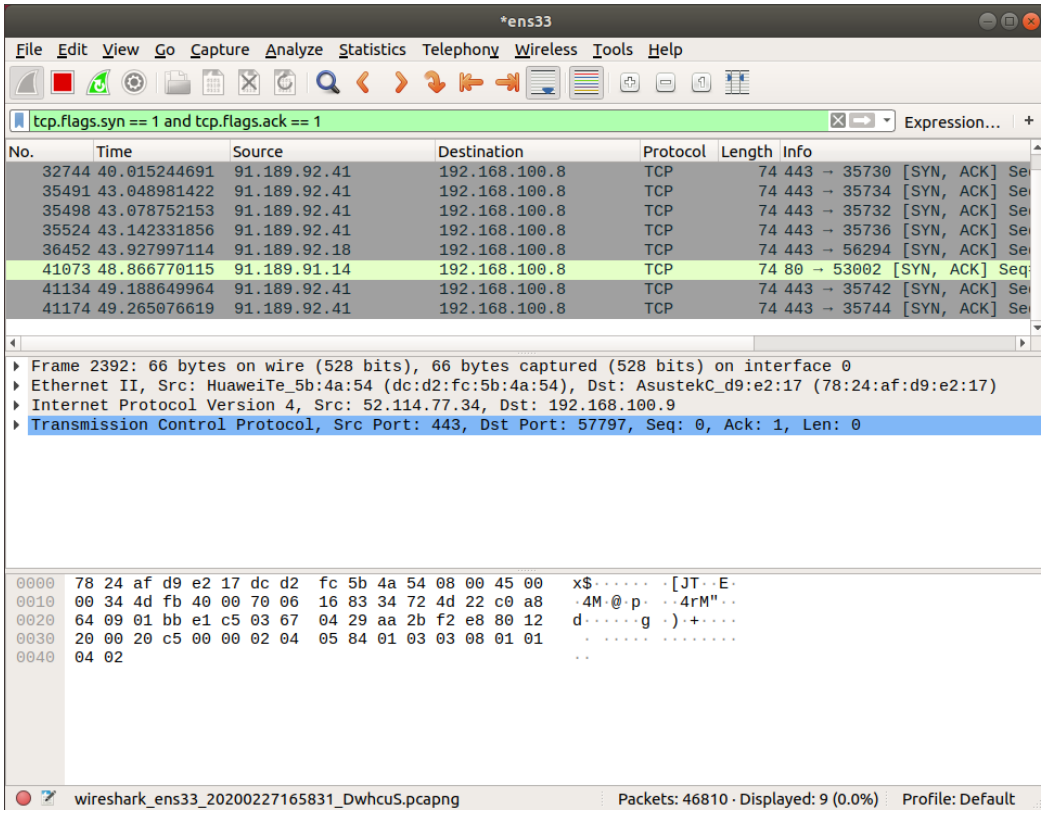

Figura 43 – Tráfico de red EMQ con *Wireshark*

Se muestra el panel de control y monitoreo mediante la herramienta *wireshark*, la que permite observar la operatividad del broker EMQ.

Anexo II – Control de Tráfico con *WIRESHARK* Mosquitto

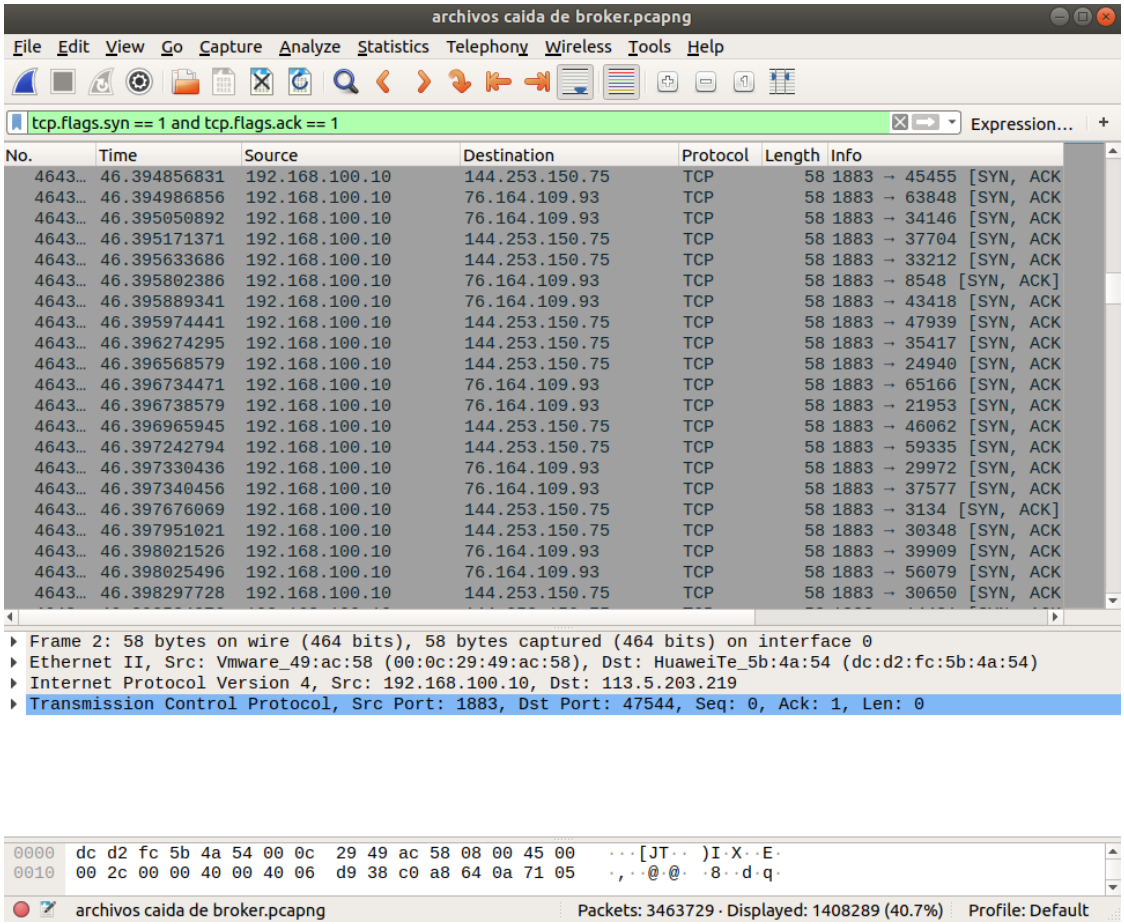

Figura 44 – Tráfico de red Mosquitto con *Wireshark*

Se muestra el panel de control y monitoreo mediante la herramienta *wireshark*, la que permite observar la operatividad del broker Mosquitto.

Anexo III – Control de Tráfico con *WIRESHARK* Mosca

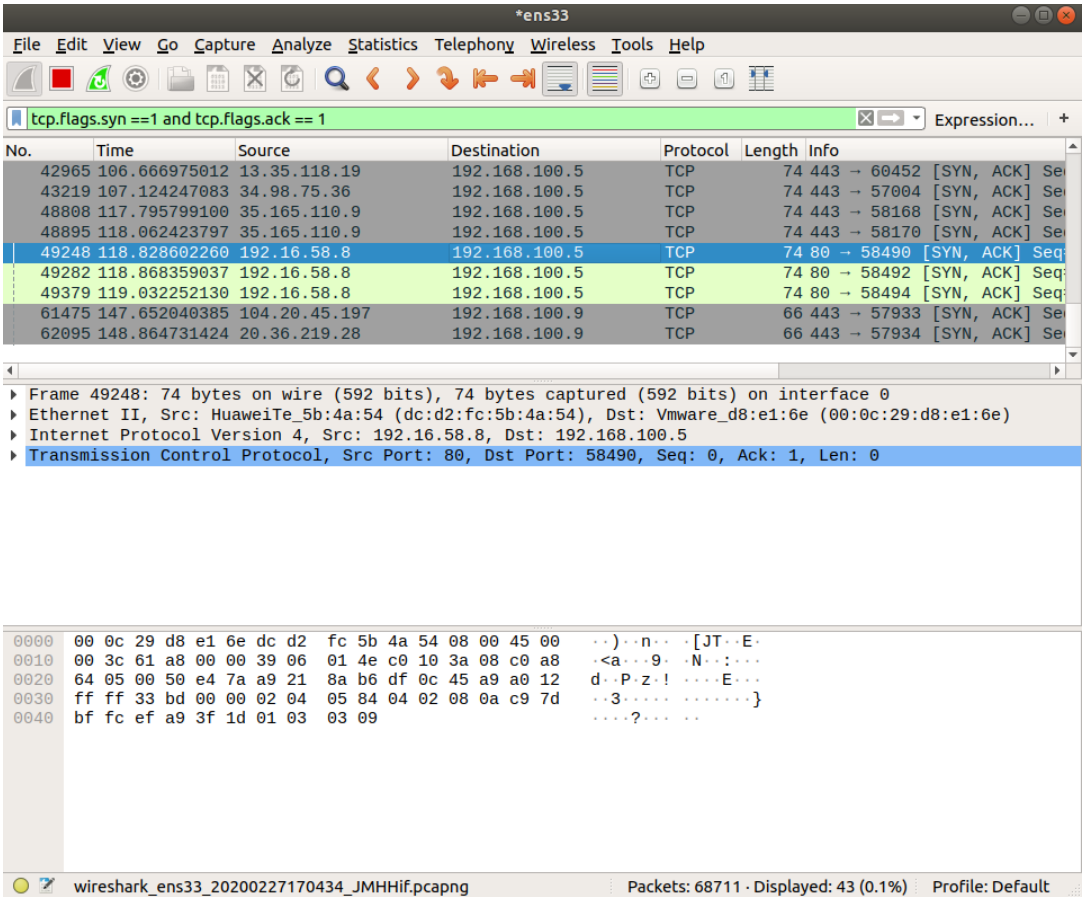

Figura 45 – Tráfico de red Mosca con *Wireshark*

Se muestra el panel de control y monitoreo mediante la herramienta *wireshark*, la que permite observar la operatividad del broker Mosca.

Anexo IV – Recepción de peticiones de los nodos

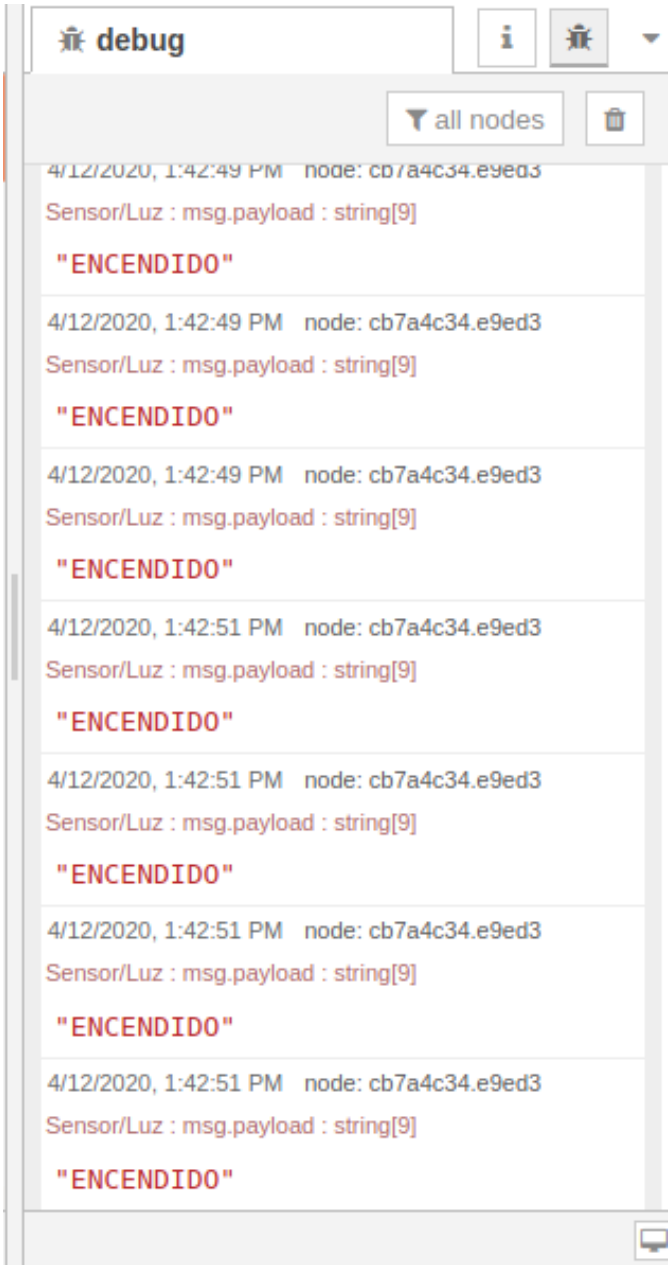

Figura 46 – Menú debug datos de envío de datos hacia el broker

Panel “Debug” de la herramienta *Node-Red* sobre el envío de la información por parte de los nodos conectados a la infraestructura IoT.

Anexo V – Caída del bróker Mosquitto

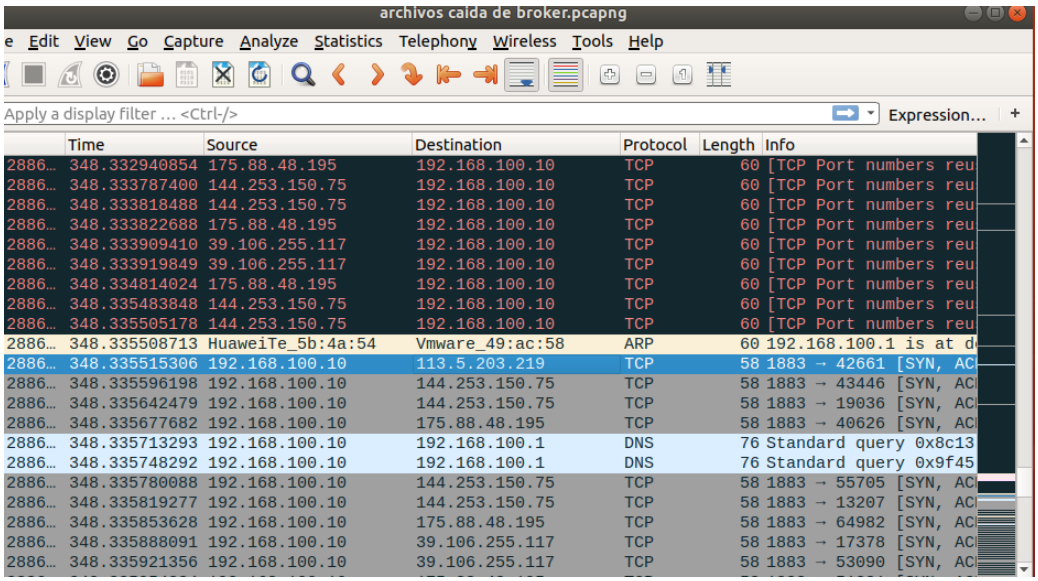

Figura 47 – Perdida de conexión con el broker Mosquitto

Mediante la herramienta *wireshark* se puede observar la pérdida de conexión con el broker Mosquitto y se pudo observar que sigue ingresando las peticiones por parte de las máquinas atacantes.

Anexo VI – Caída del bróker Mosca

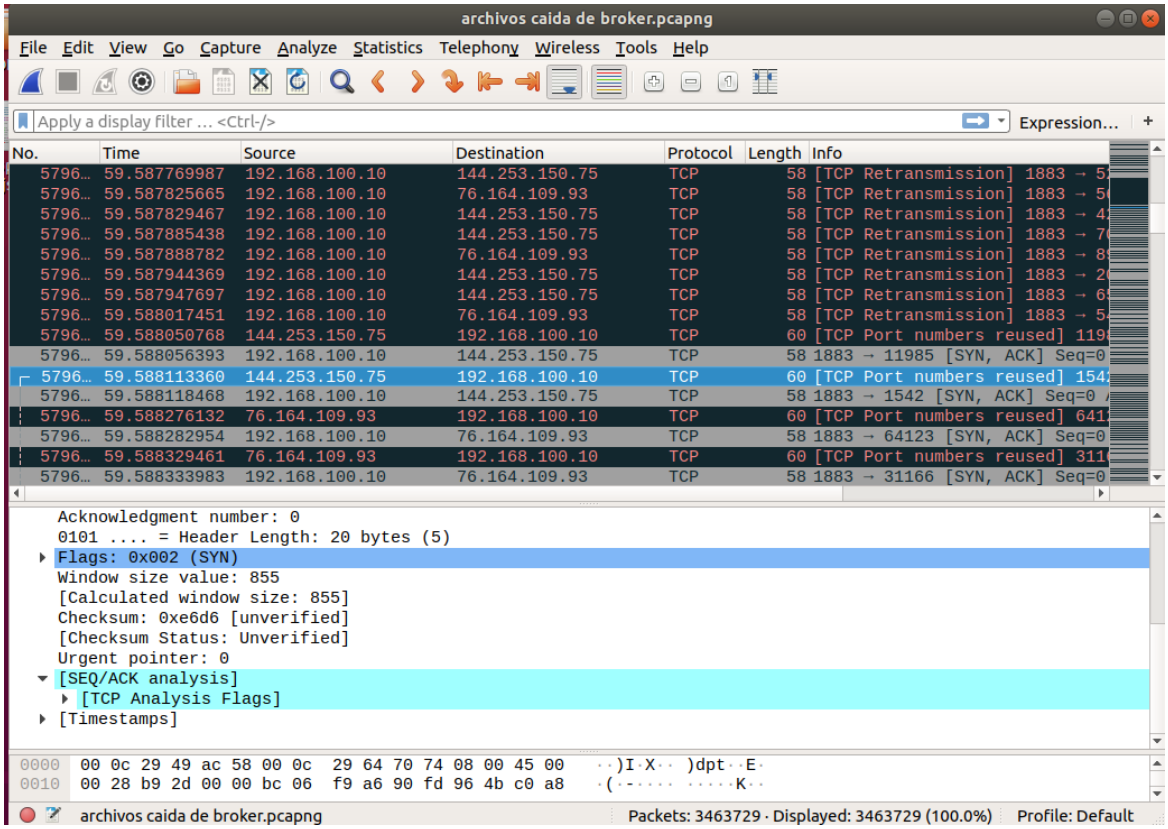

Figura 48 – Perdida de conexión con el broker Mosca

Mediante la herramienta *wireshark* se puede observar la pérdida de conexión con el broker Mosca y se pudo observar que sigue ingresando las peticiones por parte de las máquinas atacantes.

Anexo VII – Caída del bróker EMQ

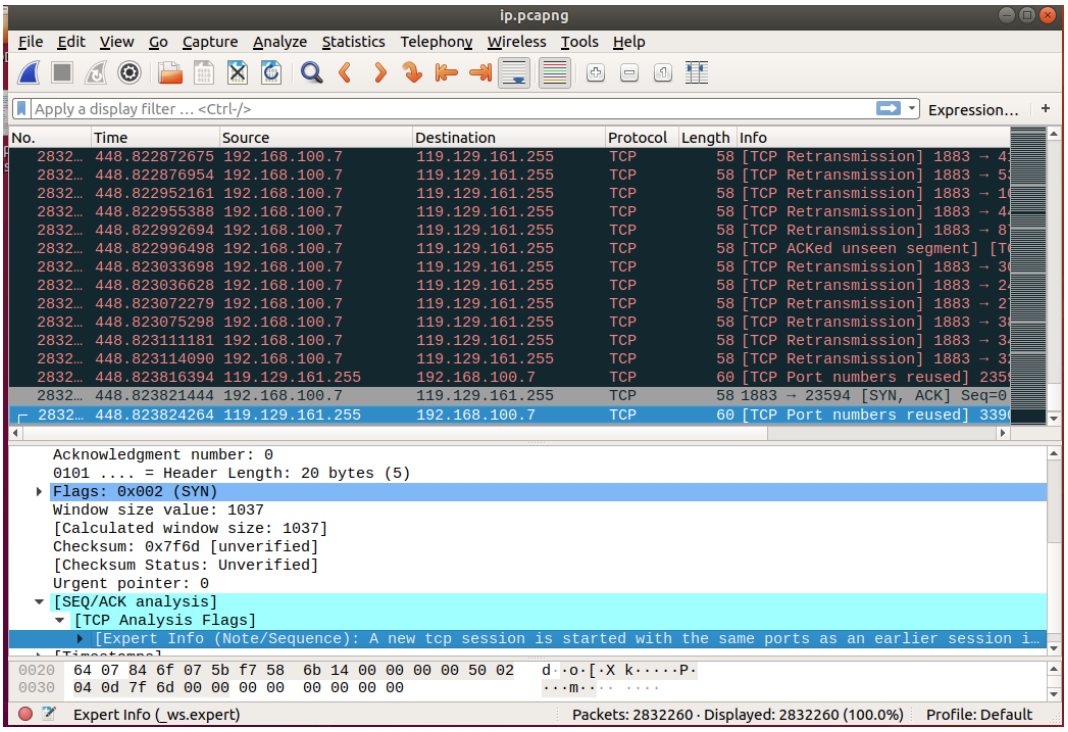

Figura 49 – Perdida de conexión con el broker EMQ

Mediante la herramienta *wireshark* se puede observar la pérdida de conexión con el broker EMQ y se pudo observar que sigue ingresando las peticiones por parte de las máquinas atacantes.

Anexo VIII – Configuraciones brokers *Node-Red*

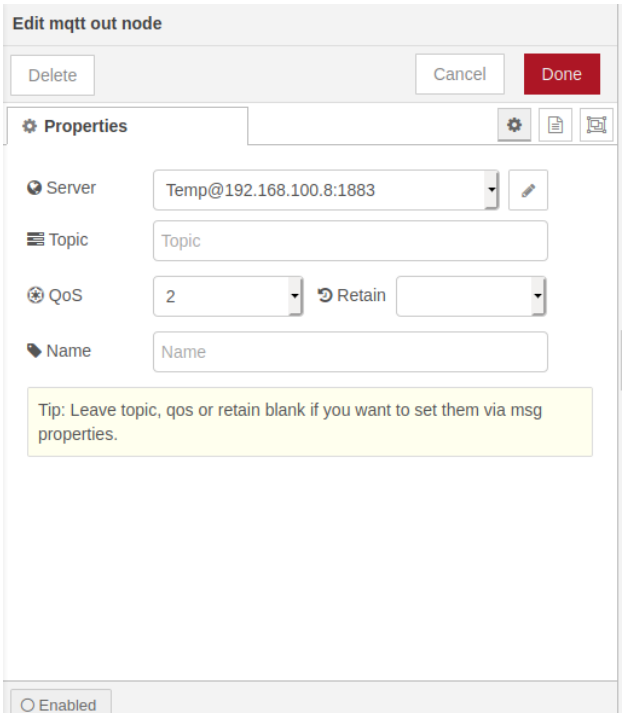

Figura 50 – Configuración nodo de salida

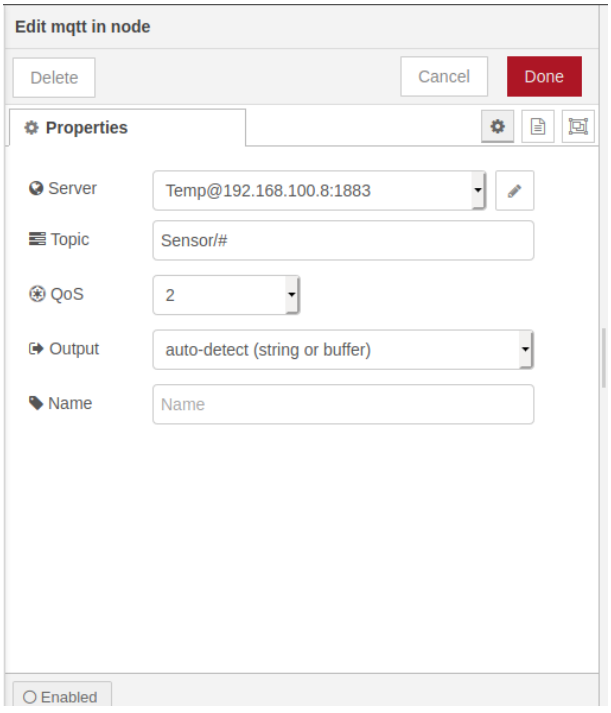

Figura 51 – Configuración nodo de entrada

Configuración de nodos MQTT en *Node-red* con la IP de la máquina virtual, del puerto del broker, Topic y QoS2

Anexo IX – Código Sensores de Luz

```
var foco=["ENCENDIDO","APAGADO"];
var mensaje=msg.topic;
var respuesta=mensaje.split("/");
var l=respuesta.length;
var message=msg.payload.toUpperCase();
if (respuesta[l-1]=="Oficina"){
    if (message==foco[0] || message==foco[1]){
        msg.payload=message;
        msg.topic=mensaje.replace("/Oficina","");
        return msg;
    }
    else{
    return null;
    }
}
return null;
```

```
var foco=["ENCENDIDO","APAGADO"];
var mensaje=msg.topic;
var respuesta=mensaje.split("/");
var l=respuesta.length;
var message=msg.payload.toUpperCase();
if (respuesta[l-1]=="Sala"){
    if (message==foco[0] || message==foco[1]){
        msg.payload=message;
        msg.topic=mensaje.replace("/Sala","");
        return msg;
    }
    else{
    return null;
    }
}
return null;
```

```
var foco=["ENCENDIDO","APAGADO"];
var mensaje=msg.topic;
var respuesta=mensaje.split("/");
var l=respuesta.length;
var message=msg.payload.toUpperCase();
if (respuesta[l-1]=="Habitacion"){
    if (message==foco[0] || message==foco[1]){
        msg.payload=message;
        msg.topic=mensaje.replace("/Habitacion","");
        return msg;
    }
    else{
    return null;
    }
}
return null;
```

Código de configuración de sensores de encendido y apagado de simulación de un foco en la herramienta *Node-red*, conectado a un nodo de la infraestructura IoT.