Maestría en Ciberseguridad

Trabajo previo a la obtención de título de

Magister de Ciberseguridad

AUTORES:

Cuenca Cuenca Diego Esteban

Males Anagumbla Alexis Israel

Muñoz Alarcón David Ricardo

Vera Jaramillo Kevin Estuardo

TUTORES:

Cortes Lopez Alejandro

Reyes Chacón Iván

TEMA:

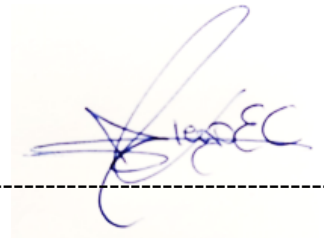
Sistema de monitoreo de seguridad con Splunk para IoT en PYMEs del Ecuador, integrando MQTT y SNMP

Quito – Ecuador

Certificación de autoría

Nosotros, Diego Esteban Cuenca Cuenca, Alexis Israel Males Anagumbla, David Ricardo Muñoz Alarcón y Kevin Estuardo Vera Jaramillo declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.



Firma del graduando

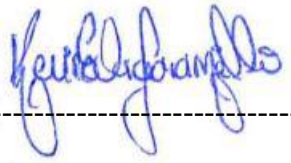
Diego Esteban Cuenca Cuenca

Firma del graduando

Alexis Israel Males Anagumbla

Firma del graduando

David Ricardo Muñoz Alarcón



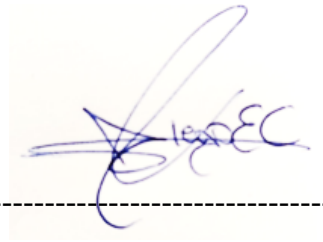
Firma del graduando

Kevin Estuardo Vera Jaramillo

Autorización de Derechos de Propiedad Intelectual

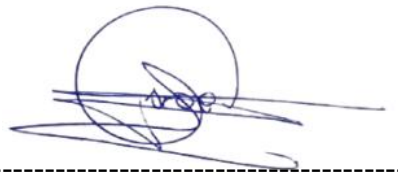
Nosotros, Diego Esteban Cuenca Cuenca, Alexis Israel Males Anagumbla, David Ricardo Muñoz Alarcón y Kevin Estuardo Vera Jaramillo en calidad de autores del trabajo de investigación titulado *Sistema de monitoreo de seguridad con Splunk para IoT en PYMEs del Ecuador, integrando MQTT y SNMP* autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, 08 de junio del 2025



Firma del graduando

Diego Esteban Cuenca Cuenca

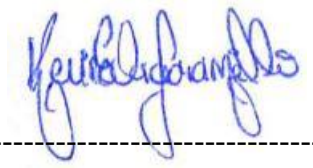


Firma del graduando

Alexis Israel Males Anagumbla

Firma del graduando

David Ricardo Muñoz Alarcón

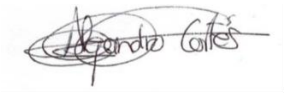


Firma del graduando

Kevin Estuardo Vera Jaramillo

Aprobación de dirección y coordinación del programa

Nosotros, Alejandro Cortés e Iván Reyes, declaramos que: Diego Esteban Cuenca Cuenca, Alexis Israel Males Anagumbla, David Ricardo Muñoz Alarcón y Kevin Estuardo Vera Jaramillo son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés L.
Maestría en Ciberseguridad

Iván Reyes Ch.
Maestría en Ciberseguridad

DEDICATORIAS

A toda mi familia, por su paciencia, amor y constante apoyo durante todo este tiempo de estudio, quienes fueron mi pilar fundamental para seguir adelante.

A los docentes, por su guía y enseñanza invaluable, y a mis compañeros, por compartir esta gran experiencia de aprendizaje que nos permitió alcanzar juntos la meta de culminar este proyecto.

Diego Cuenca

Dedico este trabajo de fin de maestría a mi familia, pilar fundamental en cada paso de mi vida. Su amor, paciencia y apoyo incondicional han sido la fuerza que me impulsó a seguir adelante. Con gratitud y profundo cariño, les dedico este esfuerzo culminado.

Alexis Males

Dedico este trabajo a mi esposa e hija, por su apoyo, confianza y constancia para seguir cosechando logros.

Kevin Vera

Este trabajo de fin de maestría lo dedico a mi familia y a mi compañera de vida, quienes han sido mi apoyo fundamental para no rendirme en cada paso que doy. Gracias por su cariño, paciencia y respaldo incondicional en los momentos buenos y malos que me han permitido terminar con éxito un desafío más y que sin dudas ha sido fuente de motivación y fortaleza para afrontar nuevos retos que el futuro me depare.

David Muñoz

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a la Universidad Internacional del Ecuador (UIDE) por brindarme la oportunidad de formarme académicamente y por el acompañamiento recibido durante este proceso.

A los docentes, por su compromiso, conocimiento y guía constante que fueron fundamentales para la realización de este trabajo.

A mis compañeros de maestría, por su colaboración y el espíritu de equipo que enriquecieron esta experiencia.

También extiendo mi gratitud a todas las personas e instituciones que, directa o indirectamente, contribuyeron al desarrollo de este proyecto.

Diego Cuenca

Agradezco a Dios por darme la fortaleza y sabiduría para culminar esta etapa. A mi familia, por su amor incondicional, comprensión y constante motivación.

A mis docentes y tutores, por compartir sus conocimientos y guiarme con paciencia. A mis compañeros de maestría, por el apoyo mutuo y los aprendizajes compartidos.

A mi institución, por brindarme las herramientas necesarias para mi formación. Este logro es el resultado del esfuerzo conjunto de todos ustedes. ¡Gracias!.

Alexis Males

Agradezco a Dios que me ha permitido contar con salud y sabiduría para tomar el sendero correcto.

Agradezco a todos los docentes por depositar su confianza y vocación en formar profesionales de alto valor.

Agradezco a mis compañeros por el esfuerzo dedicado para compartir experiencias y profesionalismo para culminar el objetivo.

Kevin Vera

Quiero agradecer a Dios por permitirme cumplir una meta más, por darme esa fortaleza y energía extra en los momentos difíciles y que siempre sé que me acompaña y me lleva por el buen camino.

Deseo expresar mi más profundo reconocimiento a la Universidad Internacional del Ecuador por la formación académica recibida y por el acompañamiento permanente a lo largo de esta etapa.

Extiendo mi gratitud a los profesores y tutores que formaron parte de este posgrado; gracias por compartir sus conocimientos con generosidad, paciencia y dedicación, permitiéndonos crecer en el aprendizaje y aplicarlo en nuestra vida profesional.

Y a mis compañeros de este posgrado, por su colaboración, respaldo mutuo y el compañerismo que enriquecieron esta etapa académica.

David Muñoz

RESUMEN

Este Trabajo de Fin de Maestría presenta el diseño de un sistema integral de monitoreo de seguridad basado en Splunk, diseñado específicamente para PYMEs en Ecuador, que aborda los desafíos de seguridad en entornos IoT mediante la integración de los protocolos MQTT y SNMP. La solución propuesta combina tecnologías de virtualización, análisis en tiempo real y protocolos ligeros para ofrecer una herramienta accesible y escalable. La implementación se realizó en un entorno simulado con GNS3, incorporando componentes clave como un firewall pfSense para protección perimetral, un switch Cisco L3 para gestión de tráfico, dispositivos IoT virtualizados incluyendo sensores de temperatura/humedad mediante MQTT y cámaras IP con Syslog, y un servidor Splunk encargado de recolectar, indexar y analizar los datos los registros de seguridad. La configuración de Splunk incluyó el HTTP Event Collector para recepción de datos MQTT, Syslog para logs de red y dispositivos IoT, y SNMP para monitoreo de infraestructura crítica. Los resultados demostraron capacidades avanzadas de detección proactiva, generando alertas automáticas por temperaturas críticas, caídas de interfaces e intentos de fuerza bruta, con notificaciones vía correo electrónico y paneles de control en Splunk. La solución destacó por su eficiencia en la visualización centralizada mediante dashboards interactivos que muestran en tiempo real el estado de sensores, interfaces de red y eventos clasificados por criticidad, facilitando la toma de decisiones. La integración de flujos de datos desde múltiples fuentes demostró ser robusta, permitiendo una correlación efectiva de eventos de seguridad. Entre los principales beneficios se encuentra el aprovechar software open-source como pfSense y Mosquitto junto con versiones gratuitas de Splunk, su escalabilidad para incorporar nuevos dispositivos y protocolos, y su capacidad para reducir los tiempos de respuesta ante incidentes mediante automatización. El proyecto valida que es posible implementar sistemas avanzados de monitoreo de seguridad en PYMEs con recursos limitados. La combinación de virtualización, protocolos optimizados para IoT y capacidades analíticas de Splunk ofrece un modelo sostenible para la protección de infraestructuras críticas contra amenazas cibernéticas crecientes. Los hallazgos recalcan la necesidad de monitoreo continuo en entornos con creciente adopción de dispositivos IoT.

Palabras Claves: Splunk, IoT, MQTT, SNMP, Ciberseguridad, PYMEs, Monitoreo en tiempo real, Virtualización.

ABSTRACT

This Master's Thesis presents the design of a comprehensive security monitoring system based on Splunk, specifically designed for SMEs in Ecuador, which addresses security challenges in IoT environments by integrating MQTT and SNMP protocols. The proposed solution combines virtualization technologies, real-time analysis, and lightweight protocols to provide an accessible and scalable tool. The implementation was carried out in a simulated environment with GNS3, incorporating key components such as a pfSense firewall for perimeter protection, a Cisco L3 switch for traffic management, virtualized IoT devices including temperature/humidity sensors via MQTT and IP cameras with Syslog, and a Splunk server responsible for collecting, indexing, and analyzing security log data. The Splunk configuration included the HTTP Event Collector for receiving MQTT data, Syslog for network logs and IoT devices, and SNMP for monitoring critical infrastructure. The results demonstrated advanced proactive detection capabilities, generating automatic alerts for critical temperatures, interface failures, and brute force attempts, with notifications via email and control panels in Splunk. The solution stood out for its efficiency in centralized visualization through interactive dashboards that show the status of sensors, network interfaces, and events classified by criticality in real time, facilitating decision-making. The integration of data streams from multiple sources proved to be robust, enabling effective correlation of security events. Key benefits include leveraging open-source software such as pfSense and Mosquitto alongside free versions of Splunk, its scalability to incorporate new devices and protocols, and its ability to reduce incident response times through automation. The project validates that it is possible to implement advanced security monitoring systems in SMEs with limited resources. The combination of virtualization, IoT-optimized protocols, and Splunk's analytics capabilities offers a sustainable model for protecting critical infrastructure against growing cyber threats. The findings underscore the need for continuous monitoring in environments with increasing adoption of IoT devices.

Keywords: Splunk, IoT, MQTT, SNMP, Cybersecurity, SMEs, Real-time monitoring, Virtualization.

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

CAPITULO 1

INTRODUCCIÓN

1.1. Definición del proyecto

La digitalización ha impulsado la transformación operativa de las pequeñas y medianas empresas (PYMEs), permitiéndoles acceder a nuevas oportunidades en el mercado. Sin embargo, esta adopción tecnológica también conlleva riesgos importantes en términos de seguridad de la información. Garantizar la protección de los sistemas y datos es ahora una prioridad estratégica para las organizaciones. Por este motivo, las PYMEs ecuatorianas enfrentan debilidades en la gestión de la ciberseguridad, debido a la falta de conocimiento, errores humanos y omisiones en la implementación de controles adecuados. Ante este panorama, se hace necesario diseñar soluciones prácticas y escalables que fortalezcan la seguridad digital de estas organizaciones, orientadas por buenas prácticas y normas internacionales como ISO 27001 y COBIT 5.0 (Bueno & Haz, 2022).

En este contexto del avance tecnológico, las empresas quieren estar a la vanguardia de este movimiento que se ha disparado en esta última década por tal motivo optan por adquirir infraestructura orientada a la seguridad externa e interna. Ahora se ha tomado en cuenta 2 aspectos para el desarrollo de este trabajo, el aspecto del entreteniendo y la seguridad, por lo que las empresas llegan a comprar dispositivos inteligentes y poder abarcar una mejor imagen y seguridad hacia sus clientes pero que son estos dispositivos inteligentes.

Aunque el término “Internet de las Cosas” fue acuñado en 1999, su evolución ha sido constante desde los años 70, y ha cobrado especial relevancia en la última década con la integración de sensores en dispositivos cotidianos. Esta madurez tecnológica ha permitido que

incluso pequeñas empresas puedan incorporar IoT a sus operaciones, aunque sin contar siempre con mecanismos adecuados de ciberseguridad (Kaspersky, 2025).

La Internet de las Cosas (IoT) se define como un sistema de dispositivos electrónicos interconectados capaces de recopilar y transmitir datos a través de redes, sin necesidad de intervención humana constante. Cualquier objeto equipado con un interruptor de encendido y asignado una dirección IP puede formar parte de esta red, lo que abarca desde dispositivos cotidianos hasta aplicaciones complejas en sectores como la salud, informático o la automatización industrial (Kaspersky, 2025).

El funcionamiento de la IoT se basa en cuatro elementos fundamentales:

- **Sensores o dispositivos:** Son los componentes encargados de recolectar datos del entorno. Un único dispositivo puede integrar múltiples sensores (como GPS, cámaras o acelerómetros) para monitorear diversas variables.
- **Conectividad:** Una vez recopilados, los datos se transmiten a través de diferentes canales – Wi-Fi, Bluetooth, LPWAN o Ethernet, entre otros – hacia la nube, lo que permite que la información esté disponible para su procesamiento.
- **Procesamiento de datos:** En la nube, los datos son analizados mediante software especializado que decide, de forma automática o previa intervención, si es necesaria alguna acción (por ejemplo, emitir alertas o ajustar parámetros de operación).
- **Interfaz de usuario:** Si la situación requiere la intervención o supervisión humana, la interfaz permite visualizar la información, emitir comandos y controlar los dispositivos conectados.

Si bien la IoT ofrece ventajas como eficiencia operativa, reducción de costos y automatización de procesos, también introduce desafíos técnicos y de seguridad. Entre los más relevantes se encuentran la complejidad en la gestión de dispositivos, la falta de estándares de compatibilidad y los riesgos de privacidad, factores que afectan con mayor intensidad a las PYMEs debido a sus limitaciones técnicas, y uno de los mayores desafíos que plantea la IoT es la protección de datos generado por los dispositivos. La falta de políticas claras de seguridad, diversas marcas y modelos de dispositivos, junto con una gestión débil de la privacidad, puede facilitar el acceso no autorizado, el espionaje o la explotación de vulnerabilidades. Estos riesgos hacen necesario contar con sistemas de monitoreo que permitan identificar comportamientos anómalos y reaccionar de forma proactiva ante posibles amenazas.

Además, la IoT tiene diversas aplicaciones que abarcan desde el ámbito doméstico con hogares inteligentes, pasando por la automatización de procesos en ciudades y vehículos autónomos, hasta sectores especializados como la telesalud, la agricultura inteligente o el comercio minorista, donde se optimizan procesos y se genera una mayor eficiencia operativa.

El crecimiento de la IoT y el avance de tecnologías como la inteligencia artificial anticipan sistemas de monitoreo más predictivos. Este proyecto prepara a las PYMEs para adoptar estas tecnologías de forma segura y controlada. Sin embargo, esta transformación digital conlleva desafíos de seguridad, ya que muchos de estos dispositivos carecen de mecanismos avanzados de monitoreo, lo que los hace vulnerables a ataques cibernéticos, accesos no autorizados y fallos operativos (Kaspersky, 2025; Bueno y Haz, 2022).

1.2. Justificación e importancia del trabajo de investigación

El avance de la transformación digital ha generado un impacto significativo en las pequeñas y medianas empresas, impulsándolas a adoptar tecnologías emergentes muy variadas en el mercado como la Internet de las Cosas (IoT) para optimizar sus procesos, mayor seguridad interna y externa, reducir costos y mejorar su competitividad. Sin embargo, esta rápida adopción ha traído consigo un aumento considerable en los riesgos de las vulnerabilidades a amenazas cibernéticas.

Esto puede darse por la falta de recursos económicos, el poco conocimiento técnico especializado, la ausencia de políticas de ciberseguridad claras y acordes a la realidad de la empresa que convierten a las PYMEs en blancos fáciles para ataques digitales, accesos no autorizados y fallos operativos que resultan en pérdidas económicas significativas, daño reputacional e incluso el cierre de la empresa.

La importancia y desarrollo de esta investigación se enfoca en ofrecer una solución escalable y accesible mediante el diseño de un sistema de monitoreo basado en la plataforma *Splunk*, que aprovecha los protocolos MQTT y SNMP para integrar dispositivos IoT, recolectar datos en tiempo real, identificar eventos anómalos y emitir alertas de seguridad automatizadas. Este enfoque mejora significativamente la visibilidad operativa de la infraestructura tecnológica, permitiendo una respuesta proactiva ante incidentes y asegurando la continuidad del negocio en entornos empresariales con recursos limitados.

Con esta propuesta, se espera fortalecer la seguridad en entornos IoT, mejorar la eficiencia operativa de las empresas y contribuir a la adopción segura de tecnologías emergentes en Ecuador.

1.3. Alcance

El presente trabajo se centra en el diseño de un sistema de monitoreo de seguridad para dispositivos IoT orientado a las PYMEs del Ecuador. La solución propuesta integra la plataforma *Splunk* junto con los protocolos MQTT y SNMP, permitiendo la supervisión en tiempo real del estado operativo de los dispositivos, la detección de eventos anómalos y la emisión de alertas de seguridad automatizadas.

El desarrollo se basará exclusivamente en entornos simulados, utilizando herramientas de software, dispositivos IoT domésticos y máquinas virtuales que representen escenarios típicos de infraestructura tecnológica en PYMEs. El proyecto contempla la configuración de reglas de correlación, la generación de reportes de seguridad y la validación del sistema a través de pruebas de laboratorio.

Este TFM no contempla la implementación directa en infraestructuras empresariales reales, ni el desarrollo de hardware adicional. La solución se basa en herramientas existentes y accesibles, con el objetivo de garantizar su aplicabilidad en contextos organizacionales con recursos limitados.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar un sistema virtualizado de monitoreo de seguridad basado en Splunk, integrando los protocolos MQTT y SNMP, que permita a las PYMEs del Ecuador supervisar dispositivos IoT y detectar amenazas de forma eficiente y en tiempo real.

1.4.2. Objetivo especifico

- Diseñar y configurar un entorno de prueba que simule una infraestructura IoT de una PYME.

- Investigar las capacidades de Splunk, MQTT y SNMP para el monitoreo de eventos en entornos IoT.
- Implementar reglas de correlación y alertas en Splunk para la detección de eventos anómalos.
- Evaluar el rendimiento y aplicabilidad del sistema en un entorno simulado.
- Generar alertas en tiempo real y reportes sobre eventos críticos de seguridad.

CAPITULO 2

REVISIÓN DE LITERATURA

2.1. Estado del Arte

El estudio desarrollado por López (2024) propone una plataforma para almacenar, procesar y accesar datos, gracias a esta plataforma, los usuarios tienen una visión completa y coherente que mejora la toma de decisiones y sobre todo la eficiencia en la Universidad Oviedo. En la identificación de las necesidades de la infraestructura, el autor analiza.

- **El Open Source**: Son herramientas con código abierto.
- **Los entornos necesarios**: Se destaca la importancia de una red robusta que garantiza el flujo de comunicación con el sistema.
- **Disponibilidad**: La infraestructura debe garantizar niveles altos de disponibilidad y continuidad operativa para que se integren los datos.
- **Servicios críticos**: El autor manifiesta que la plataforma debe contener los servicios de seguridad, autentificación, autorización y monitoreo, a fin que garantice la protección y el control de los datos y recursos del sistema.
- **Flexibilidad, datos recurrentes y procesamiento de datos**: La infraestructura tiene la capacidad de manejar diversos formatos, protocolos y orígenes para que sea flexible; al mismo tiempo sea capaz de soportar la ocurrencia de los usuarios que trabajen al mismo tiempo.
- **Exploración de datos y despliegue**: El autor de la plataforma expone que debe ser capaz de extraer, filtrar y buscar información; y, que todos los componentes se desplieguen a todos los contenedores para garantizar la portabilidad y

escalabilidad.

El autor de este estudio desarrolla la plataforma utilizando tecnologías NiFi, ELK Y Kunna para integrar y visualizar datos.

El estudio expuesto se asocia indirectamente a la problemática detectada, dado que el proyecto de “Diseño de un Sistema de Monitoreo de Seguridad IoT para Pymes del Ecuador integrado Splunk, MQTT y SNMP requiere supervisar IoT para recopilar y analizar datos; si se analiza la brecha entre el tema de este proyecto con el estudio mencionado se manifiesta que López (2024) netamente se centra en desarrollar una plataforma para recopilar datos, más no utiliza otras tecnologías para garantizar la seguridad de dicha información mientras los usuarios interactúan, el presente proyecto mejora dicha área de seguridad, dado que al utilizar los protocolos de comunicación MQTT y SNMP, se monitorea la gestión de los dispositivos IoT y las redes, a su vez, la comunicación permite que se monitoree los datos en tiempo real.

En cuanto al uso de la plataforma de Splunk, el proyecto citado por López (2024), almacena datos, en cambio en el presente proyecto se analiza datos, organiza, almacena, busca y visualiza grandes volúmenes de datos tomando como prevalencia el blindaje de información.

Otro tema que se cita es la investigación realizada por Romero (2021) que dispone de una plataforma SIRP con software de código abierto (Open Source) para fines académicos, a nivel metodológico, el autor inicialmente analiza las funciones que requiere la plataforma, posteriormente escoge los componentes y diseña la solución, para ello implementa e integra diferentes componentes, entre ellos, el autor indica que usar productos comerciales como Splunk es poco viable debido al presupuesto de las PYMEs, incluso, grandes empresas analizan el costo de la licencia, para disminuir costos Romero (2021) utiliza Open Source considerando que hay una mega comunidad de soporte.

Con estos datos, se evidencia que el estudio tampoco se enfoca en garantizar la seguridad de los datos, sino más bien la plataforma de uso académico se desarrolla con código abierto y se indica que utilizar Splunk no es viable por los altos costos de la licencia. Una vez más se prueba que el presente proyecto es innovador y a más de consolidar datos brinda seguridad que es una arista esencial cuando los datos y la información es el capital invaluable en las empresas, la evolución e implementación tecnológica obliga a las empresas estar a la vanguardia de la tecnología sin olvidar la seguridad informática.

El presente estudio corrobora a centralizar los procesos, disminuir brechas de seguridad, además, esta solución tiene una mínima inversión, debido a que se abarata costos si se compara las soluciones comerciales cuyas licencias son más costosas y con una implementación que toma más tiempo.

Es vital acotar que las soluciones con Splunk ofrece al corporativo recopilar, analizar y visualizar datos provenientes de diversas fuentes, pues, proporciona una visión integral de los sistemas y de las aplicaciones que se utiliza en la empresa. También, el sistema correlaciona, captura e indexa datos en tiempo real, creando alertas, paneles gráficos, informes y visualizaciones para que así la empresa reconozca patrones de datos comunes y diagnostique problemas potenciales.

El sistema Splunk guarda búsquedas y etiquetas que se reconoce como información importante, ayudando a la empresa a crear sistemas inteligentes con resultados instantáneos que logra a que lo usuarios tomen menos tiempo en resolver problemas con brechas de seguridad.

En el campo de la seguridad informática Splunk ayuda a recopilar datos de varias fuentes como firewall, sistema de detección de instrucciones, SOAP, aplicaciones creando una visión general del estado de seguridad de la empresa. Al momento que *Splunk* utiliza algoritmos

avanzados y gracias al machine *learning* la empresa identifica patrones sospechosos o comportamientos maliciosos permitiendo que los analistas de sistemas detecten de forma temprana amenazas, optimicen la infraestructura de seguridad y mejoren la productividad.

1.2. Marco Teórico

1.2.1. Definición de IOT

El Internet de las Cosas (IoT) se refiere a la interconexión embebida de dispositivos y objetos físicos a través de internet, permitiendo que estos recopilen, intercambien y analicen datos de manera automática y en tiempo real. Este concepto ha ganado relevancia en los últimos años debido a la creciente capacidad de los sensores, la expansión de la conectividad y los avances en el procesamiento de datos. Al permitir conectar dispositivos inteligentes entre sí y con otros sistemas conectados a la red generan un ecosistema amplio, inteligente y automatizado para el intercambio de datos y la ejecución de tareas sin la intervención humana. Este tipo de tecnologías se han aplicado a lo largo del tiempo en diversos sectores industriales, de construcción y empresariales, mismos que han transformado las operaciones cotidianas, optimizando recursos y haciendo más eficiente el cumplimiento de actividades específicas, posterior a ello el análisis de estos datos en tiempo real facilita la identificación de patrones o anomalías, contribuyendo a mejorar la eficiencia operativa y los resultados organizacionales.

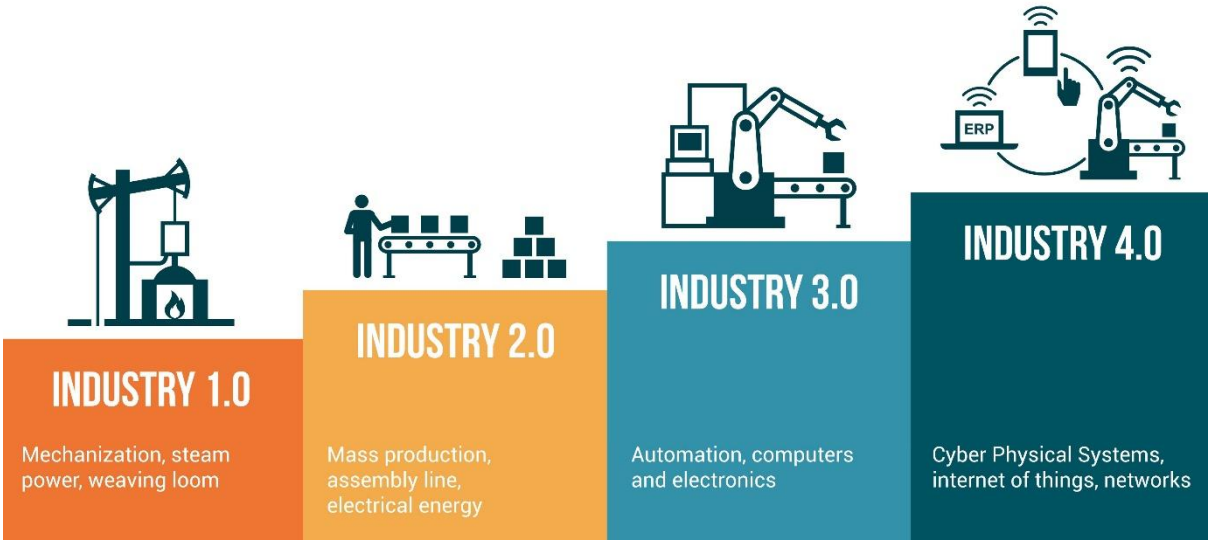
Es importante considerar que los objetos físicos conectados a Internet tienen la capacidad de comunicar su estado y cambios, responder a eventos del entorno e incluso actuar de forma autónoma, sin necesidad de intervención humana. Esta autonomía acorta la distancia entre el mundo físico y el digital, permitiendo la creación de rutinas inteligentes que operan de manera automatizada. La tecnología IoT ha sido clave para la convergencia de redes inalámbricas,

sistemas de datos, dispositivos y tecnologías microelectromecánicas (MEMS), integrándolos con las capacidades de la conectividad en línea.

En este contexto, la Internet de las Cosas se ha consolidado como uno de los pilares fundamentales de la denominada Industria 4.0, que impulsa la digitalización de procesos (actividades tradicionales manuales o analógicas) y cadenas de valor aplicando la tecnología en cada etapa. Esta transformación se basa en la integración de tecnologías avanzadas como el procesamiento de datos, la automatización, la robotización y el software inteligente, aplicados a prácticamente todos los ámbitos del quehacer humano (Carrión, 2024).

Figura 1

Evolución de la Industria



Nota. En la figura se observa los cuatro niveles de evolución de la industria, iniciando con industry 1.0 y finalizando en industry 4.0. Tomado de (Kriesche, 2017).

La importancia de IoT en la vida cotidiana ha generado que actualmente estén presentes en los dispositivos como teléfonos móviles, entretenimiento, asistentes de voz y vehículos. Por

ejemplo, una casa inteligente puede tener luces o termostatos que se controlan desde un dispositivo móvil, lo cual incurre en optimización de recursos como mayor eficiencia con automatizaciones y optimizaciones que dan como resultado una mayor productividad, la toma de decisiones basadas en datos con mejor información y nuevos modelos de negocio, ahorro de costos operativos mediante la automatización de tareas repetitivas y la optimización del uso de recursos como la energía, lo que mejora la rentabilidad y contribuye a la sostenibilidad. Además, existen tecnologías que permite recopilar datos sobre el comportamiento del cliente en tiempo real, lo que facilita la creación de experiencias más personalizadas y relevantes. Por ejemplo, en el sector minorista, los sensores pueden rastrear patrones de movimiento dentro de una tienda y generar ofertas adaptadas a las preferencias del consumidor (International Business Machines, 2023). Ahora se podrá identificar dispositivos como:

- Relojes inteligentes que permiten ver la frecuencia cardíaca o llevar un recuento de los pasos al caminar.
- Aplicaciones para abrir y cerrar las ventanas en la vivienda o graduar la luz.
- Robots que hacen la limpieza.
- Collares para controlar el sueño de las mascotas o si tienen signos de alguna enfermedad.
- Sensores para colocar en las raíces de una planta, que nos avisan si le falta agua.
- Cestos de basura inteligentes para medir los kilos de basura que se reciclan.
- Una nevera que indica la fecha de caducidad de los productos.
- Sistemas de seguridad y sensores médicos (Alaisecure, s. f.)

Si bien la IoT ofrece grandes beneficios, también introduce importantes desafíos de seguridad. La intercepción de datos entre dispositivos y la nube representa un riesgo crítico, ya que información sensible puede ser robada y utilizada con fines maliciosos. Cada dispositivo conectado se convierte en un posible punto de entrada para atacantes, lo que puede facilitar la instalación de malware o comprometer toda la red.

Además, cuando los dispositivos IoT están integrados en procesos industriales o de infraestructura crítica, como manufactura, transporte, seguridad o redes eléctricas, un ataque puede tener consecuencias severas, desde la alteración de procesos hasta la interrupción de servicios esenciales. Estos riesgos requieren sistemas de detección y respuestas robustos para proteger los activos digitales y operativos de las organizaciones.

Ahora también debemos considerar los desafíos de seguridad en este tipo de dispositivos algunos heredados de tecnologías previas, pero con particularidades propias. Uno de los más preocupantes es el aumento de botnets, redes de dispositivos IoT comprometidas que pueden ser controlados por ciberdelincuentes para realizar ataques masivos. A esto se suma la falta de cifrado, ya que muchos dispositivos no cuentan con la capacidad de implementar protocolos de seguridad robustos, lo que expone la información transmitida.

Otro punto crítico es el uso de contraseñas predeterminadas débiles, que frecuentemente no son cambiadas por los usuarios y facilitan el acceso no autorizado. Además, los dispositivos IoT suelen ser objetivos de ataques de suplantación de identidad (phishing), los cuales pueden derivar en la instalación de malware capaz de controlar o desactivar los equipos. Por último, está en juego la privacidad del usuario, ya que muchos dispositivos recopilan datos personales que, si son vulnerados, pueden ser utilizados para fines maliciosos.

Finamente esta tecnología es y será parte del mundo y que se considera revolucionaria al posicionarse como una tecnología clave en la transformación digital de empresas y ciudades, gracias a su capacidad para mejorar la eficiencia operativa, optimizar recursos y generar nuevas oportunidades de negocio. Su crecimiento ha sido impulsado por la reducción de costos en sensores, el aumento en la demanda de conectividad y la expansión de redes como 5G. Entre las principales tendencias que consolidan su papel revolucionario se destacan:

- La masificación de dispositivos IoT, accesibles para empresas y consumidores.
- El incremento de inversiones en investigación y soluciones innovadoras.
- La evolución de plataformas IoT para gestionar y analizar grandes volúmenes de datos.
- Un mayor enfoque en la seguridad, ante la creciente exposición de dispositivos conectados.
- La integración de la inteligencia artificial (IA), que potencia la toma de decisiones automatizadas y predictivas.
- El desarrollo de ciudades inteligentes, donde sensores y dispositivos mejoran la gestión urbana, el tráfico, la seguridad y el medio ambiente (Fortinet, s. f.-a)

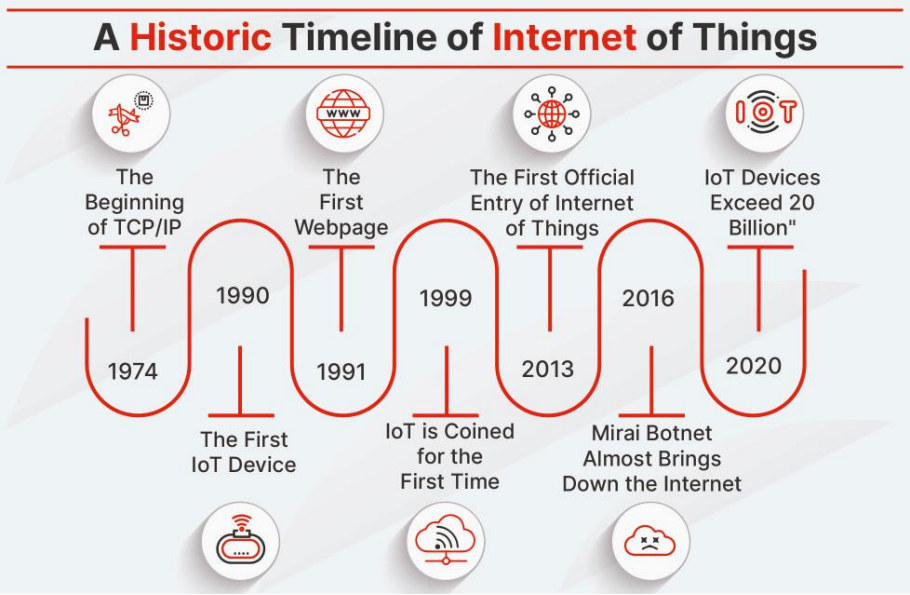
Como parte del desarrollo del IoT se establece una línea del tiempo capaz de visualizar la creciente evolución del concepto y desarrollo progresivo desde la década de 1970. En 1974, se establecieron los protocolos TCP/IP, base para la comunicación en red. En 1990, se creó el primer dispositivo IoT (una tostadora controlada por Internet). Poco después, en 1991, surgió la primera página web. El término “Internet de las Cosas” fue acuñado por Kevin Ashton en 1999,

al proponer dispositivos con sensores capaces de interactuar con su entorno. En 2013, la IoT fue reconocido oficialmente por el Diccionario de Oxford.

En 2016, Botnet Mirai expuso las vulnerabilidades de los dispositivos conectados al ejecutar un ataque DDoS masivo. Para 2020, el número de dispositivos IoT superó los 20 mil millones, consolidando su papel clave en la economía digital y evidenciando la necesidad de enfoques seguros y escalables en su implementación.

Figura 2

Histórico de evolución de IoT



Nota. En la Figura 2 se observa los 5 hitos de evolución de IoT que inició en 1974 y se registró la última evolución en el año 2020. Tomado de (Fortinet, s. f.-a).

1.2.2. Protocolos de comunicación en IoT

a. **MQTT *(Message Queuing Telemetry Transport)***

MQTT es un protocolo abierto, ligero y eficiente, desarrollado específicamente para facilitar la comunicación entre dispositivos con capacidades limitadas, operando en redes con bajo ancho de banda, alta latencia o poca fiabilidad. Su arquitectura se basa en un esquema de

publicación/suscripción, donde los dispositivos pueden enviar y recibir mensajes a través de un intermediario conocido como *broker*.

Es diseñado para entornos IoT (Internet de las Cosas) y M2M (Máquina a Máquina), MQTT permite el intercambio asíncrono de datos en tiempo real entre sensores, dispositivos portátiles, plataformas industriales, hogares inteligentes y servicios en la nube. Esta mensajería se estructura en temas, a los cuales los clientes se suscriben para recibir la información publicada, permitiendo una conexión escalable, flexible y segura.

Debido a su simplicidad, bajo consumo de recursos y facilidad de implementación, MQTT se ha convertido en el estándar de factor en mensajería IoT, siendo adoptado por millones de dispositivos a nivel global. Ha sido estandarizado por OASIS e ISO, lo que refuerza su confiabilidad en escenarios empresariales y técnicos donde la conectividad distribuida es esencial (Jude, 2025).

Ventajas de MQTT

El protocolo MQTT se ha consolidado como uno de los más eficientes y populares en el ámbito del Internet de las Cosas (IoT) gracias a su diseño ligero, escalable y seguro. Su arquitectura permite una comunicación bidireccional, eficiente incluso en redes poco confiables, lo que lo convierte en una solución ideal para dispositivos con recursos limitados. Entre sus principales ventajas destacan:

- Ligereza y eficiencia: MQTT requiere una mínima carga de procesamiento y ancho de banda, permitiendo su uso en microcontroladores y sensores con capacidades reducidas. Un mensaje puede tener solo dos bytes, optimizando el tráfico de red.
- Escalabilidad: Puede gestionar millones de dispositivos conectados gracias a su

bajo consumo de recursos y al modelo de publicación/suscripción que desacopla emisor y receptor.

- Fiabilidad: Soporta tres niveles de calidad de servicio (QoS 0, 1 y 2), garantizando una entrega controlada de mensajes, incluso en conexiones inestables o con alta latencia.
- Seguridad: Admite mecanismos modernos como TLS/SSL, autenticación con credenciales y certificados de cliente, protegiendo la confidencialidad y el acceso a los datos transmitidos.
- Sesiones persistentes: Permite mantener sesiones con estado, lo que asegura la entrega de mensajes pendientes incluso después de una desconexión temporal del dispositivo.
- Compatibilidad amplia: MQTT es compatible con múltiples lenguajes de programación (como Python, PHP, Node.js y Golang), lo que facilita su implementación en diversos entornos y plataformas tecnológicas.

Estas características hacen que MQTT sea un protocolo robusto, adaptable y esencial para sistemas IoT modernos que requieren comunicación continua, fiable y segura (Amazon Web Services, s.f.).

Funciones y clientes MQTT

El protocolo MQTT opera bajo un modelo de publicación/suscripción, donde los dispositivos conectados conocidos como clientes MQTT pueden actuar como publicadores (que envían mensajes sobre un tema) o suscriptores (que reciben mensajes según su interés). Todos los

mensajes son gestionados por un intermediario denominado *broker* MQTT, responsable de enrutar y distribuir la información.

Cada mensaje se publica en un tema jerárquico (por ejemplo: hogar/piso1/cocina/temperatura). El *broker* se encarga de filtrar y entregar estos mensajes únicamente a los clientes suscritos al tema correspondiente, manteniendo a los dispositivos completamente desacoplados.

Figura 3

Modelo de funcionamiento de MQTT

Nota. En la Figura 3 se observa un modelo de funcionamiento de temperatura de un sensor donde MQTT suple el rol de cliente. Tomado de (Jude, 2025).

Uno de los aspectos claves de MQTT es su soporte para tres niveles de calidad de servicio (QoS), que definen la fiabilidad con la que se entregan los mensajes:

- **QoS 0 (entrega como máximo una vez)**: El mensaje se entrega una vez y no se garantiza su recepción. Es el más rápido y con menor consumo de recursos.
- **QoS 1 (al menos una vez):** El mensaje se entrega al menos una vez, y el receptor envía una confirmación. Puede recibir duplicados.
- **QoS 2 (exactamente una vez)**: Garantiza que el mensaje se entregue solo una vez

mediante un proceso de confirmación más complejo. Es el más seguro, pero también el más costoso en rendimiento.

Tabla 1

Descripción de los niveles de QoS

Nivel QoS	Nombre	Descripción	Ventaja	Desventaja	Ejemplo de uso
0	A lo sumo una vez	El mensaje se entrega una sola vez, sin confirmación del receptor.	Más rápido y menor uso de recursos.	No hay garantía de entrega.	Lectura de sensores no críticos.
1	Al menos una vez	El mensaje se entrega y se espera confirmación; puede repetirse si es necesario.	Garantiza recepción del mensaje.	Posibles duplicados.	Alarmas, monitoreo ambiental.
2	Exactamente una vez	Se asegura que el mensaje se entregue solo una vez mediante un intercambio seguro.	Alta fiabilidad en la entrega.	Mayor complejidad y uso de recursos.	Transacciones financieras, comandos críticos.

Nota. En la Tabla 1 se describen los tres niveles de QoS, las ventajas, desventajas y ejemplos con la finalidad que el lector tenga una mayor comprensión del tema.

MQTT ofrece flexibilidad para adaptarse a distintos escenarios, desde sensores de baja prioridad hasta sistemas críticos que requieren alta fiabilidad (Fortinet, s. f.-c).

a. SNMP *(Simple Network Management Protocol)*

SNMP es un protocolo de nivel de aplicación utilizado para supervisar y administrar dispositivos dentro de redes IP. Facilita la recopilación de información operativa, el monitoreo de desempeño, la detección temprana de errores y la configuración remota de dispositivos conectados.

Gracias a su enfoque estándar y no propietario, SNMP es compatible con *routers, switches,* servidores e impresoras de distintos fabricantes, lo que lo convierte en una solución accesible y flexible para organizaciones con infraestructura tecnológica diversa.

En el contexto de las pequeñas y medianas empresas (PYMEs), SNMP ofrece una alternativa económica y eficiente para mantener visibilidad del estado de sus sistemas, anticipar fallos críticos y optimizar la administración de recursos TI sin necesidad de plataformas costosas o complejas. Su implementación contribuye a mejorar la continuidad operativa y la seguridad de la red, factores clave en entornos empresariales con recursos limitados (Paessler The Monitoring Experts, s.f.).

Versiones de SNMP

Según Fortinet (s.f), existen varias versiones de SNMP:

- **SNMPv1**: Primera versión, básica, con seguridad limitada.
- **SNMPv2**: Mejora en el rendimiento y más comandos, pero también con limitaciones de seguridad.
- **SNMPv3**: Agrega seguridad fuerte con autenticación y cifrado, recomendada actualmente.

La evolución de SNMP busca mejorar la seguridad y eficiencia en la gestión de redes (Fortinet, s. f.-b)

Componentes principales de SNMP

- Dispositivos gestionados: Equipos de red (como *routers* o servidores) que proporcionan información de estado.
- Agentes SNMP: *Software* que corre en los dispositivos, recolecta métricas (como uso de CPU, tráfico, errores) y las guarda en la MIB (*Management Information Base*).

- NMS (*Network Management System*): Software centralizado que consulta, analiza y presenta la información mediante paneles o alertas.
- Utiliza principalmente dos puertos del protocolo UDP: El puerto 161, que es utilizado por los agentes SNMP para recibir solicitudes provenientes del administrador, y el puerto 162, empleado por el administrador para recibir mensajes tipo *trap* enviados desde los agentes. En ambos casos, los puertos de origen pueden ser dinámicos o no estar predefinidos (Omnitron Systems, s.f.).

Operaciones clave de SNMP

Las operaciones basadas en agentes requieren de pequeños servicios que se encargan de obtener las métricas e informar a los a los administradores y mismas que pueden acelerar la monitorización en ciertos casos.

- Get: El NMS solicita información a un dispositivo.
- Set: El NMS modifica parámetros del dispositivo.
- Traps: El agente envía alertas no solicitadas ante eventos críticos.
- Getnext (walk): Permite recorrer múltiples valores relacionados en la mib.
- Recorrido masivo: Solicitud grande para recuperar fragmentos de árboles uio de una red.
- Tabla SNMP: Muestra un resultado similar para las solicitudes en formato tabular.
- Conjunto SNMP: Solicitud para cambiar valores de un dispositivo administrado.
- Trampa: Son notificaciones de un agente sobre eventos ocurridos (Techtarget, s.f).

Funcionamiento técnico

SNMP funciona con un modelo cliente-servidor (*manager-agente*). Puede operar en dos modos:

- *Pull* (extracción): El NMS solicita información.
- *Push* (inserción): El agente envía datos sin petición previa (traps).

Las métricas recolectadas se definen y organizan jerárquicamente en la MIB, y se identifican mediante OIDs (*Object Identifiers*) únicos por fabricante o tipo de dispositivo.

Beneficios para las organizaciones

- Visibilidad en tiempo real del estado de la red.
- Respuesta proactiva ante fallos mediante TRAPS.
- Configuración remota de dispositivos.
- Escalabilidad en redes grandes.
- Mejora de la seguridad usando SNMPv3, que incluye cifrado y autenticación (Omnitron Systems, s. f.).

Tabla 2

Comparación entre ventajas y limitaciones del Protocolo SNMP

Ventajas de SNMP	Limitaciones de SNMP
Protocolo estándar y ampliamente soportado por diversos dispositivos de red.	Los mensajes *trap* enviados vía UDP pueden perderse, lo que puede ocultar fallos importantes.
Permite el monitoreo remoto, recolección de métricas y configuración básica.	La implementación incorrecta del protocolo por parte de fabricantes puede generar datos erróneos.

Ventajas de SNMP	Limitaciones de SNMP
Eficiente en redes pequeñas y medianas, con recursos limitados.	No proporciona información detallada como versiones de sistemas operativos o aplicaciones instaladas.
Compatible con herramientas gratuitas o de bajo costo para PYMEs.	El recorrido SNMP (*walk*) en redes con muchos hosts puede ser lento y consumir muchos recursos.
Puede complementarse con herramientas como *Checkmk* que optimizan el rendimiento.	Empresas como Microsoft y Google han dejado de utilizarlo por sus limitaciones técnicas.
Permite supervisar equipos heterogéneos desde una única plataforma.	A pesar de su uso generalizado, no se vislumbra como una solución a largo plazo sin mejoras o sustituciones.

Nota. En la Tabla se describen las principales ventajas y limitaciones del protocolo SNMP.

Tabla 3

Comparativa de MQTT vs SNMP en entornos de monitoreo IoT

Aspecto	MQTT	SNMP
Modelo de comunicación	Publicador/Suscriptor	Cliente/Servidor (Manager/Agente)
Transporte	TCP/IP (puerto 1883), opcionalmente con TLS.	UDP (puertos 161 y 162).
Uso típico	IoT, sensores, entornos con poco ancho de banda.	Monitoreo de redes empresariales tradicionales
Consumo de recursos	Muy bajo, ideal para dispositivos con capacidad limitada.	Ligero, pero requiere más procesamiento en grandes redes.
Fiabilidad del mensaje	Soporta QoS 0, 1 y 2 para control de entrega.	Sin confirmación en TRAPS; posibilidad de pérdida de paquetes UDP

Aspecto	MQTT	SNMP
Seguridad	TLS/SSL, autenticación de cliente.	SNMPv3 ofrece cifrado y autenticación; versiones anteriores no son seguras.
Escalabilidad	Alta, para millones de dispositivos IoT.	Alta, especialmente en infraestructuras de red tradicionales.
Facilidad de implementación	Requiere un *broker*; fácil de integrar con múltiples lenguajes.	Ampliamente soportado; configuración específica por fabricante.
Limitaciones	No diseñado para lectura directa de configuración de red.	No proporciona información detallada del sistema operativo o software.

Nota. En la Tabla se observa aspectos comparativos de MQTT vs SNMP en entornos de monitoreo IoT.

a. ZigBee

ZigBee es un protocolo inalámbrico lanzado en 2004 como alternativa a Bluetooth y Wifi, especialmente utilizado en domótica y dispositivos inteligentes. Está diseñado para consumir poca energía, permitiendo que los aparatos funcionen más tiempo con baterías. Además, ayuda a reducir la congestión en redes Wifi y ofrece un buen alcance gracias a su estructura en red de malla. Soporta comunicación entre dispositivos de distintas marcas, opera con rapidez (hasta 250 Kbps) y protege las conexiones con cifrado AES de 128 bits.

Este protocolo fue definido por la *ZigBee Alliance* y se basa en el estándar IEEE 802.15.4 para sus niveles más bajos de comunicación. Sus características más destacadas son su bajo consumo, la baja velocidad de transferencia y su topología en malla, lo que permite que los

dispositivos se conecten y mantengan la red de forma automática y dinámica. Esto lo hace ideal para aplicaciones en campo, como sensores o medidores, y también en el hogar, integrándose fácilmente en bombillas, electrodomésticos y sistemas de alarma (Venco, s.f.).

ZigBee permite conectar dispositivos como luces, enchufes y cerraduras en una red doméstica que puede usarse directamente con controles físicos o conectarse a *hubs* como *Homey*, para controlarlos a través del teléfono o aplicaciones. Aunque se parece a otros protocolos como *Z-Wave, Wifi y Bluetooth, ZigBee* destaca por su compatibilidad con un mayor número de dispositivos de iluminación y enchufes inteligentes (Homey, s. f.).

Finalmente, *ZigBee* es un estándar abierto que garantiza una comunicación segura y eficiente entre dispositivos IoT de bajo consumo. La versión *ZigBee* 3.0 introduce mejoras en seguridad y en la gestión de la red, permitiendo un mejor control remoto de los dispositivos a través de Internet (Digi, s.f.).

Evolución

La evolución de *ZigBee* comenzó en los años 90 con la idea de crear redes inalámbricas de bajo consumo basadas en topologías en malla. A partir de la aprobación del estándar IEEE 802.15.4, se formó la *ZigBee Alliance* a inicios de los 2000 para desarrollar este protocolo. La primera gran actualización llegó en 2006, donde se mejoró la estructura interna del protocolo, y poco después, en 2007, se lanzó la versión *ZigBee PRO*. Esta nueva versión fue retro compatible y trajo mejoras importantes, como la introducción de los *Application Profiles*, que estandarizan cómo los dispositivos se comunican según su función (Venco, s.f.).

Gracias a estos perfiles, los fabricantes pudieron integrar *ZigBee* en sus productos sin tener que rediseñar todo el sistema de red, facilitando la interoperabilidad entre equipos de distintas marcas. El primer perfil fue *Home Automation* (HA), pensado para domótica, y con el

tiempo se sumaron otros, como *Smart Energy* (SE) para sistemas de medición, *Commercial Building Automation* (CBA) y *Personal Health and Hospital Care* (PHHC).

Más recientemente, en 2021, la *ZigBee Alliance* cambió su nombre a *Connectivity Standards Alliance* (CSA) para reflejar un enfoque más amplio hacia la conectividad en general dentro del ecosistema del Internet de las Cosas (Digi, s.f.).

Estructura de una red

La red *ZigBee* se basa en una topología en malla donde los dispositivos pueden actualizar sus rutas dinámicamente, logrando así una red robusta y eficiente. Esta estructura permite que los dispositivos se comuniquen entre sí y con el exterior de manera estable. Dentro de la red, cada dispositivo cumple uno de estos tres roles:

- **Coordinador**: Es el nodo principal y único por red. Se encarga de crear, gestionar y coordinar toda la red *ZigBee*. Actúa como punto central, permitiendo conexiones de nuevos dispositivos y administrando la red de área personal (PAN). Generalmente, este rol lo asumen dispositivos como *hubs o bridges*, que también conectan la red *ZigBee* a Internet y a otras tecnologías como *Wi-Fi o Z-Wave*.
- ***Router***: Son dispositivos que, además de cumplir sus funciones normales, ayudan a repetir y enrutar la señal *ZigBee* a otros dispositivos dentro de la red. Normalmente son equipos que están siempre alimentados por corriente y no usan baterías, garantizando una red con mayor cobertura.
- **Dispositivo Final (*End Device*)**: Estos son los dispositivos más simples, como sensores o interruptores a batería. Solo se comunican con *routers* o el coordinador, no repiten señales ni enrutan paquetes. Pueden entrar en modo de bajo consumo para conservar batería y reactivarse cuando es necesario.

Adicionalmente, en implementaciones reales, un *bridge* (o pasarela) se conecta al *router* de Internet mediante un cable ethernet, y es el encargado de distribuir la señal *ZigBee* a los dispositivos usando radiofrecuencia (Homey, s. f.).

Características de *Zigbee*

ZigBee es un protocolo basado en la norma IEEE 802.15.4 que permite crear redes inalámbricas de área personal (WPAN) de bajo consumo y estructura en malla. Los dispositivos *ZigBee* se comunican mediante un chip de radiofrecuencia que opera en la banda de 2,4 GHz, similar a *Wi-Fi* y *Bluetooth*, pero usando menos energía, lo que prolonga notablemente la duración de las baterías. Aunque su alcance en interiores es de unos 10 a 20 metros, *ZigBee* permite que los dispositivos se conecten en cadena para cubrir distancias mayores, ya que pueden reenviar los mensajes entre ellos. Entre sus principales características destacan:

- Los dispositivos *ZigBee* pueden enviar, recibir y reenviar datos a otros dispositivos dentro de la red.
- Es posible conectar múltiples dispositivos a un único *hub* central.
- Gracias a su estructura en malla, la red *ZigBee* ofrece una comunicación robusta y con mayor alcance efectivo.

La topología en malla es clave en *ZigBee*, ya que permite que varios dispositivos actúen como repetidores, transmitiendo las señales a otros nodos. Esto no solo amplía el alcance de la red, sino que también le da la capacidad de autorrecuperarse: si un nodo falla o se desconecta, los dispositivos buscan automáticamente rutas alternativas para mantener la red operativa. Esto la hace más fiable frente a fallos que una red tradicional en estrella, donde la caída del *router* principal afecta a todos los dispositivos (Homey, s. f.).

Por qué usar *ZigBee*

ZigBee es una opción destacada en el mundo de la automatización del hogar por su combinación de bajo consumo energético, escalabilidad y seguridad. Aunque no es la tecnología perfecta para todas las aplicaciones, ofrece ventajas claras frente a otras alternativas. Entre sus beneficios clave se encuentran:

- Control remoto mejorado: Permite manejar dispositivos desde cualquier lugar mediante un *hub* inteligente, facilitando el control a través del móvil y la automatización.
- Conectividad en la nube: A través del *hub*, los dispositivos *ZigBee* pueden ser gestionados mediante aplicaciones y permiten la participación de varios usuarios.
- Seguridad robusta: Emplea cifrado AES de 128 bits, ofreciendo un nivel alto de protección.
- Redes estables y autorrecuperables: Si un dispositivo falla o se apaga, la red automáticamente redirige la comunicación sin perder funcionalidad.
- Alta capacidad de conexión: Puede soportar hasta 65.000 dispositivos en una sola red gracias a su estructura en malla.
- Consumo energético eficiente: Ideal para dispositivos que requieren larga duración de batería.
- Costo accesible: En general, los dispositivos *ZigBee* son más económicos que los basados en *Z-Wave.*

Además, *ZigBee* es flexible, compatible con múltiples topologías de red y dispositivos de marcas como *Philips Hue*, Samsung *SmartThings* y Amazon Echo. Su capacidad para actualizaciones OTA (*Over-the-Air*) permite mantener los dispositivos al día de manera sencilla.

En comparación con *Wifi, ZigBee* ofrece menor velocidad de transmisión, pero es mucho más eficiente en términos de energía, algo crucial para la domótica. Frente a *Bluetooth Low Energy* (BLE), *ZigBee* tiene mayor alcance y soporta más dispositivos, aunque BLE puede ofrecer mayor velocidad en la transferencia de datos.

No obstante, *ZigBee* no es universal: Requiere un *hub* para conectarse con smartphones o PCs, tiene un alcance limitado por dispositivo (10-20 metros) y no todos los aparatos inteligentes son compatibles con este estándar (Homey, s.f.).

Aplicaciones

ZigBee es una tecnología versátil que, gracias a su baja latencia y amplia compatibilidad, se adapta a múltiples sectores, más allá del hogar. Además, su adopción resulta atractiva para fabricantes, ya que no implica costos de licencias o regalías. Algunos de sus usos más destacados son:

- **Tecnología verde**: Ideal para gestionar parques solares, parques eólicos y redes de carga de vehículos eléctricos, aprovechando su estructura en malla para conectar múltiples puntos de forma eficiente.
- **Hogar inteligente**: Ampliamente usado para controlar luces, cerraduras, detectores de humo, ventiladores y electrodomésticos. Grandes plataformas como Amazon Echo Plus, Samsung *SmartThings* y *Philips Hue (Signify)* utilizan *ZigBee*, con cientos de millones de dispositivos desplegados en todo el mundo.
- **Energía inteligente**: Los dispositivos compatibles con *ZigBee Pro* 2023 pueden

integrarse en redes energéticas, facilitando un control más eficiente del consumo eléctrico.

- **Medicina**: Permite que sensores portátiles transmitan en tiempo real datos vitales de pacientes, como la frecuencia cardíaca, la presión arterial y niveles de glucosa, hacia centros médicos.
- **Automatización industrial**: Es empleado para gestionar sistemas de iluminación, climatización, seguridad y control de accesos dentro de edificios, optimizando la operación y el ahorro energético (Digi, s.f.).

1. Mosquitto

Mosquitto es una aplicación ligera y de código abierto que implementa el protocolo MQTT (*Message Queuing Telemetry Transport*), diseñado para facilitar la comunicación entre dispositivos en entornos distribuidos, especialmente aquellos con recursos limitados, como los sistemas IoT. Funciona como un *broker* o intermediario encargado de recibir, gestionar y distribuir mensajes publicados por diferentes dispositivos que usan el modelo publicador-suscriptor (Eclipse Foundation, 2018).

Este software permite que múltiples clientes MQTT (sensores, actuadores, aplicaciones, etc.) se comuniquen entre sí sin necesidad de establecer conexiones directas, lo que simplifica y optimiza la arquitectura de red. Al operar bajo el protocolo MQTT, Mosquitto garantiza un bajo consumo de ancho de banda y energía, manteniendo una comunicación constante y eficiente entre los nodos del sistema (Eclipse Foundation, 2018).

Entre sus principales ventajas se destacan su compatibilidad con múltiples sistemas operativos, la capacidad de manejar muchas conexiones simultáneas, su facilidad de configuración, y su soporte para mecanismos de seguridad como autenticación y cifrado TLS.

Estas características hacen que Mosquitto sea una herramienta ampliamente adoptada en soluciones de domótica, monitoreo remoto, automatización industrial y simulaciones de dispositivos IoT. En este proyecto, Mosquitto se utilizó como el canal de transporte de datos simulados generados por sensores IoT, actuando como núcleo de comunicación entre el simulador de dispositivos y el sistema SIEM *Splunk* (Eclipse Foundation, 2018).

2. Máquinas virtuales

Una máquina virtual (VM) es un entorno de software que simula un equipo físico completo, incluyendo componentes como CPU, memoria, almacenamiento y red. Este entorno permite ejecutar sistemas operativos y aplicaciones como si estuvieran funcionando en un hardware real. Cada máquina virtual funciona de forma aislada, lo que proporciona seguridad y flexibilidad para probar o implementar software sin afectar al sistema principal.

Estas máquinas se ejecutan sobre un hipervisor, una capa de software que gestiona los recursos del equipo físico (llamado host) y los distribuye entre las diferentes máquinas virtuales (denominadas guests o invitadas). Esto permite, por ejemplo, correr Linux en un entorno virtual sobre un sistema Windows, sin conflictos entre sistemas.

Gracias a este aislamiento, es posible que una sola computadora física ejecute múltiples sistemas operativos o entornos independientes, lo que optimiza el uso del hardware. Esta capacidad es ampliamente utilizada en entornos corporativos, pruebas de software, educación y desarrollo de sistemas.

Las máquinas virtuales no solo permiten esta versatilidad técnica, sino que también son clave en infraestructuras basadas en la nube, donde se aprovechan para escalar servicios, reducir costos y simplificar el mantenimiento. Este enfoque ha impulsado el crecimiento del mercado de VMs, que superó los 9.500 millones de dólares en 2023 y sigue creciendo con fuerza,

especialmente por la adopción de tecnologías como contenedores y servidores virtuales (RedHat, s. f.).

La virtualización es una tecnología que permite abstraer y dividir los recursos físicos de un sistema, como CPU, memoria y almacenamiento, para ser compartidos por múltiples entornos o usuarios al mismo tiempo. Esta técnica mejora el aprovechamiento del hardware, ya que permite ejecutar varios sistemas operativos o aplicaciones sobre un mismo dispositivo físico, sin interferencias entre sí. Existen diferentes tipos de virtualización, cada una adaptada a necesidades específicas:

- **De datos**: Agrupa diferentes fuentes de información en una sola vista lógica.
- **De escritorios**: Centraliza la gestión de múltiples entornos de usuario simulados.
- **De servidores**: Permite dividir un servidor físico en múltiples entornos virtuales especializados.
- **De sistemas operativos**: Posibilita la ejecución simultánea de varios SO en un único dispositivo.
- **De redes**: Desacopla funciones como direccionamiento IP o servicios de red para distribuirlas de forma independiente.

El componente clave en este proceso es el hipervisor, una capa de software que controla y gestiona los recursos del equipo físico. Este software actúa como intermediario entre el hardware y las máquinas virtuales (VMs), asegurando que cada una reciba los recursos necesarios sin afectarse mutuamente. Existen dos tipos de hipervisores:

- **Tipo 1 (Bare Metal)**: Se ejecutan directamente sobre el hardware físico y son más comunes en entornos empresariales o de centros de datos. Ejemplos: KVM,

VMware ESXi.

- **Tipo 2**: Se instalan como una aplicación dentro de un sistema operativo *host*, más comunes en equipos de escritorio para un solo usuario. Ejemplos: *Oracle VirtualBox, VMware Workstation.*

Ambos tipos permiten configurar recursos, crear múltiples entornos virtuales, y realizar pruebas o despliegues de software sin necesidad de múltiples equipos físicos (Susnjara y Smalley, 2024).

Clasificación

Las máquinas virtuales (VMs) se clasifican principalmente en dos grandes tipos, según su propósito y nivel de virtualización:

1. **Máquinas Virtuales de Sistema (System VMs):** Este tipo de VM emula un sistema físico completo, permitiendo ejecutar un sistema operativo independiente. Su funcionalidad es tan amplia que puede utilizarse para instalar diferentes SO dentro del equipo anfitrión, como correr *Linux* en un entorno *Windows*. Son especialmente comunes en entornos empresariales y centros de datos, donde permiten implementar múltiples servidores virtuales sobre un único hardware físico. Estas VMs se conocen también como máquinas de virtualización completa, porque permiten compartir recursos como CPU, memoria y almacenamiento entre varias máquinas virtuales independientes.
2. **Máquinas Virtuales de Proceso (Process VMs):** Diseñadas para ejecutar un único proceso o aplicación dentro de un entorno controlado, sin virtualizar el sistema completo. Son ideales para proporcionar un entorno aislado para ejecutar una aplicación, garantizando compatibilidad entre plataformas. El ejemplo más conocido

es la Máquina Virtual de Java (JVM), que permite ejecutar programas escritos en Java sin importar el sistema operativo anfitrión.

Estas categorías permiten adaptar la virtualización tanto a entornos de infraestructura completa como a necesidades más específicas, como la portabilidad y seguridad de aplicaciones individuales (Susnjara y Smalley, 2024).

Ventajas

Las máquinas virtuales ofrecen múltiples beneficios frente al uso exclusivo de *hardware* físico. Una de las ventajas más destacadas es la posibilidad de ejecutar múltiples sistemas operativos de forma simultánea en un solo equipo físico, optimizando el uso de recursos como CPU, memoria y almacenamiento.

Una de sus principales aplicaciones es la consolidación de servidores, que permite combinar diversas cargas de trabajo en un solo servidor, reduciendo la necesidad de adquirir nuevos equipos y ahorrando en consumo energético, espacio físico y refrigeración. Este enfoque también mejora la eficiencia operativa y los costos de TI.

Las VMs proporcionan entornos aislados, ideales para probar software o ejecutar aplicaciones críticas sin riesgo para el sistema anfitrión. Esta capacidad las hace útiles en entornos de desarrollo, pruebas, y recuperación ante fallos, ya que pueden ser clonadas, restauradas desde instantáneas, o eliminadas/reemplazadas rápidamente.

Además, su portabilidad permite trasladarlas entre equipos físicos o moverlas entre ambientes locales y la nube, lo que resulta valioso en escenarios de nube híbrida. También mejoran la flexibilidad para crear entornos bajo demanda, y la seguridad, al permitir escaneos externos de las VMs y una rápida recuperación en caso de infecciones.

Finalmente, desde una perspectiva ambiental, reducen la cantidad de hardware necesario, contribuyendo a una mayor sostenibilidad energética (Susnjara y Smalley, 2024).

Desventajas

No todo es positivo las VMs pueden presentar problemas de rendimiento, ya que comparten recursos físicos limitados con otras VMs. Además, requieren conocimientos técnicos avanzados para su configuración y administración.

También existe el riesgo de un punto único de falla, si el equipo físico que aloja todas las VMs falla, todas las máquinas virtuales que dependen de él podrían verse afectadas (Susnjara & Smalley, 2024).

Principales usos

Son herramientas versátiles que desempeñan un papel importante tanto en entornos empresariales como personales. Su capacidad de aislar entornos operativos y ofrecer flexibilidad las hace ideales para una amplia gama de aplicaciones, entre las que destacan:

Computación en la nube y nube híbrida

Las VMs son la base de las infraestructuras de nube pública, privada e híbrida, ya que permiten ejecutar y escalar múltiples aplicaciones simultáneamente. Gracias a su portabilidad, facilitan la migración rápida de cargas de trabajo desde entornos locales a la nube, y soportan arquitecturas híbridas combinando distintas plataformas en una única solución de TI.

Entornos de desarrollo y DevOps

En el ámbito del desarrollo de software, las VMs permiten a los equipos de *DevOps* crear plantillas con configuraciones específicas para pruebas, integración y automatización de procesos. Esto agiliza los flujos de trabajo y reduce errores durante el ciclo de vida del software.

Pruebas técnicas y seguridad

Las VMs son ideales para probar nuevos sistemas operativos sin comprometer el sistema principal, o para analizar malware en un entorno controlado. También son útiles para ejecutar software incompatible con el SO principal, permitiendo a los usuarios trabajar en múltiples plataformas desde un solo dispositivo.

Navegación segura

Usar una VM para navegar por internet de forma aislada protege al usuario de infecciones o ataques web. Es posible tomar instantáneas (*snapshots*) antes de cada sesión y restaurar el sistema en segundos si ocurre algún incidente.

Recuperación ante desastres (DR)

En situaciones críticas, las VMs permiten replicar o clonar sistemas en minutos, lo que facilita una rápida recuperación sin necesidad de aprovisionar servidores físicos desde cero. Esta agilidad es vital en entornos que exigen alta disponibilidad (Susnjara & Smalley, 2024).

Virtual box

VirtualBox es un *software* de virtualización multiplataforma desarrollado por *Oracle* que permite ejecutar múltiples sistemas operativos simultáneamente sobre un solo equipo físico. Se trata de un hipervisor de tipo 2, lo que significa que requiere un sistema operativo anfitrión para funcionar, a diferencia de los hipervisores de tipo 1 que operan directamente sobre el hardware.

Gracias a VirtualBox, los usuarios pueden crear máquinas virtuales (VM) para instalar sistemas como *Windows, Linux, macOS* y otros menos comunes (ej. DOS, Solaris, QNX). Su entorno aislado permite probar software, ejecutar aplicaciones incompatibles o crear espacios seguros (*sandbox*) para pruebas con malware u otros riesgos, sin afectar el sistema principal.

VirtualBox es gratuito para uso personal bajo la Licencia Pública General de GNU, aunque ofrece una versión Enterprise con características avanzadas como compatibilidad con RDP, USB 3.0, cifrado de disco, arranque por PXE/NVMe y soporte técnico.

Su flexibilidad lo hace ideal tanto para usuarios domésticos como para profesionales de TI y desarrolladores. Estos pueden crear entornos de desarrollo, laboratorios virtuales o incluso redes completas para pruebas. Además, permite clonar máquinas, usar imágenes ISO o discos virtuales (*VDI, VMDK*, etc.), y configurar recursos como CPU, memoria y gráficos de forma personalizada.

En cuanto a su arquitectura técnica, *VirtualBox* utiliza un Monitor de Máquina Virtual (VMM) que gestiona el comportamiento del código en los distintos niveles de privilegio de la CPU (anillos de protección), interceptando instrucciones de bajo nivel y optimizando su ejecución. Cuando se requiere, también recurre a emulación mediante QEMU, por ejemplo, en fases tempranas del arranque o con BIOS. Sin embargo, para mejorar el rendimiento, reemplaza ciertas rutas de código con llamadas directas al hipervisor.

La modularidad de *VirtualBox* permite que las máquinas e imágenes generadas sean compatibles entre sistemas operativos host distintos, haciendo que la portabilidad y administración sean consistentes sin importar la plataforma (Spiceworks, s.f.).

Usos de máquinas virtuales

1. **Ejecución de diferentes sistemas operativos**

Aunque *VirtualBox* debe instalarse en sistemas compatibles, permite ejecutar prácticamente cualquier sistema operativo siempre que el equipo anfitrión cuente con los recursos necesarios. Esto posibilita que los usuarios utilicen aplicaciones diseñadas para sistemas operativos específicos sin importar el sistema principal instalado en su equipo. Además,

VirtualBox ofrece la opción de configurar el hardware virtual de cada sistema operativo invitado, facilitando la instalación de sistemas antiguos como OS/2 o DOS, incluso si el hardware físico no es compatible. También es posible operar varios sistemas operativos a la vez, una función que resulta útil para diversos escenarios prácticos.

2. Facilidad en la instalación de software

Las máquinas virtuales simplifican la entrega de *software*, permitiendo a los proveedores distribuir configuraciones completas y listas para usar. Por ejemplo, la instalación y configuración de un servidor de correo electrónico, que normalmente es un proceso complicado, puede prepararse dentro de una máquina virtual y luego compartirse fácilmente. Así, el usuario solo debe importar y ejecutar dicha máquina virtual en *VirtualBox*, evitando todo el proceso manual de configuración.

3. Mejora en la redundancia y seguridad

Una máquina virtual funciona como un contenedor que puede iniciarse, pausarse, respaldarse y transferirse entre diferentes equipos. *VirtualBox* ofrece la función de instantáneas que permite guardar el estado de una máquina virtual y restaurarlo cuando se requiera. Esto facilita probar cambios o instalaciones sin riesgos y sin necesidad de realizar procesos largos de respaldo. Además, el aislamiento de las máquinas virtuales protege el sistema principal, permitiendo a profesionales de seguridad o desarrolladores experimentar o probar amenazas sin comprometer el equipo anfitrión.

4. Desarrollo y pruebas multiplataforma

La virtualización facilita el desarrollo de software para múltiples sistemas operativos en una misma máquina. Por ejemplo, se puede desarrollar una aplicación con versiones para móviles y escritorio sin necesidad de cambiar físicamente de dispositivo. También se puede

compilar software para diferentes plataformas (como archivos *APP* para *macOS o EXE* para Windows) desde un solo equipo, eliminando la necesidad de sistemas de arranque dual.

5. Reducción de costos en infraestructura

Las computadoras modernas suelen utilizar solo una fracción de su capacidad de procesamiento, lo que hace que implementar máquinas virtuales en un entorno empresarial pueda optimizar los recursos. Ejecutar múltiples máquinas virtuales en pocos hosts potentes reduce el costo de infraestructura, al aprovechar mejor el hardware disponible y minimizar el gasto asociado con los dispositivos finales de los usuarios (Spiceworks, s. f.).

Factores a tener en cuenta antes de utilizar máquinas virtuales

En muchas situaciones, es posible utilizar la infraestructura en la nube ofrecida por proveedores como *Arsys*, lo que simplifica tanto la infraestructura necesaria como su administración. Sin embargo, si se opta por gestionar hardware propio para implementar virtualización, es fundamental evaluar ciertos aspectos clave:

Requisitos de hardware y software

Antes de proceder con la virtualización, es imprescindible analizar las características y capacidades del hardware disponible, ya que no todos los equipos soportan esta tecnología. Asimismo, es necesario contar con un software de virtualización compatible con el hardware para garantizar un funcionamiento adecuado y eficiente.

Selección del hipervisor apropiado

Existen diferentes tipos de hipervisores, cada uno adecuado para distintos escenarios y necesidades. Por ello, elegir el hipervisor correcto resulta esencial, considerando factores como el sistema operativo que se va a virtualizar, las características del hardware y el presupuesto disponible para el proyecto (García, 2024).

1. Python

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general, lo que significa que puede ser utilizado para crear desde aplicaciones sencillas hasta sistemas complejos. Fue diseñado con un fuerte enfoque en la claridad y legibilidad del código, lo cual facilita tanto el aprendizaje como la colaboración entre desarrolladores.

Una de sus principales características es su sintaxis simple y clara, que prescinde del uso de llaves para estructurar bloques de código, utilizando en su lugar la indentación. Esta particularidad, además de hacer que el código sea más limpio visualmente, promueve una escritura ordenada y coherente.

Python soporta múltiples paradigmas de programación, incluyendo la programación orientada a objetos, la programación imperativa, e incluso algunos elementos de programación funcional. Esto le da gran flexibilidad y adaptabilidad a distintos estilos de desarrollo.

Desde su creación en 1991 por Guido van Rossum, Python ha evolucionado constantemente, incorporando nuevas funcionalidades modernas y convirtiéndose en una herramienta clave para múltiples disciplinas, incluyendo el desarrollo web, la ciencia de datos, la automatización, la inteligencia artificial, la visualización de datos y más.

Además, Python destaca por contar con una amplia biblioteca estándar y un ecosistema robusto de librerías de terceros, lo que facilita el desarrollo de todo tipo de soluciones sin tener que crear funcionalidades desde cero. Esto se complementa con una comunidad activa y global, que proporciona constante apoyo, documentación, tutoriales y herramientas de desarrollo.

En resumen, Python es una herramienta poderosa, versátil y fácil de usar, ideal tanto para programadores novatos como para profesionales que trabajan en proyectos complejos y avanzados (García, s.f.).

Características Principales

- **Legibilidad y simplicidad**: Su código es fácil de escribir y entender gracias a su estructura basada en indentaciones, en lugar de llaves, como en otros lenguajes.
- **Lenguaje interpretado**: No necesita ser compilado antes de ejecutarse, lo cual agiliza el desarrollo.
- **Tipado dinámico**: No es necesario declarar el tipo de las variables, lo que facilita la programación rápida.
- **Multiplataforma**: Funciona sin problemas en sistemas como Windows, macOS y Linux.
- **Gran biblioteca estándar y librerías de terceros**: Posee módulos incorporados para múltiples tareas y una gran cantidad de recursos desarrollados por la comunidad.
- **Versatilidad**: Se usa en diversas áreas como desarrollo web, análisis de datos, inteligencia artificial, automatización, videojuegos, entre otros.
- **Extensible e integrable**: Se puede complementar con bibliotecas en otros lenguajes como C, C++ o Java.
- **Comunidad activa**: Tiene una de las comunidades más grandes, lo cual garantiza documentación abundante, foros de ayuda y continua mejora del lenguaje.

Ventajas frente a otros lenguajes

- **Fácil de aprender**: Ideal para principiantes gracias a su sintaxis amigable y su enfoque en la legibilidad.
- **Alta productividad**: Permite lograr más con menos líneas de código.
- **Amplia documentación y soporte**: La gran comunidad provee guías, tutoriales y

recursos en constante crecimiento.

- **Código abierto**: Es gratuito y mantenido por una comunidad global.
- **Popularidad**: Ocupa los primeros lugares en rankings como TIOBE y encuestas como la de *Stack Overflow* (García, s.f.).

Áreas de Aplicación

- **Desarrollo web**: Con *frameworks* como *Django, Flask o FastAPI*.
- **Análisis de datos y ciencia de datos**: Usando herramientas como Pandas, *NumPy, Dask y SciPy.*
- **Automatización de tareas**: Scripts sencillos para procesos repetitivos.
- **Inteligencia Artificial y Machine Learning**: Soporte de bibliotecas como *TensorFlow, PyTorch y scikit-learn*.
- **Desarrollo de videojuegos y realidad virtual**: Con bibliotecas como *Pygame o Panda3D.*
- **Aplicaciones móviles**: Aunque menos común, es posible usar Python mediante herramientas como Kivy o BeeWare (Summer, s.f.).

2. Splunk

Splunk es una plataforma integral de análisis de datos orientada a la gestión de registros generados por máquinas. En el ámbito de la ciberseguridad, se destaca como una herramienta SIEM (*Security Information and Event Management*) que permite recopilar, indexar y analizar datos en tiempo real provenientes de diversas fuentes como servidores, dispositivos IoT, aplicaciones, redes y más. Su propósito principal es detectar, correlacionar, visualizar y responder a amenazas de seguridad de manera automatizada, de esta manera se detalla las ventajas y desventajas.

Ventajas

- Escalabilidad: Adaptable a organizaciones de diversos tamaños y necesidades.
- Flexibilidad: Compatible con una amplia gama de fuentes de datos y tecnologías.
- Automatización: Reducción del tiempo de respuesta mediante la automatización de tareas repetitivas.

Desventajas

- Costo: La implementación y mantenimiento pueden ser costosos, especialmente para grandes volúmenes de datos.
- Complejidad: Requiere personal capacitado para su configuración y administración óptima (Splunk Cisco, s.f.).

Con lo expuesto se puede indicar que sus funcionalidades clave se incluyen la centralización de registros, la correlación de eventos para detectar anomalías, la emisión de alertas automáticas, la creación de paneles interactivos para monitoreo, y la automatización de respuestas ante incidentes mediante la integración con soluciones como *Splunk SOAR*. Además, al generar reportes de auditoria detallados facilita el cumplimiento normativo con marcos como:

- **ISO 27001**: Gestión de la seguridad de la información.
- **NIST**: Instituto Nacional de Estándares y Tecnología
- **COBIT**: Objetivos de control para la información y tecnologías relacionadas
- **PCI-DSS**: Seguridad de datos para la industria de tarjetas de pago.
- **HIPAA**: Regulaciones para la protección de información médica en EE.UU.
- **GDPR**: Reglamento General de Protección de Datos de la Unión Europea (CyberArrow team, 2025).

Splunk es valorado por su versatilidad, escalabilidad y capacidad de personalización. Permite operar tanto en entornos locales como en la nube y se integra con numerosas tecnologías. Sin embargo, también presenta desventajas, como un alto consumo de almacenamiento, costos elevados y cierta complejidad en su implementación y gestión.

Desde la perspectiva de la seguridad, *Splunk* mejora la visibilidad de los eventos dentro de la infraestructura TI, permite reaccionar proactivamente ante amenazas mediante la correlación de eventos y la automatización de respuestas, y fortalece la resiliencia digital de las organizaciones. Su arquitectura modular y su ecosistema de aplicaciones extensibles lo convierten en una solución eficaz tanto para el análisis de amenazas como para el cumplimiento y la inteligencia de negocios (Splunk Cisco, s. f.).

3. GNS3 *(Graphical Network Simulator-3)*

GNS3 es una plataforma de simulación de redes de código abierto que permite diseñar, emular y probar topologías de red complejas sin necesidad de hardware físico. Es ampliamente utilizada por profesionales de redes, estudiantes y empresas para la formación, pruebas de concepto y preparación para certificaciones como CCNA, CCNP y CCIE, de esta manera permite crear laboratorios virtuales que simulan el comportamiento real de dispositivos de red como *routers*, switches, firewalls y servidores. Utiliza emuladores como *Dynamips* (para IOS de Cisco), *QEMU y VirtualBox* para ejecutar imágenes reales de sistemas operativos de red. Esto facilita la experimentación y el aprendizaje en un entorno controlado y sin riesgos para la infraestructura real (Walton, 2024).

Ventajas

- **Simulación de Redes:** Dispositivos Cisco como *routers* y *switches*, protocolos de red, como TCP/IP, OSPF, EIGRP, servicios de red como *firewalls*, VPN, seguridad y de comunicación como *gateways y PBX.*
- **Interfaz gráfica intuitiva**: Permite diseñar topologías de red arrastrando y soltando dispositivos virtuales.
- **Compatibilidad con múltiples fabricantes**: Soporta dispositivos de Cisco, Juniper, MikroTik, Fortinet, entre otros.
- **Integración con herramientas externas**: Se puede integrar con Wireshark para análisis de tráfico y con VirtualBox o VMware para ejecutar máquinas virtuales.
- **Simulación en tiempo real**: Permite observar el comportamiento de la red en tiempo real, facilitando la detección y resolución de problemas.
- **Escalabilidad**: Capacidad para simular desde pequeñas redes hasta infraestructuras complejas.

Desventajas

- **Requisitos de hardware**: La emulación de múltiples dispositivos puede requerir un equipo con recursos significativos.
- **Curva de aprendizaje**: La configuración inicial y el manejo de imágenes IOS pueden ser complejos para principiantes.
- **Licenciamiento de imágenes**: Es necesario contar con las imágenes de los sistemas operativos, lo que puede implicar restricciones legales o de acceso (GNS3, s.f.).

La integración de *Splunk* con dispositivos IoT dentro de máquinas virtuales (VMs) representa una solución versátil, escalable y segura para el monitoreo inteligente de entornos

conectados. Este enfoque combina la potencia del análisis de datos en tiempo real que ofrece *Splunk*, con la flexibilidad que proporciona la virtualización para desplegar sistemas operativos aislados y configurables.

Al utilizar máquinas virtuales como plataforma base, se consigue un entorno controlado en el que es posible instalar y ejecutar *Splunk* sin comprometer el sistema operativo principal. Esto no solo facilita la portabilidad y replicación del entorno, sino que permite también la escalabilidad horizontal, donde nuevas instancias pueden desplegarse rápidamente en función de la carga de dispositivos o volumen de datos generado por los sensores IoT.

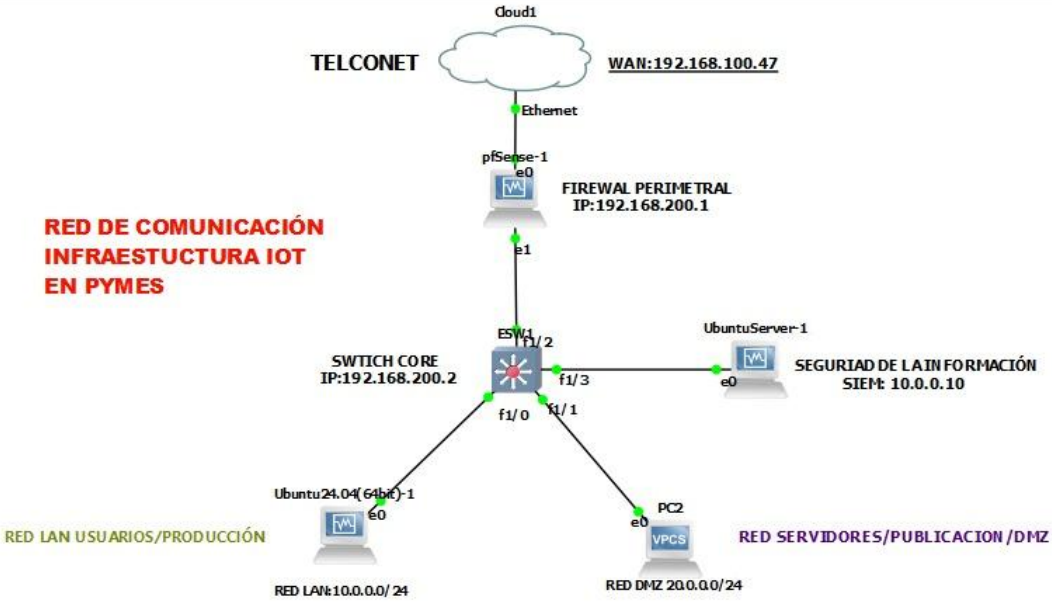
La virtualización también permite emular o probar múltiples dispositivos IoT en paralelo, lo cual resulta útil para validar configuraciones, políticas de seguridad o integraciones antes de aplicarlas en producción. Además, el uso de VMs proporciona aislamiento, un aspecto crítico en ciberseguridad, ya que en caso de vulnerabilidades en el software de los dispositivos IoT o en sus protocolos (como MQTT o SNMP), el impacto puede contenerse dentro de la máquina virtual.

Desde el punto de vista operativo, la combinación de estas tecnologías permite a las organizaciones centralizar la recolección, análisis y visualización de eventos generados por los dispositivos, facilitando el diagnóstico de fallos, la detección de anomalías y la reacción ante incidentes de forma automatizada.

Finalmente, el despliegue de *Splunk* sobre VMs facilita su integración en arquitecturas de nube híbrida, permitiendo mover instancias entre entornos locales y remotos, optimizando recursos y reduciendo costos de infraestructura. Esta arquitectura se alinea perfectamente con las necesidades de las pequeñas y medianas empresas (PYMEs), que buscan soluciones ágiles y eficientes para mejorar su postura de seguridad en un entorno tecnológico cada vez más conectado y expuesto.

Figura 4

Diagrama de Red de la Simulación IOT – Splunk



Nota: En la Figura se observa la red de comunicación de la infraestructura IoT en PYMEs, siendo las bases la red de servidores / publicación /DMZ.

4. Ubunto

Ubuntu es un sistema operativo gratuito y de código abierto basado en Linux, desarrollado por Canonical Ltd. Se caracteriza por su estabilidad, seguridad y facilidad de uso, lo que lo hace ideal tanto para equipos personales como servidores. Posee versiones específicas para diferentes entornos, incluyendo *Ubuntu Core*, diseñada para dispositivos IoT.

Ofrece compatibilidad con una amplia gama de hardware, una comunidad activa de soporte, actualizaciones frecuentes y una gestión eficiente de recursos, lo que permite su implementación incluso en máquinas virtuales de bajo rendimiento. Estas cualidades lo convierten en una opción óptima para laboratorios de simulación y entornos de monitoreo de seguridad como el desarrollado en este proyecto (Piensa Solutions, s.f.).

Ahora con respecto a la versión que se ocupara en la *Ubuntu* 24.04 LTS incorpora importantes novedades técnicas que fortalecen su rendimiento y estabilidad, destacando el uso del *Kernel Linux* 6.8, el sistema de arranque *systemd* 255 y la pila gráfica Mesa 24.0. También se integra el nuevo gestor de red *Netplan* 1.0, ahora como *backend* predeterminado para *Network Manager*, lo que mejora la gestión de conexiones en entornos virtualizados y distribuidos, como los utilizados en pruebas con IoT.

Incluye versiones actualizadas de herramientas esenciales como *Python 3.12, Open JDK 21, Rust 1.75* y GCC 14, lo cual facilita el desarrollo y ejecución de aplicaciones modernas, incluyendo scripts de simulación en proyectos académicos o de ciberseguridad.

Otra novedad relevante es la incorporación de *Thunderbird* como paquete Snap, siguiendo la misma estrategia que Firefox, y el estreno de *Firmware Updater*, herramienta útil para la gestión de actualizaciones de hardware, aunque disponible solo en algunas ediciones. El nuevo instalador gráfico desarrollado con *Flutter* mejora significativamente la accesibilidad y experiencia de instalación, incluyendo opciones mínimas o completas para la selección de aplicaciones, lo cual se adapta fácilmente a distintos casos de uso, como servidores de pruebas o entornos de laboratorio (Muylinux, 2024).

CAPITULO 3

DESARROLLO

Documento Técnico: Sistema de Monitoreo de Seguridad con *Splunk* para IoT en PYMEs del Ecuador

3.1. Arquitectura General del Sistema

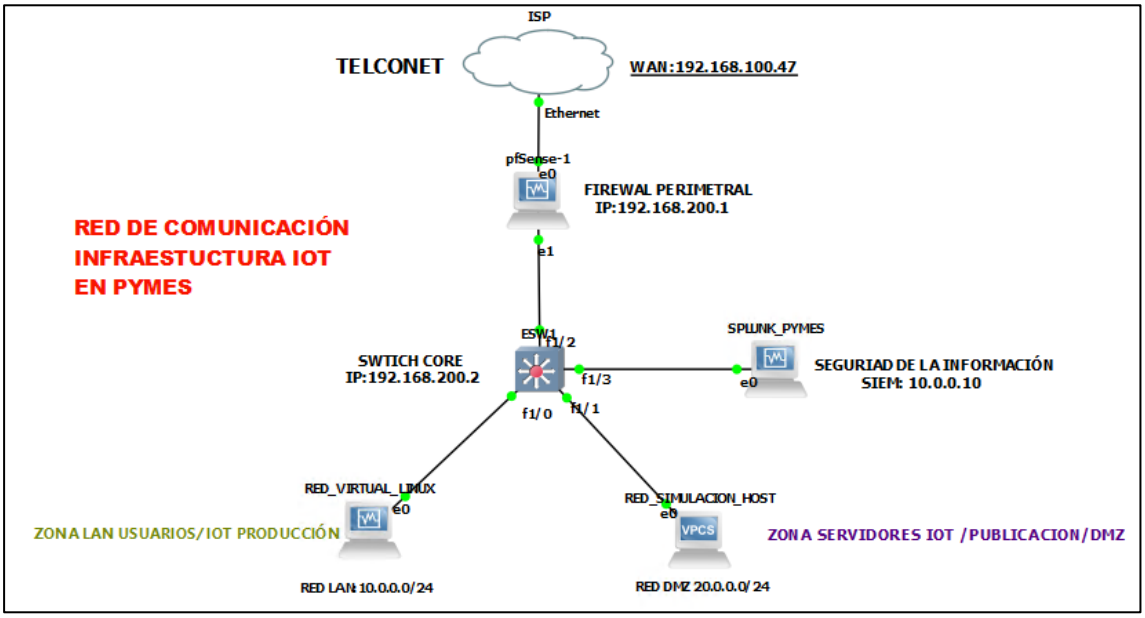
Para la arquitectura de red de una PYME en el Ecuador en base al capítulo anterior los elementos básicos que se han tomado para el diseño del proyecto consta de un ISP (Proveedor de servicio de internet), un Firewall perimetral que comunica la red interna con la externa interconectado con un *switch* capa 3 como el Core de la empresa, switches de distribución, estaciones de trabajo, servidores locales y de dispositivos IoT que para este diseño son virtualizados asumiendo las funcionalidades de un equipo real, tales como cámaras IP, sensores ambientales, controladores de acceso o sistemas de automatización. Estos dispositivos, están conectados en base a una segmentación general IOT para el control centralizado y de seguridad evitando puntos vulnerables dentro de la infraestructura. Todo este proceso se lo ha realizado en GNS3, descrito en la Figura 5, como parte fundamental del diseño de este proyecto técnico integrando a SIEM (Splunk) en la arquitectura de red.

3.1.1 Diagrama de red

La red simulada en GNS3 consta de los componentes y servicios principales de una infraestructura de red IOT la cual es la siguiente:

Figura 5

Diagrama de Red de simulada



Nota. En la Figura se mira el diagrama de red simulado donde la red LAN y DMZ .

- Firewall Perimetral (Pfsense)
- Un switch core (Cisco L3)
- Un SIEM (Splunk)
- Un sensor IoT simulando temperatura y humedad (vía MQTT)
- Una cámara IP IoT (Syslog)
- Un servidor Linux (con SNMP activado)
- Componentes generales red de comunicación simulados (Host/Servidores/Usuarios/Wifi)

3.1.2. Protocolos involucrados

Contextualizando la arquitectura de red para este proyecto se identifica que el diseño considera la integración de protocolos de monitoreo como SNMP para equipos tradicionales y MQTT para sensores y dispositivos IoT descritos en este capítulo. Además, es importante establecer que topología lógica permite recolectar datos desde cada zona de red hacia el servidor central de monitoreo (SIEM) basado una solución *Splunk*. Por lo que el SIEM actúa como punto

de correlación y análisis, permitiendo a este tipo de empresas (PYMEs) contar con sistema de ciberseguridad que registre eventos, detecte anomalías y genere alertas de seguridad en tiempo real y permita tomar acciones preventivas y correctivas. Como se observa en la tabla 4

Tabla 4

Protocolos involucrados

Componente	Protocolo	Descripción
Sensor IoT	MQTT	Publica datos de temperatura y humedad
Cámara IP	Syslog	Genera alertas o eventos del sistema
Servidor Linux	SNMP	Uptime, Exposición de métricas y recursos sistema operativo
Switch L3	Syslog	Uptime, Envío logs de red, operación y configuración
Splunk Server	MQTT / Syslog / SNMP	Recolección y visualización eventos de seguridad, Gestión de alertas

Nota. En la Tabla se describen los protocolos y componentes de la arquitectura de red.

3.2. Implementación Técnica

3.2.1. Instalación de Splunk

Splunk es una plataforma poderosa de análisis de datos que permite recopilar, indexar y visualizar grandes volúmenes de datos generados por máquinas en tiempo real. Utilizada principalmente en entornos de TI, seguridad y monitoreo de infraestructura, *Splunk* facilita la detección de incidentes, el análisis forense, la generación de alertas y la creación de paneles interactivos. Su capacidad para integrar múltiples fuentes de datos, como registros de red, dispositivos IoT, aplicaciones y sistemas operativos, lo convierte en una herramienta clave para la observabilidad y la inteligencia operativa en organizaciones modernas (Splunk, 2025).

3.2.1.1.Requerimientos.

Máquina Virtual: Distribución de Linux Server (Ubuntu, CentOs, Fedora)

RAM: 4 GB

Espacio Disco: 70 GB

Processor: Quad-core processor 2

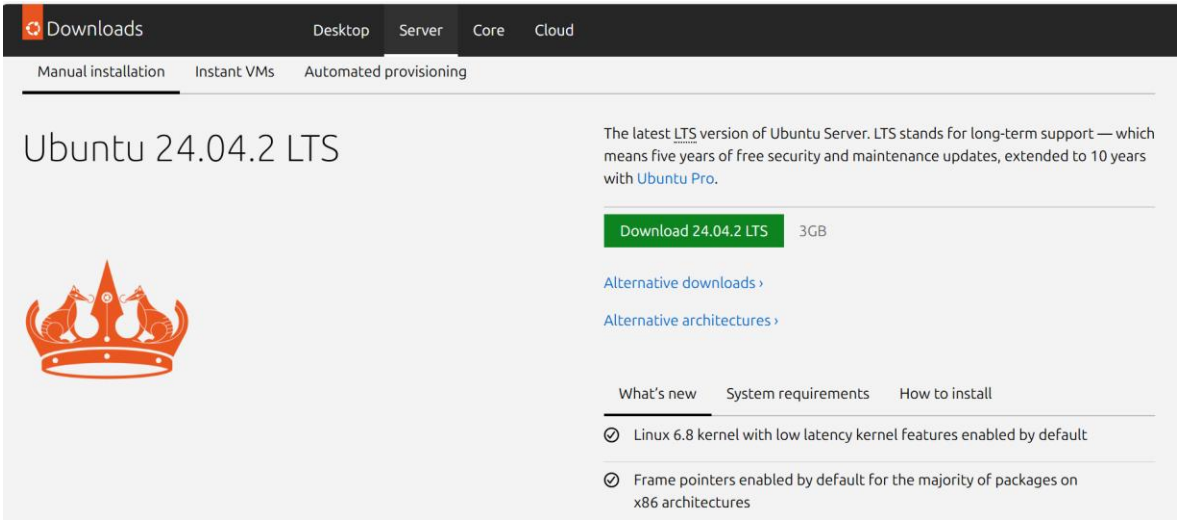
Network: Mode Bridge

3.2.1.2. Instalación Máquina Virtual

- Se usa la distribución de Ubuntu Server, para ello descargamos la .iso de su página oficial.

Figura 6

Ubuntu Server

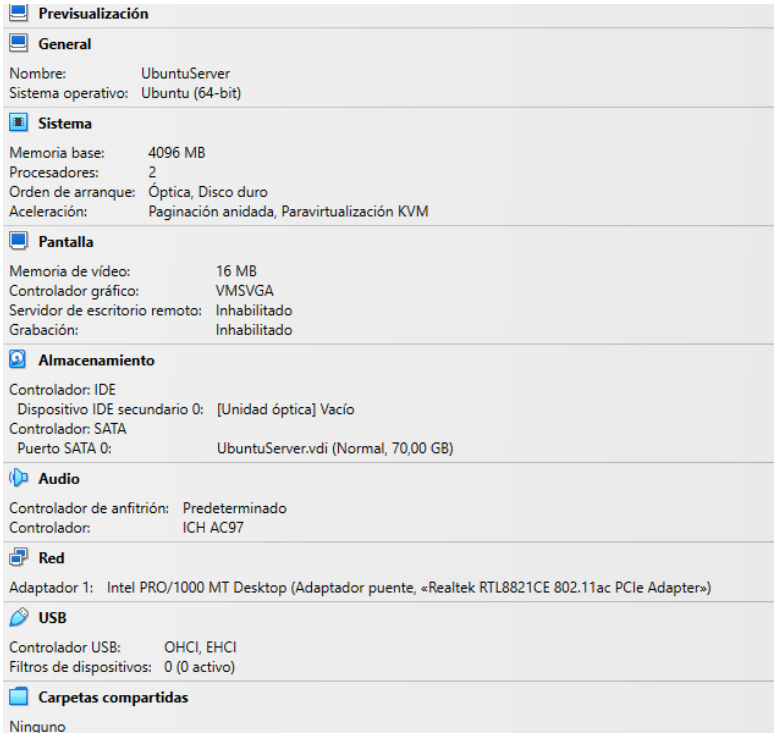


Nota. En la Figura se observa la página de descarga de 24.04.2 LTS Ubuntu.

- Se inicia VirtualBox y se crea una nueva máquina virtual con los parámetros establecidos en los requerimientos.

Figura 7

Creación de máquina virtual

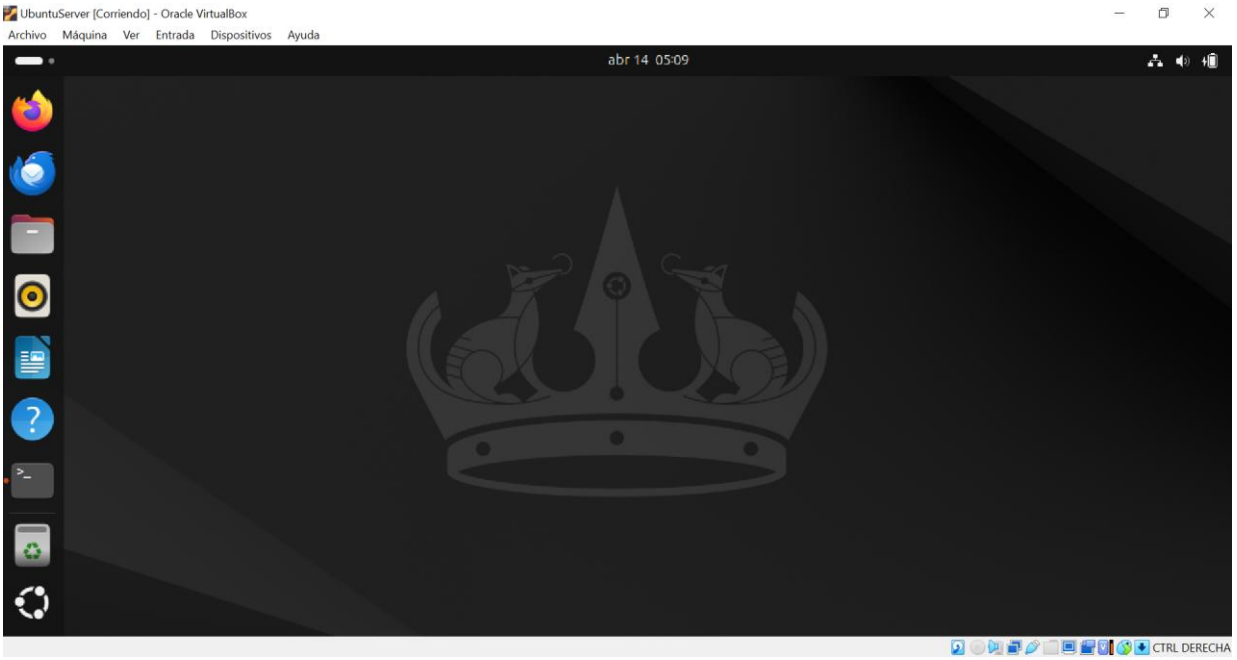


Nota. En la Figura se previsualiza la creación de la máquina virtual

- Una vez parametrizado cargamos la .iso e inicia el proceso de instalación, conservamos las configuraciones por default del servidor

Figura 8

Carga de .iso



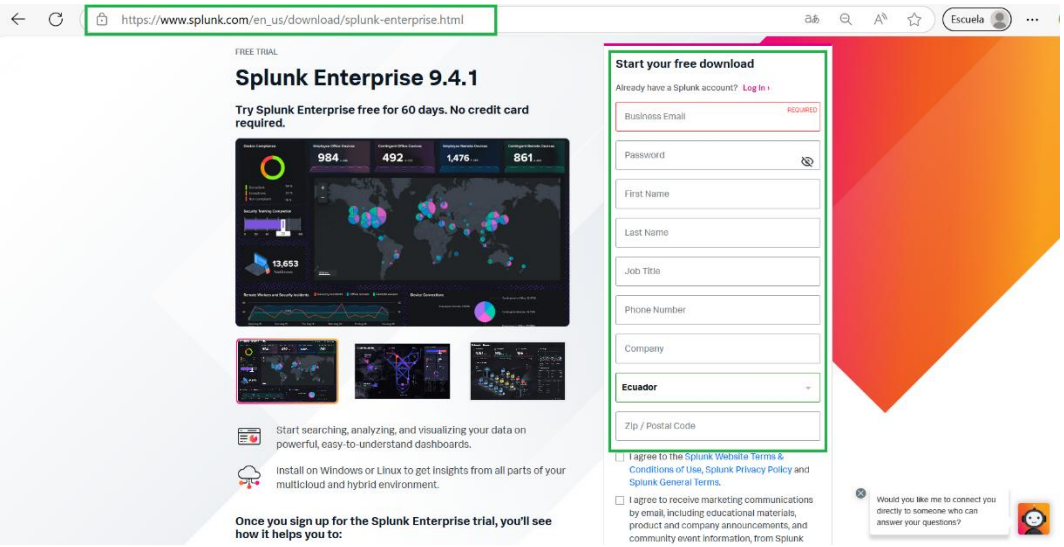
Nota. En la Figura se mira la carga de .so y el proceso de instalación.

3.2.1.3. Instalación Splunk

- Se descargar la versión Free Trial de Splunk Enterprise, Creamos un nuevo usuario en la página oficial de *splunk* e iniciamos sesión

Figura 9

Instalación de Splunk

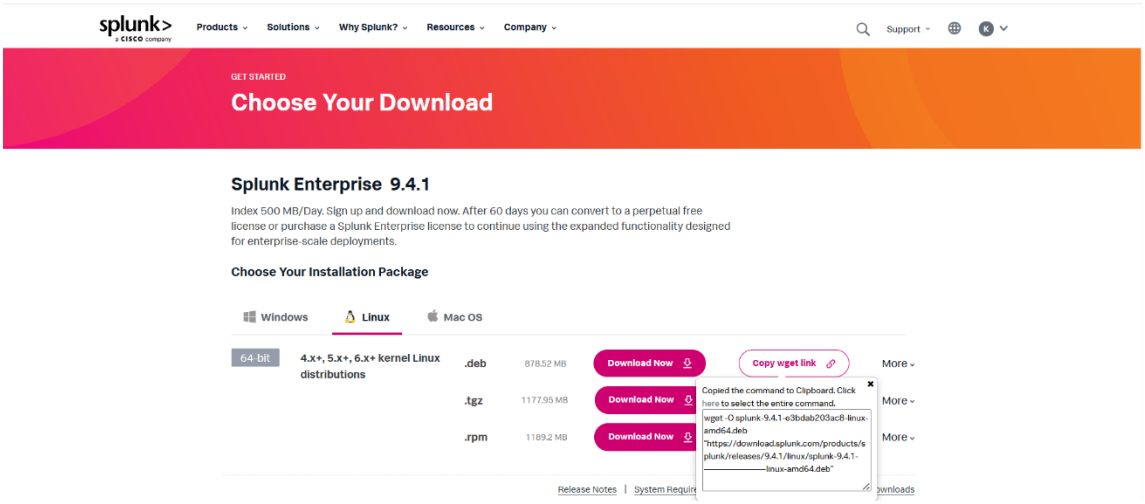


Nota. En la Figura se mira el link de instalación de splunk Enterprise 9.4.1. Tomado de: Splunk (2025)

- Una vez iniciado sesión, en la sección de descarga se selecciona la opción de Linux con paquete .deb y copiamos el enlace get

Figura 10

Selección de opción Linux



Nota. En la Figura se mira la descarga de selección de paquete y enlace get. Tomado de: Splunk (2025)

- Abrimos terminal sobre nuestro servidor de Ubuntu, con usuario root y ejecutamos el comando copiado del anterior paso

Figura 11

Servidor Ubuntu

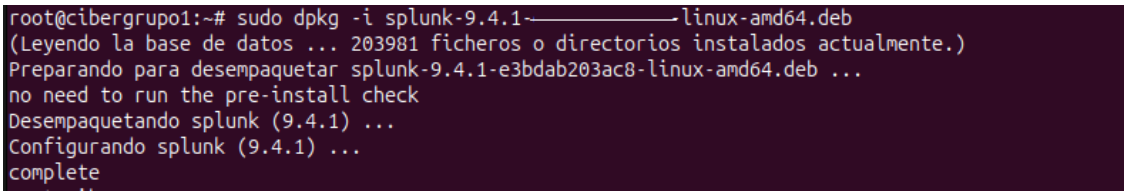
```
root@cibergrupo1:~# wget -O splunk-9.4.1-e3bdab203ac8-linux-amd64.deb "https://download.splunk.com/products/splunk/releases/9.4.1/linux/splunk
-9.4.1————————————linux-amd64.deb"
```

Nota. En la Figura se mira el terminal del servidor de Ubunto con usuario root.

- Una vez finalizado la descarga ejecutamos el siguiente comando para la instalación

Figura 12

Comando de instalación

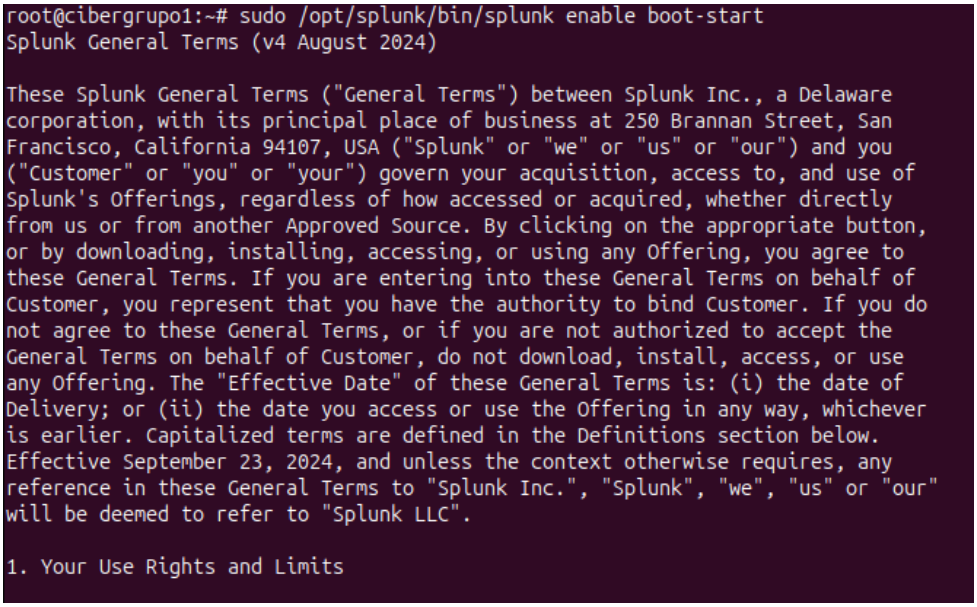


Nota. En la Figura se mira el comando de instalación.

- Se inicia para primer uso ejecutando el siguiente comando, una vez nos aparecerá los términos y condiciones lo cual debemos ir hasta el final para aceptar.

Figura 13

Inicio de primer uso

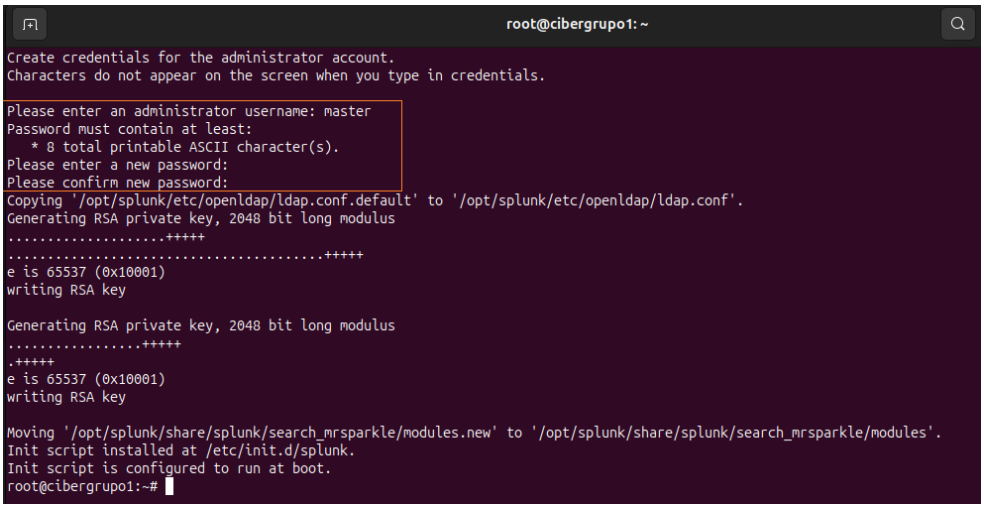


Nota. En la Figura 13 se mira el primer uso con la ejecución del comando.

- Una vez aceptado parametrizamos credenciales de acceso para primer uso

Figura 14

Acceso a primer uso

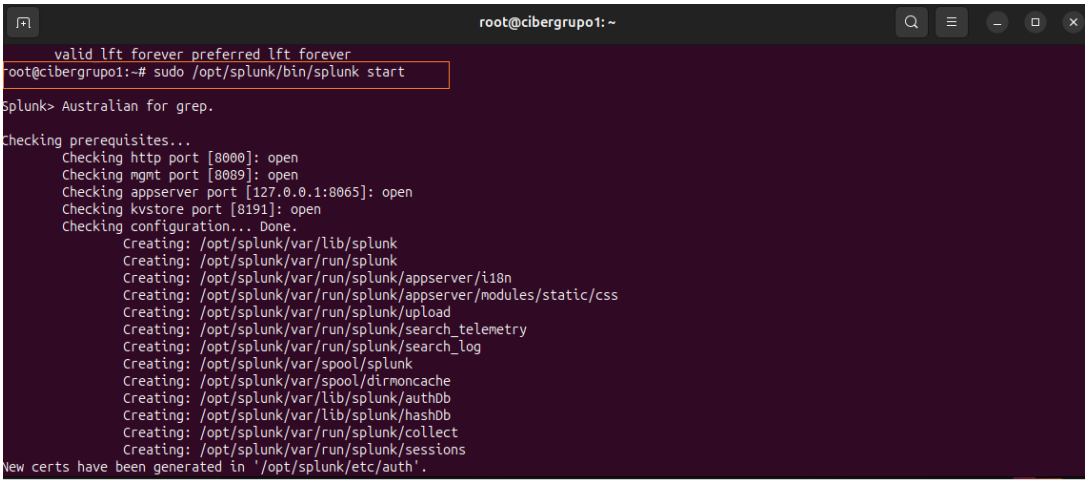


Nota. En la Figura se mira el acceso al primer uso.

- Se inicia los servicios para validar funcionamiento con el comando resaltado.

Figura 15

Inicio de servicio

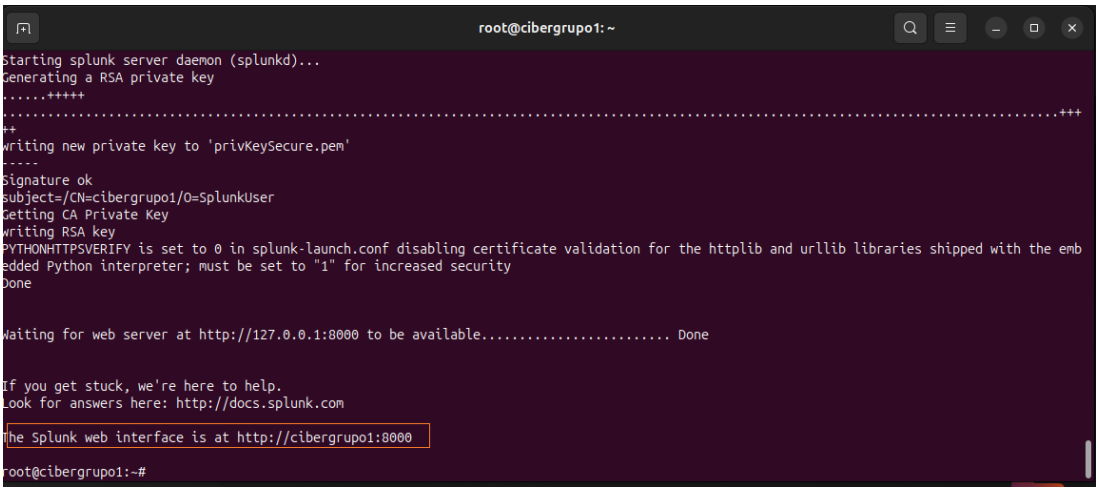


Nota. En la Figura se inicia el servicio y se prueba el funcionamiento del comando.

- Si todo sale correcto nos muestra la url de acceso, hacemos clic para validar el ingreso.

Figura 16

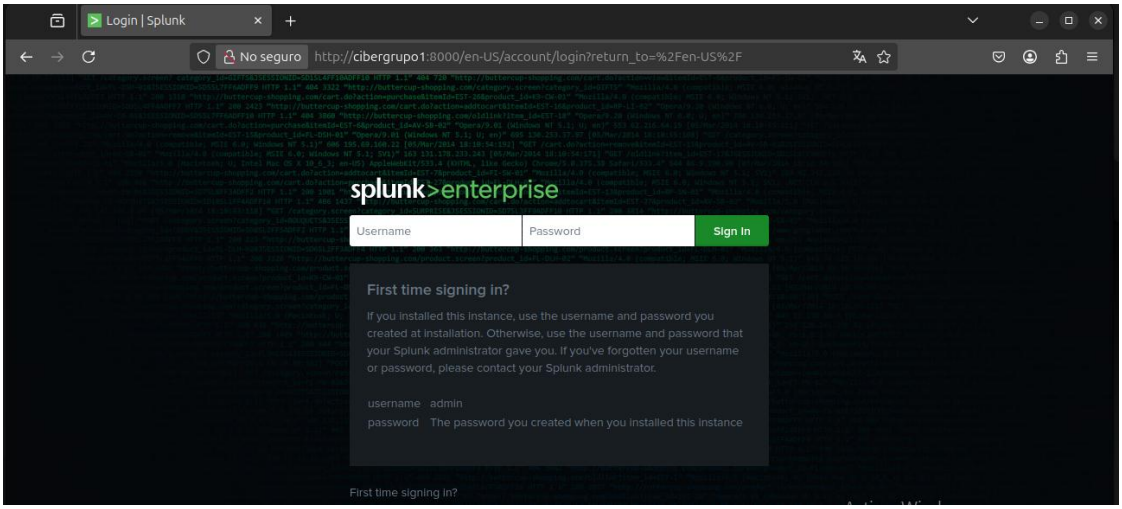
Url de acceso



Nota. En la Figura se mira la validación del acceso.

Figura 17

Validación del acceso

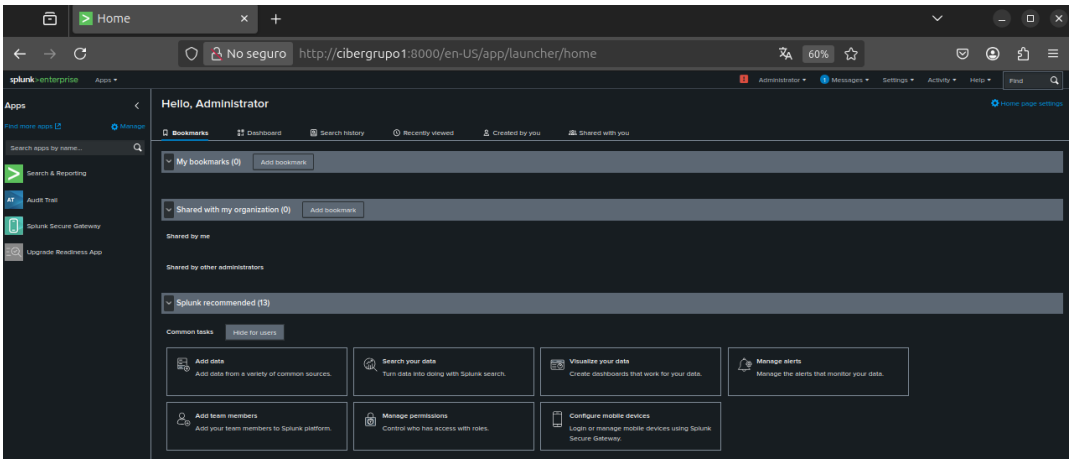


Nota. En la Figura se muestra la página de acceso de Splunk

- Ingresamos las credenciales parametrizadas en el punto 6 y certificamos acceso

Figura 18

Ingreso de credenciales



Nota. En la Figura se muestra la página de inicio por defecto de Splunk

- PD: Se tiene acceso desde una máquina local conectada a la misma red de la máquina virtual.
- Se habilitaron entradas por:
    - UDP puerto 514 para Syslog
    - HTTP Event Collector (HEC) para datos MQTT
    - Archivos monitoreados para traps SNMP

3.2.2. Instalación IoT en la infraestructura de red

La arquitectura del entorno de simulación IoT se compone de dos elementos fundamentales: Un servidor y un cliente. Cada uno cumple un rol específico dentro del flujo de datos que permite la generación, transmisión, recolección y visualización de eventos simulados.

Tabla 5

Instalación IoT en infraestructura de red

Rol	IP	Componente
Servidor	10.0.0.10	Splunk + Mosquitto (Broker MQTT)
Cliente	10.0.0.12	Scripts Python: simuladores IoT + script puente MQTT-Splunk
Cliente	10.0.0.190	Simulador Cámara IoT Syslog

Nota. En la Tabla se detalla el rol, IP y componente de la infraestructura de red

3.2.2.1. Habilitation HTTP Event Collector (HEC).

Para permitir que *Splunk* reciba eventos desde dispositivos externos, se habilitó el HTTP Event Collector (HEC). Este componente permite el ingreso de datos a través del protocolo HTTP, lo que lo hace ideal para integraciones con sistemas IoT.

Durante la configuración, se generó un token único para iot_simulacion asociado al índice *main*, que será utilizado por el script puente para enviar eventos. Se verificó que el HEC estuviera activado correctamente en *Splunk* y que el puerto 8088, necesario para la recepción de los datos, se encontrara habilitado y accesible.

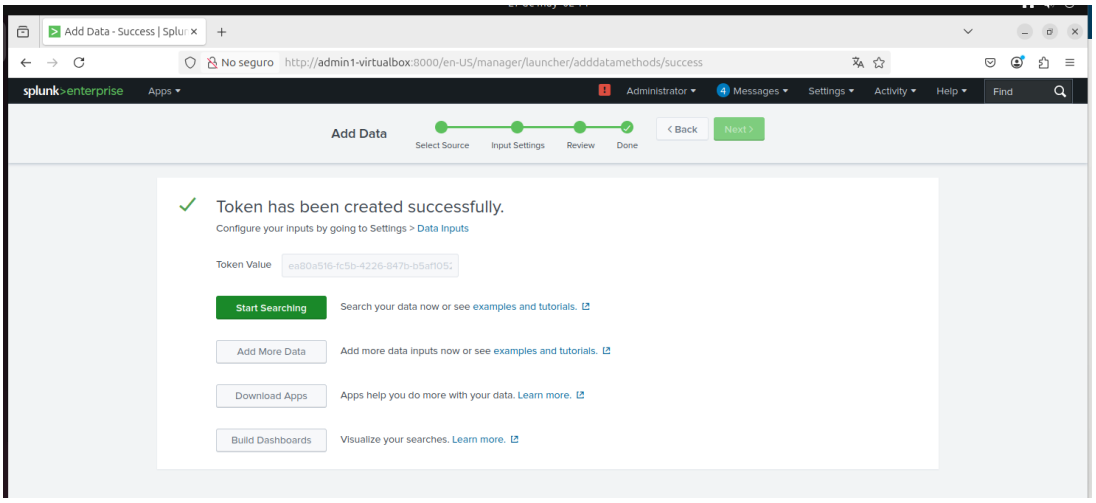
Esta configuración garantiza la recepción de eventos estructurados desde el entorno simulado y su posterior análisis en tiempo real desde la interfaz de búsqueda de *Splunk*. En resumen, se cumple:

- Creación de un token único iot_simulacion con índice main.
- Verificación HEC habilitado.

- Habilitación del puerto de comunicación 8088

Figura 19

Generación correcta de Token de HEC



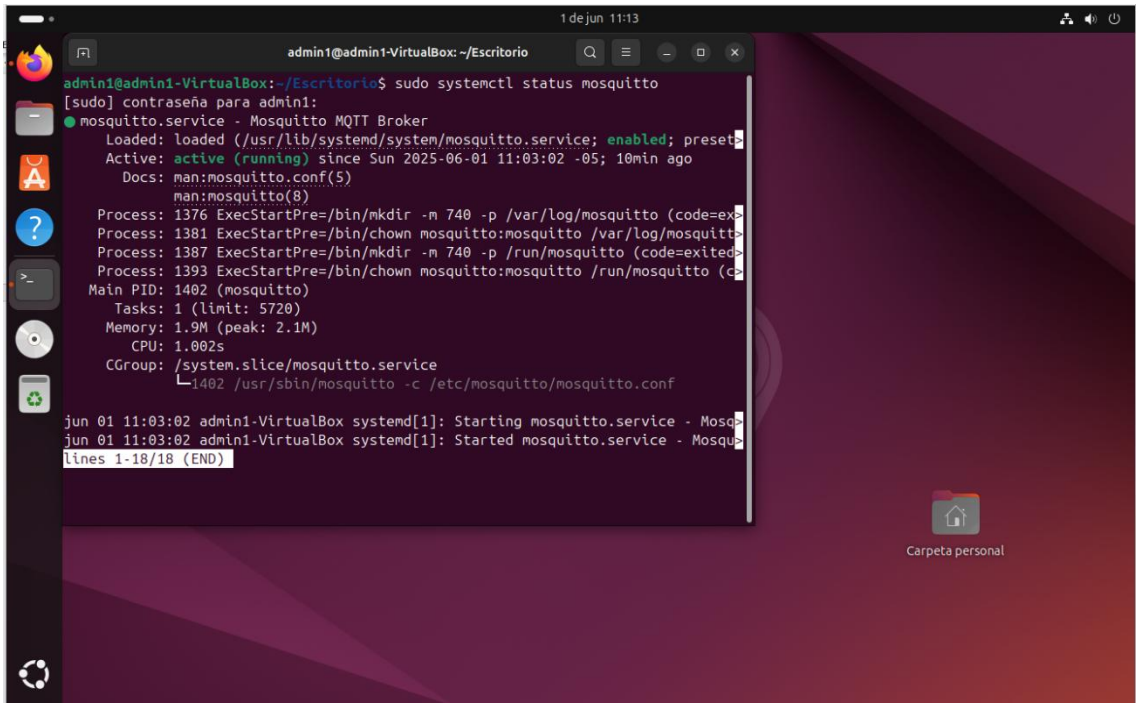
Nota. En la Figura se muestra la página de éxito de creación de Token

3.2.2.2. Instalación del broker Mosquitto.

La instalación de Mosquitto como *broker* MQTT es una parte principal en soluciones de IoT, como la propuesta en este TFM, donde actúa como intermediario entre los dispositivos simulados y *Splunk*. Su rendimiento y compatibilidad lo convierten en una herramienta para garantizar la transmisión eficiente de datos en entornos heterogéneos.

Figura 20

Instalación y habilitación de Mosquitto en el servidor Ubunto



Nota. En la figura se observa la instalación y habilitación correcta de Mosquitto en el servidor Ubunto

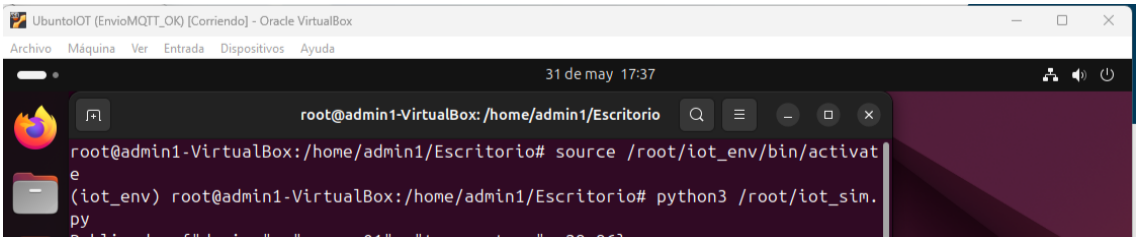
3.2.2.3 Entornos virtuales.

Se implementó un entorno simulado basado en Python utilizando *venv*, una herramienta que permite crear entornos virtuales aislados para ejecutar scripts de simulación IoT de manera controlada y eficiente.

Este entorno se desplegó en una máquina virtual cliente y permitió gestionar de forma segura las bibliotecas necesarias para la comunicación entre dispositivos simulados y el sistema *SIEM Splunk*. Con ell garantiza la portabilidad, estabilidad y trazabilidad del entorno, evitando conflictos con el sistema base y facilitando futuras extensiones.

Figura 21

Ejecución correcta del entorno virtual



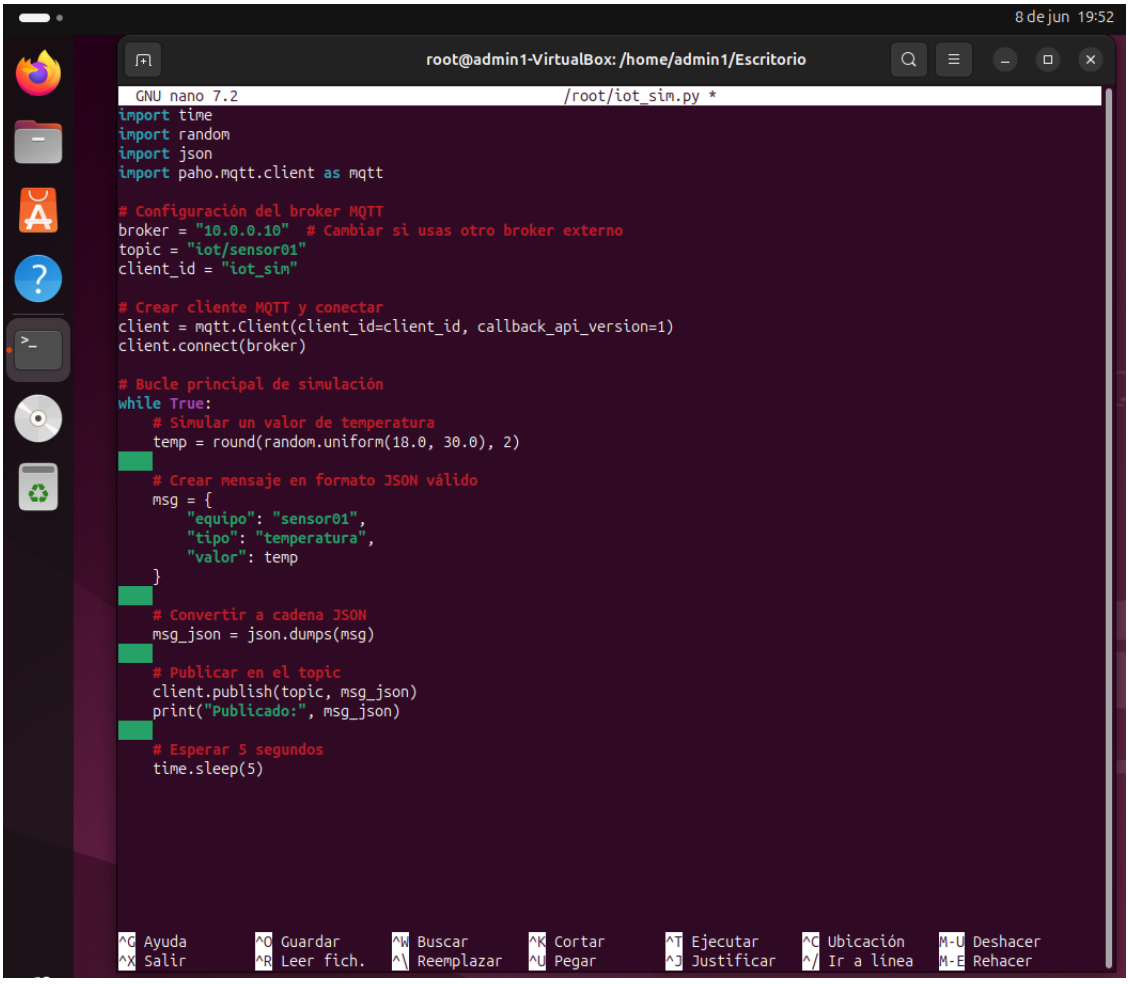
Nota. En la Figura se observa el entorno virtual en ejecución.

3.2.2.4 Generación de Simuladores.

Como parte del desarrollo del presente proyecto, se implementó un simulador en *Python* que emula el comportamiento de sensores IoT. Este simulador genera datos en intervalos de 5 segundos y los publica en un tópico MQTT específico utilizando la librería paho-mqtt. Se diseñaron dos scripts: uno para simular un sensor de temperatura (sensor01) y otro para un sensor de humedad (sensor02), ambos estructurando la información en formato JSON para representar fielmente datos de telemetría.

Figura 22

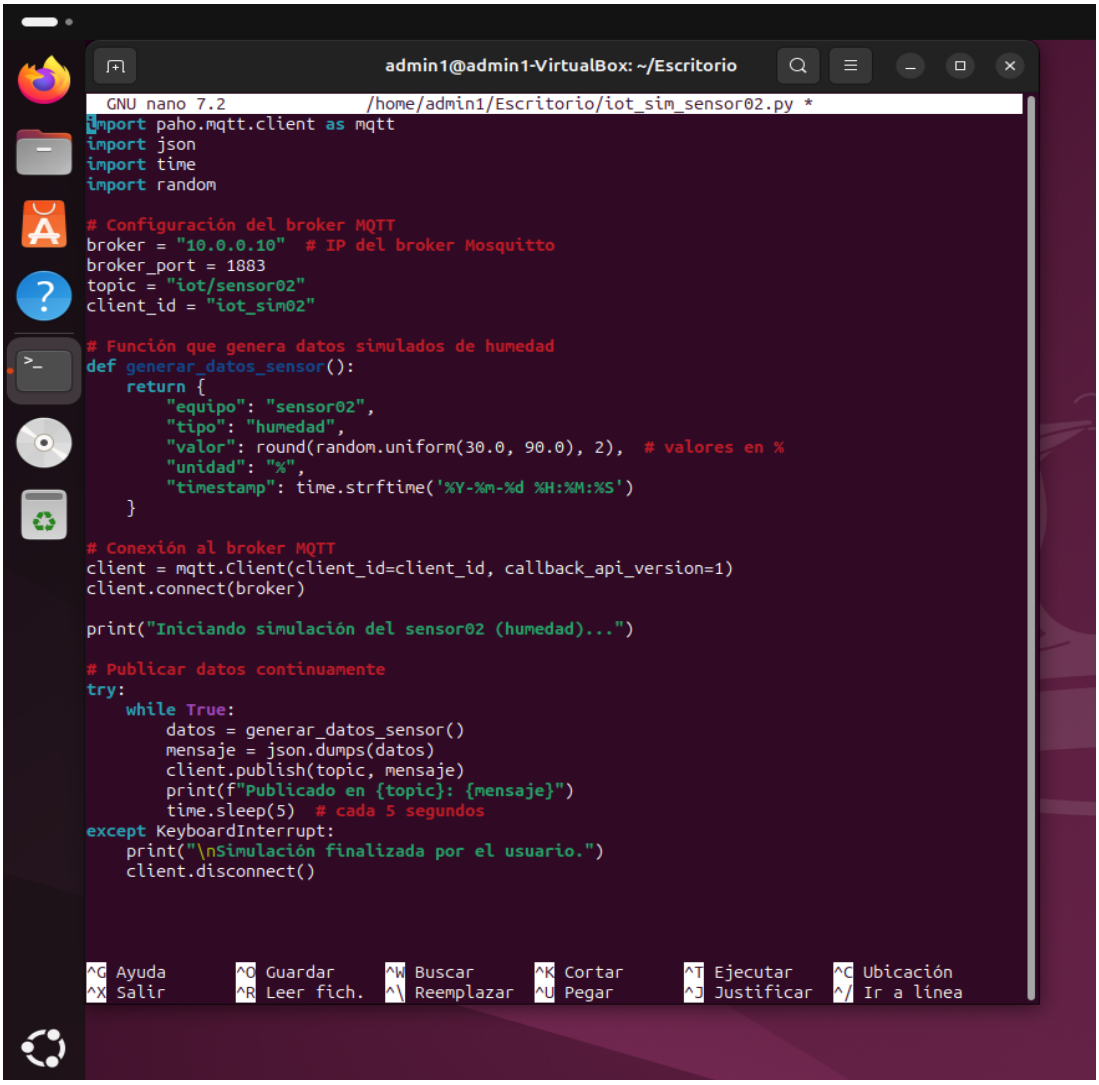
Código generado de sensor 01



Nota. En la Figura se visualiza el código de generación del sensor simulado 1

Figura 23

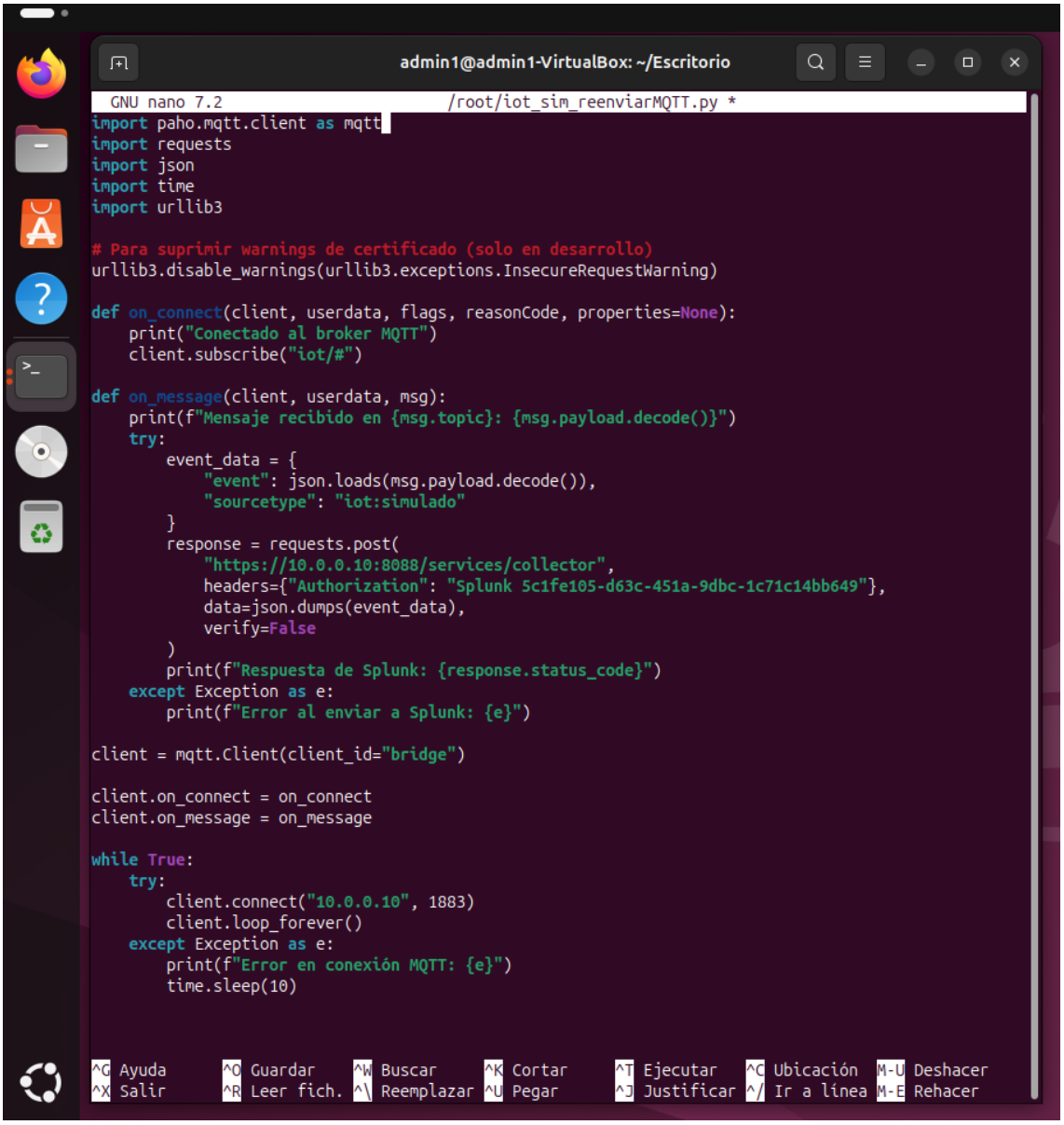
Código generado de sensor 02



Nota. En la Figura se visualiza el código de generación del sensor simulado 2

Figura 24

Código Generado de iot_sim_reenviarMQTT.py



Nota. En la Figura se visualiza el código de generación del conector de reenvió a HEC

El desarrollo de simuladores IoT en *Python* responde a la necesidad de validar la integración entre dispositivos inteligentes y un sistema SIEM sin requerir hardware físico real.

Simulando sensores de temperatura y humedad, se generaron datos representativos en formato JSON que emulan condiciones reales de operación en entornos de *PYMEs*.

3.2.3. Switch Core y Cámara IP

3.2.3.1. En el switch

En base la topología de red, para integrar el Switch Core con *Splunk*, es necesario verificar la compatibilidad y versión del equipo, que para el caso de la simulación se tomó con recurso un switch Cisco de la familia C2600-AdventerpriseK9-M, ideal para enviar logs a *Splunk*, la clave de este equipo es que soporta syslog estándar (RFC 3164 o RFC 5424) y puede configurar la salida de logs, visto en la Figura 25.

Figura 25

Integración de switch core en splunk

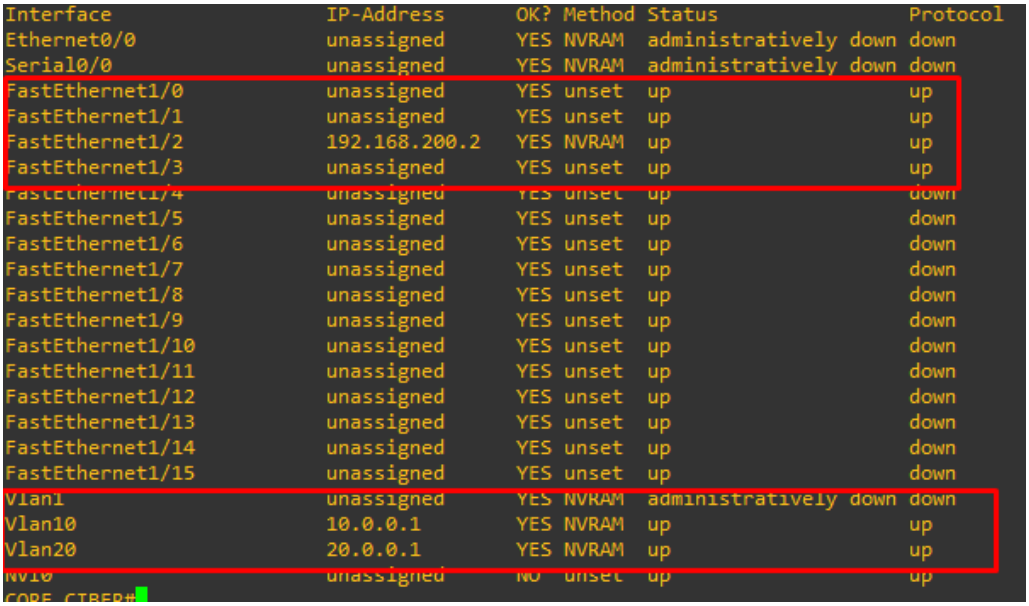
```
CORE_CIBER#show version
Cisco IOS Software, C2600 Software (C2600-ADVENTERPRISEK9-M), Version 12.4(25d), RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2010 by Cisco Systems, Inc.
Compiled Wed 18-Aug-10 04:49 by prod_rel_team

ROM: ROMMON Emulation Microcode
ROM: C2600 Software (C2600-ADVENTERPRISEK9-M), Version 12.4(25d), RELEASE SOFTWARE (fc1)

CORE_CIBER uptime is 2 minutes
System returned to ROM by unknown reload cause - suspect boot_data[BOOT_COUNT] 0x0, BOOT_COUNT 0, BOOTDATA 19
System image file is "tftp://255.255.255.255/unknown"
```

Nota. En la Figura se observa la descripción general del switch core

Dentro de las configuraciones generales del Switch de Core, se tienen habilitadas las interfaces de salía hacia el Internet (F1/0), como las internas F1/1, F1/2 y F1/3, que corresponde a la infraestructura interna tanto de la red LAN de usuarios y red de producción IOT con la VLAN 10 como la red de DMZ y publicación de servicios en la VLAN 20 recordando que este esquema aborda la simulación de una red IOT para el segmento PYME en el Ecuador. Tal y como se describe en la Figura 26.

Figura 26

Configuraciones generales del Switch de Core



Interface	IP-Address	OK?	Method	Status	Protocol
Ethernet0/0	unassigned	YES	NVRAM	administratively down	down
Serial0/0	unassigned	YES	NVRAM	administratively down	down
FastEthernet1/0	unassigned	YES	unset	up	up
FastEthernet1/1	unassigned	YES	unset	up	up
FastEthernet1/2	192.168.200.2	YES	NVRAM	up	up
FastEthernet1/3	unassigned	YES	unset	up	up
FastEthernet1/4	unassigned	YES	unset	up	down
FastEthernet1/5	unassigned	YES	unset	up	down
FastEthernet1/6	unassigned	YES	unset	up	down
FastEthernet1/7	unassigned	YES	unset	up	down
FastEthernet1/8	unassigned	YES	unset	up	down
FastEthernet1/9	unassigned	YES	unset	up	down
FastEthernet1/10	unassigned	YES	unset	up	down
FastEthernet1/11	unassigned	YES	unset	up	down
FastEthernet1/12	unassigned	YES	unset	up	down
FastEthernet1/13	unassigned	YES	unset	up	down
FastEthernet1/14	unassigned	YES	unset	up	down
FastEthernet1/15	unassigned	YES	unset	up	down
Vlan1	unassigned	YES	NVRAM	administratively down	down
Vlan10	10.0.0.1	YES	NVRAM	up	up
Vlan20	20.0.0.1	YES	NVRAM	up	up
Nv10	unassigned	NO	unset	up	up

CORE_CIBER#

Nota. En la Figura se observa las configuraciones red que maneja el switch core

Con las configuraciones ya establecidas en la tabla 6 se visualiza los parámetros que se necesitan para la integración hacia el SIEM, en donde se valida que la comunicación será hacia la ip del servidor 10.0.0.10 utilizado el puerto syslog 514/UDP.

Tabla 6

Integración de SIEM

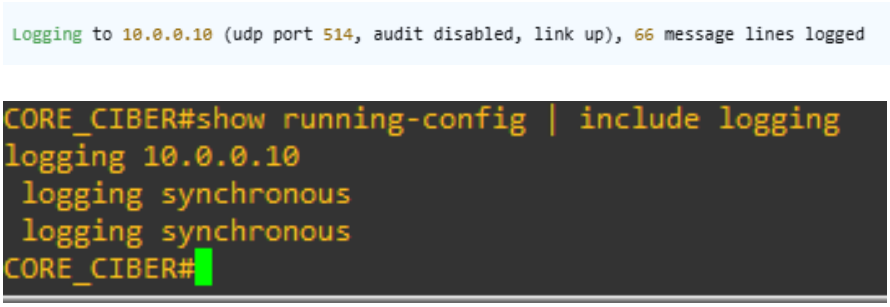
Parámetro	Valor
Dirección IP SIEM	10.0.0.10
Protocolo de comunicación	Syslog
Puerto de comunicación	514/UDP

Nota. La tabla muestra los tipos de protocolos de integración del SIEM

En la figura 27 se tiene la configuración realizada en el switch de Core en base a los parámetros contextualizados en este apartado para la integración con el SIEM. Esto indica que los logs se están enviando al servidor *Splunk* (10.0.0.10) con severidad.

Figura 27

Configuración en el switch de core



Nota. En la Figura se muestra la ejecución correcta del switch core

3.2.3.2. En la cámara IP.

Para integrar un dispositivo IoT, se simula una cámara inteligente a nivel de software utilizando *Node-RED* sobre un entorno Ubuntu. El objetivo es configurar un flujo que permita gestionar actualizaciones de configuración, realizar verificaciones de estado y simular eventos generados por la cámara.

El flujo está diseñado para procesar solicitudes HTTP, detectar contraseñas débiles e intentos de fuerza bruta, y registrar los eventos en formato JSON. Estos registros se preparan para su posterior envío a un servidor *Splunk*, facilitando así el monitoreo y análisis de seguridad.

3.2.3.2.1. Preparación ambiente.

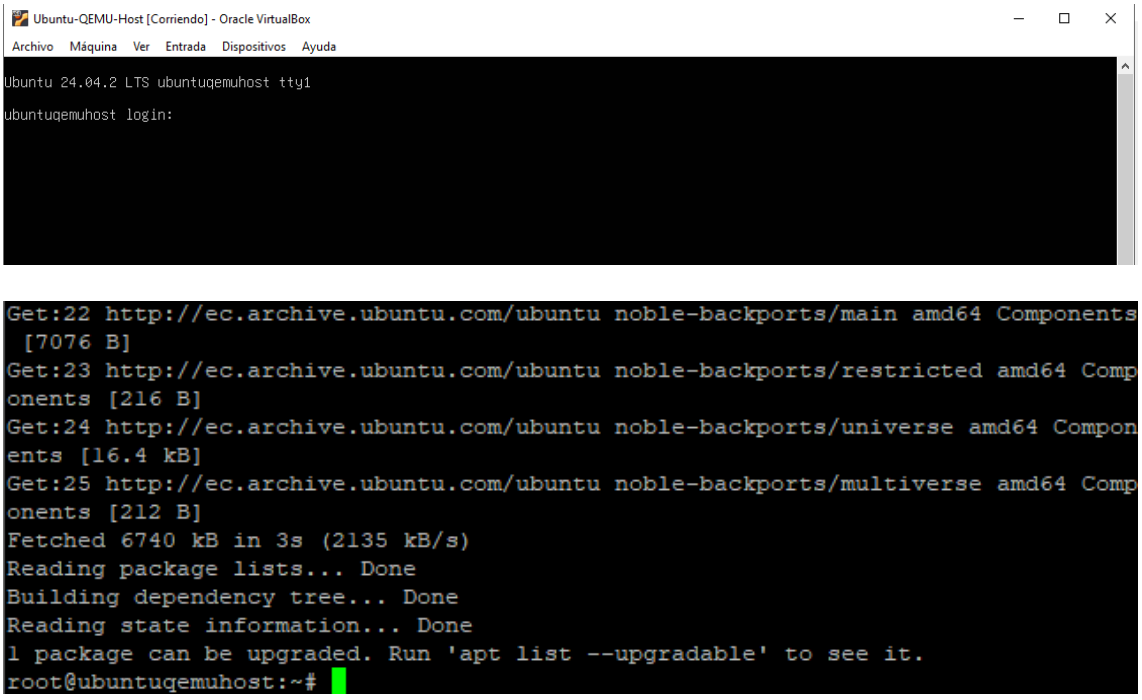
El ambiente se ejecuta sobre un servidor Ubuntu 24.04 LTS, como requisitos mínimos se requiere 2 GB Ram, 15GB en disco y garantizar acceso por terminal por SSH

- Actualización de paquetes

Con permisos de administrador, abrimos una terminal en nuestro servidor Ubuntu y ejecutamos actualización de paquetes

Figura 28

Preparación de ambiente en el servidor Ubuntu 24.04 LTS



Nota. En la Figura se muestra la actualización de paquetes del servidor Ubunto

- **Instalación de Node.js**

Node.js es un entorno de ejecución de JavaScript de código abierto y multiplataforma que permite ejecutar código JavaScript fuera del navegador, ideal para desarrollar aplicaciones escalables y de alto rendimiento, como servidores web, APIs y sistemas en tiempo real, gracias a su modelo asincrónico y basado en eventos.

Sobre la terminal con permisos de administrador ejecutamos los siguientes comandos para la instalación.

Figura 29

Instalación de node.js

```
~$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
~$ sudo apt install -y nodejs
```

Nota. Comandos para la descarga y posterior instalación del módulo de node.js

Verificamos que la instalación se haya ejecutado correctamente:

Figura 30

Verificación de instalación correcta del servidor Ubuntu

```
root@ubuntuqemuhost:~# node -v
v20.19.2
root@ubuntuqemuhost:~# npm -v
10.8.2
```

Nota. En la Figura se muestra la validación de versión instalada y ejecutada

- **Instalación Node-RED**

Node-RED es una herramienta de desarrollo basada en flujo, de código abierto, que permite programar visualmente aplicaciones para el Internet de las Cosas (IoT), automatización y servicios web, mediante una interfaz gráfica en la que se conectan nodos que representan funciones, entradas/salidas y procesos, todo ejecutado sobre Node.js.

Sobre la terminal con permisos de administrador ejecutamos los siguientes comandos para la instalación.

Figura 31

Instalación global de Node-RED mediante NPM

```
~$ sudo npm install -g --unsafe-perm node-red
```

Nota: scripts de instalación que se ejecutan con permisos de super usuario.

Verificamos que la instalación se haya ejecutado correctamente:

Figura 32

Instalación de Node-RED

```
root@ubuntuqemuhost:~# node-red --version
Node-RED v4.0.9
Node.js v20.19.2
Linux 6.8.0-60-generic x64 LE
```

Nota. En la Figura se muestra la versión y ejecución del Node-RED

- **Configuración de usuarios**

Creamos un usuario dedicado para Node-RED bajo un usuario no ROOT por seguridad con el siguiente comando:

Figura 33

Creación del usuario dedicado para Node-RED

```
~$ sudo useradd -m -s /bin/bash node-red
```

Nota. Script de creación del usuario dedicado para Node-RED

Editamos el archivo de configuración de Node-RED con el siguiente comando.

Figura 34

Edición del archivo de configuración de Node-RED

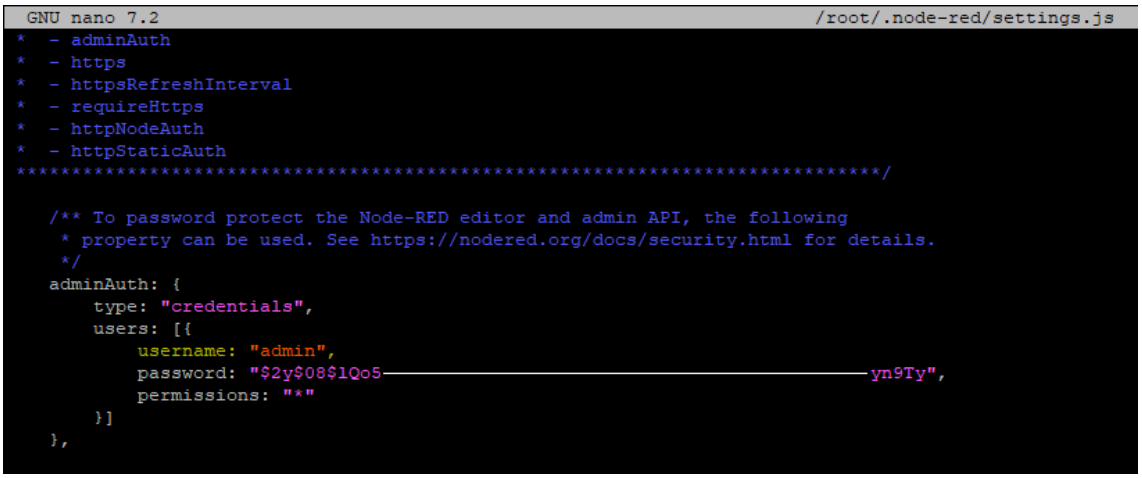
```
~$ nano ~/.node-red/settings.js
```

Nota. Modificar el archivo principal de configuración de Node-RED para configuración de parámetros requeridos para el sistema.

Para establecer el puerto 1880 y asegurar el acceso administrativo HTTP

Figura 35

Configuración de Node-RED



Nota. En la Figura se observa la configuración de credenciales de Node-RED

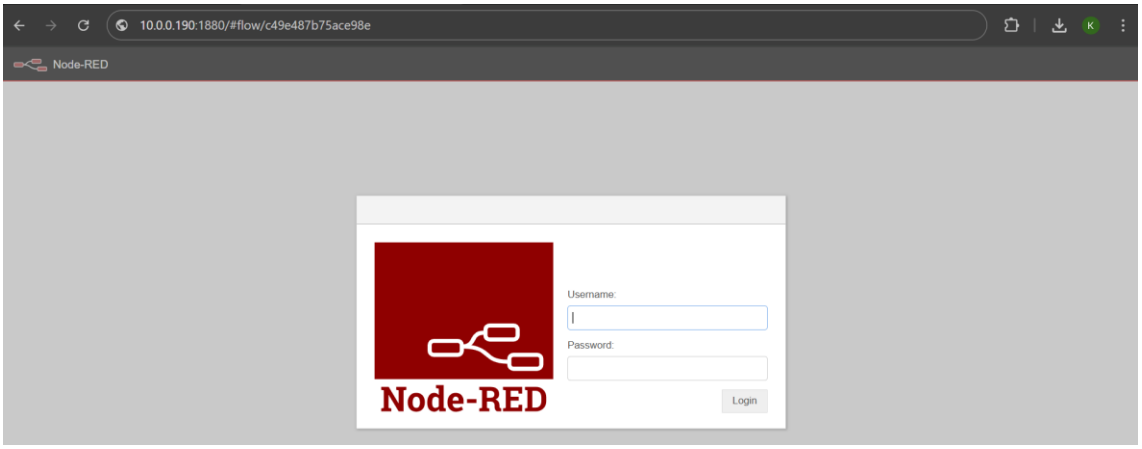
- **Acceso Interfaz GUI**

Sobre la terminal ejecutamos el comando *node-red &*, sobre un navegador ingresamos la dirección IP de nuestro servidor con el puerto especificado en la configuración

http://10.0.0.190:1880

Figura 36

Acceso a la interfaz GUI

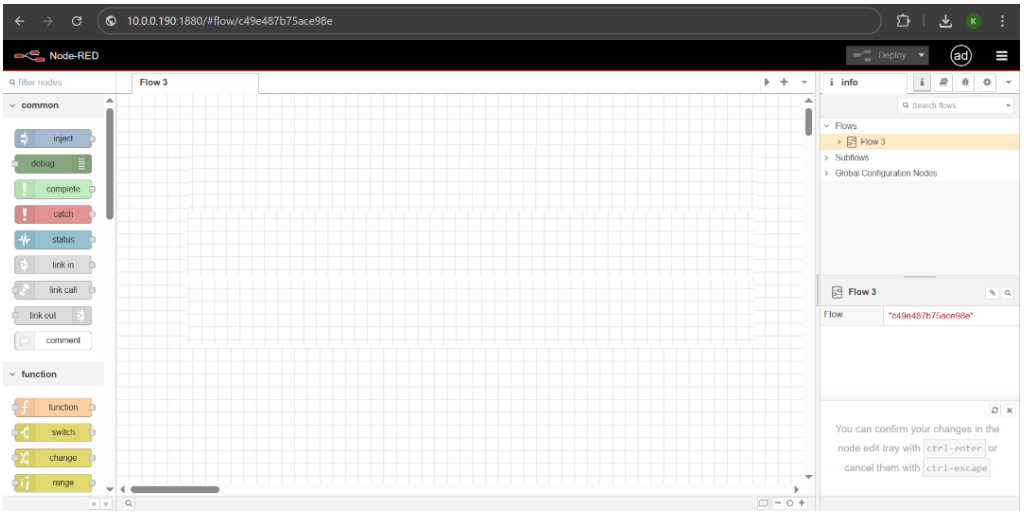


Nota. En la Figura se muestra la página login de Node-RED para el ingreso

Se ingresa con las credenciales definidas sobre el archivo de configuración.

Figura 37

Interfaz gráfica App Red Node

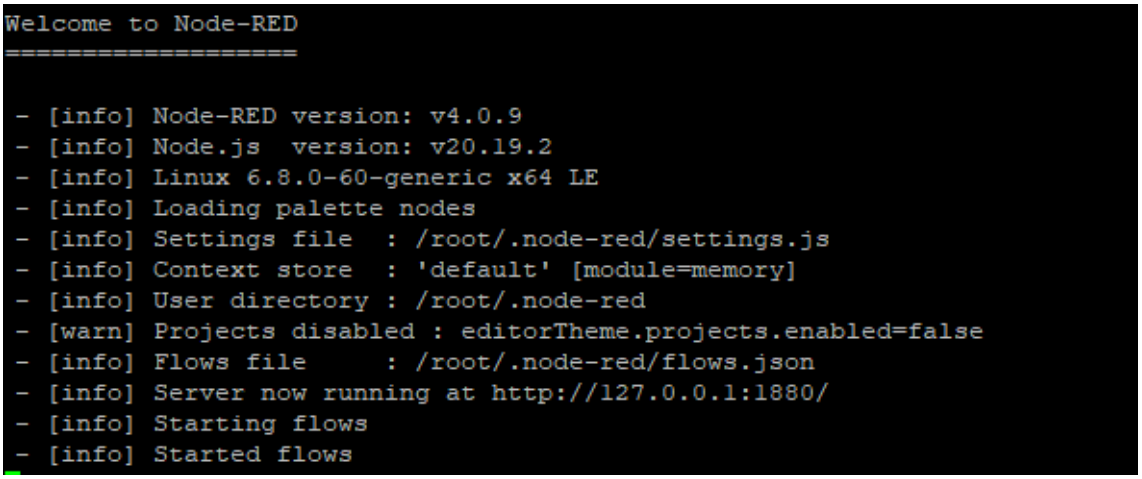


Nota. Se visualiza comportamiento óptimo de la interfaz GUI para inicio y carga de nodos

Para validar status del servicio, verificamos en consola.

Figura 38

Verificación del servicio en consola



Nota. En la Figura se observa la validación de servicio que se ejecutan de Node-RED

- **Archivo de configuración de logs**

  En nuestro terminal navegamos al directorio /var/log/ para ratificar que exista el archivo iot.log el cual se alimentará de los eventos recopilados.

Figura 39

Configuración de logs

Nota. En la Figura se observa la ruta de LOG del sistema de Node-RED

2.3.1.1 Generación de flujos

Vamos a generar un archivo. JSON el cual procese los siguientes flujos:

Tabla 7

Generación de flujos

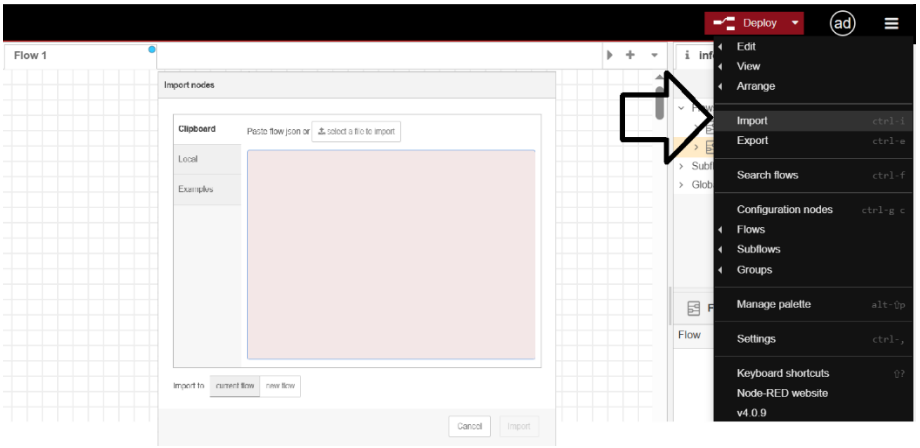
Componente/Flujo	Descripción
Flujo de configuración	Procesa actualizaciones de configuración, detecta contraseñas débiles (≤7 caracteres) e intentos de fuerza bruta (>10 solicitudes).
Flujo de estado	Devuelve el estado del dispositivo con la última marca de tiempo de movimiento.
Eventos simulados	Genera motion_detected, lens_obstructed y outdated_firmware cada 5 segundos.
Registro	Escribe eventos JSON en /var/log/iot.log.

Nota. Tabla que permite indicar el flujo completo que sigue la simulación del ataque.

Sobre la interfaz GUI importamos el código JSON (Anexo A) haciendo clic en la barra lateral, opción Importar.

Figura 40

Importación del JSON

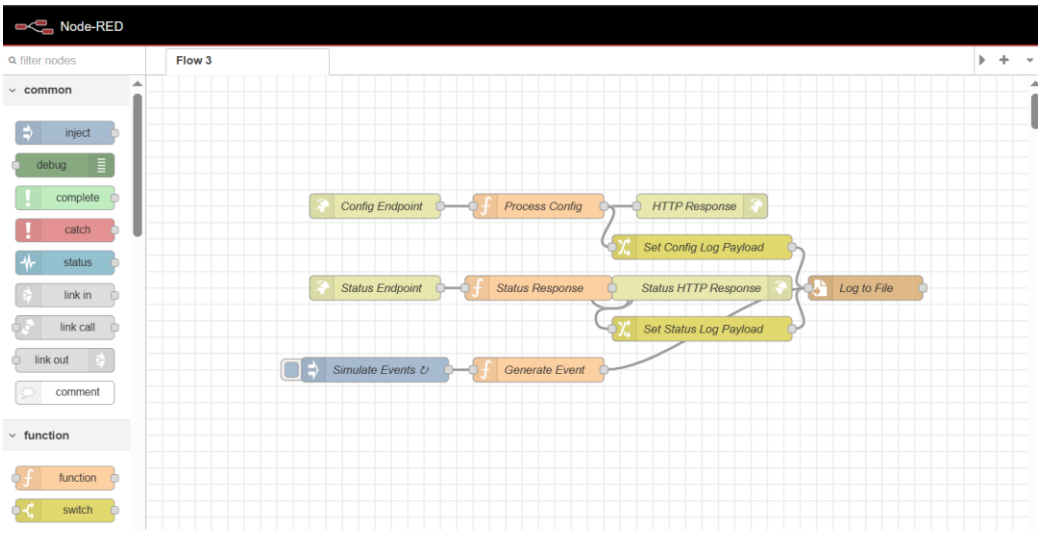


Nota. En la Figura se observa la acción de importar el archivo de LOG Json

Una vez ejecutado la importación, en la pantalla principal observamos los diferentes flujos, hacemos clic en Deploy para iniciar la operación.

Figura 41

Visualización de nodos en App Red Node



Nota. En la Figura se visualiza de manera gráfica los nodos importados y sus interacciones para su operación

3.2.3.2.2. Revisión de logs.

Sobre la terminal ejecutamos el siguiente comando.

Figura 42

Monitoreo en tiempo real del archivo de log de IoT

```
~$  tail -f /var/log/iot.log
```

Nota. Comando que permite observar en tiempo real los registros generados por los scripts simuladores de dispositivos IoT

Con el objetivo de revisar que las transmisiones se estén realizando correctamente

Figura 43

Revisión de transmisiones

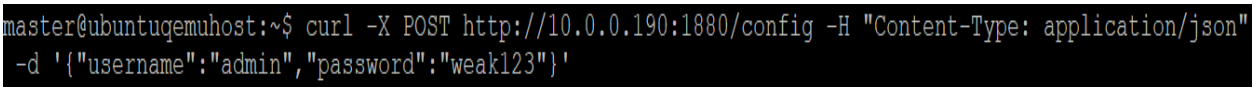
Nota. En la Figura se observa los mensajes de transmisión en consola

- **Test flujo de configuración**

  Ejecutamos la siguiente sintaxis, la cual simula un intento de configuración con contraseña débil sobre el dispositivo IoT.

Figura 44

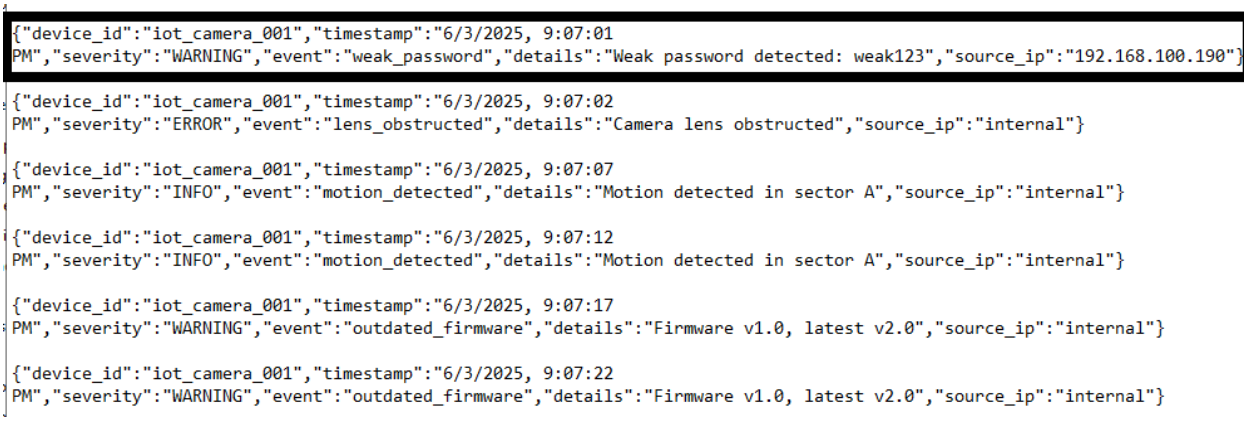
Configuración remota en Node-RED mediante solicitud POST



Nota: La figura muestra el uso del comando curl para enviar credenciales de configuración al servidor Node-RED a través del puerto 1880, simulando una interacción desde un dispositivo IoT.

Figura 45

Flujo de configuración de simulación



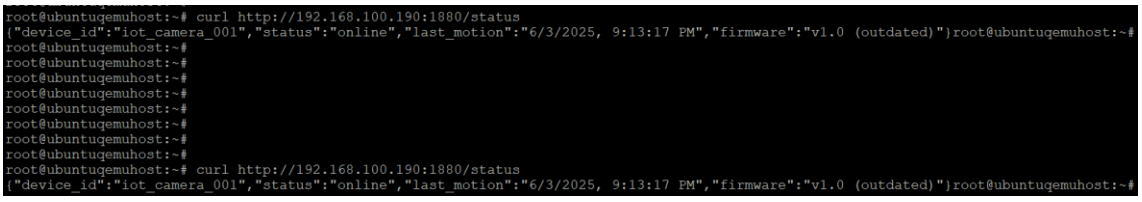
Nota. En la Figura se observa los logs de simulación de contraseña débil

- **Test flujo de estado**

  Ejecutamos la siguiente sintaxis (*curl http://10.0.0.10:1880/status*), la cual simula el estado del dispositivo con la última marca de tiempo de movimiento del dispositivo IoT

Figura 46

Test de flujo de estado



Nota. En la figura se observa la consulta del estado del dispositivo IoT con la última marca de movimiento.

- **Test detección de fuerza bruta**

  Ejecutamos la siguiente sintaxis, la cual simula intentos de configuración de tipo fuerza bruta:

Figura 47

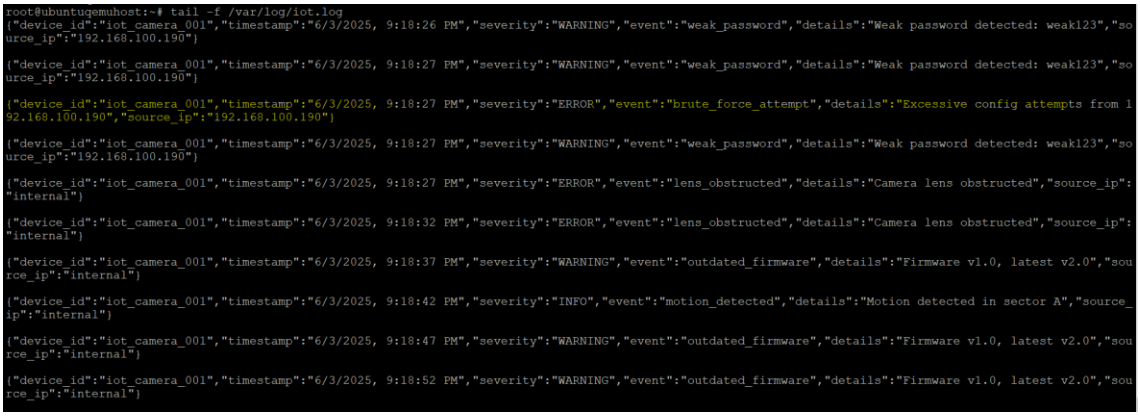
Ataque de fuerza bruta simulado a interfaz Node-RED

```
master@ubuntuqemuhost:~$      for i in {1..11}; do curl -X POST http://10.0.0.190:1880/config -d '{"use
rname":"admin","password":"weak123"}'; done
```

Nota. El script ejecuta múltiples solicitudes POST con credenciales débiles hacia el endpoint /config de Node-RED, representando un intento automatizado de acceso no autorizado, común en escenarios de evaluación de seguridad IoT.

Figura 48

Test de detección de fuerza bruta



Nota. En la figura se observa la simulación de intentos de fuerza bruta sobre el dispositivo IoT.

Con las configuraciones definidas en la Tabla 7, se identifican los parámetros necesarios para la integración con el sistema SIEM. En esta etapa, se valida que la comunicación se realizará hacia la dirección IP del servidor 10.0.0.10, utilizando el puerto 514/UDP correspondiente al protocolo Syslog.

Tabla 8

Parámetros y protocolos de uso

Parámetro	Valor

Dirección IP SIEM	10.0.0.10
Dirección IP Host IoT Cámara	10.0.0.190
Protocolo de comunicación	Syslog
Puerto de comunicación	514/UDP

Nota. En la tabla se visualiza los parámetros de comunicación entre el dispositivo IoT y el SIEM.

3.3 Integración con *Splunk*

3.3.1. Entradas configuradas

Tabla 9

Protocolos de comunicación

Tipo	Método	Fuente
MQTT	Python → HEC	Sensor01-Temperatura - Sensor02 - Humedad
SNMP	snmptrapd → archivo	Servidor Linux, Cámara
Syslog	UDP 514	Switch
Syslog	TCP 514	Cámara IoT

Nota. En la tabla se describen los protocolos y métodos utilizados para la recolección de eventos desde los distintos dispositivos.

3.3.2. Integración Switch de Core al splunk

Continuando con la integración del Switch de Core a la plataforma *Splunk* para la topología de red descrita en este proyecto se ha configurado al switch Cisco para enviar mensajes de registro (*syslog*) al servidor *Splunk* con en la IP 10.0.0.10, utilizando el puerto 514/UDP. Esta conexión permite capturar eventos o cambios de estado en interfaces, modificaciones de configuración y otros eventos operativos, accesos al dispositivo mejorando la visibilidad, el

monitoreo y la capacidad de respuesta ante incidentes de red tiempo real en un contexto simulado. Estos parámetros de configuración esta descritos en la Tabla 10.

Tabla 10

Parámetros de configuración para la integración

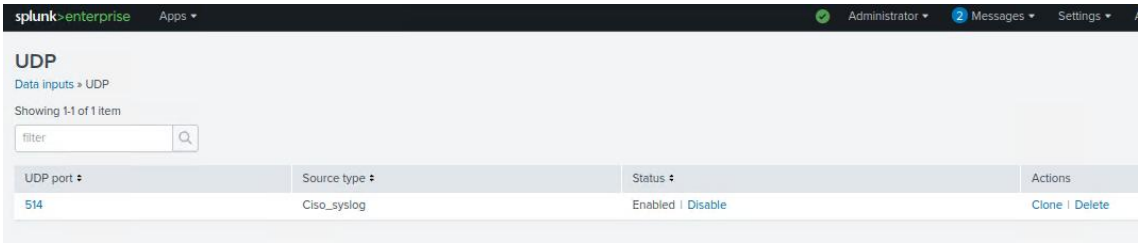
Parámetro	Valor
Configuración	Acceso Web
Puerto	514/UDP
Sourcetype	Ciso_syslog
Index	network_logs

Nota. En la tabla se muestran los parámetros utilizados para la recolección de logs del dispositivo Ciso en el SIEM.

Dicho esto, se realiza la configuración vía Web de los parámetros esenciales para la integración Switch Core con el SIEM, por lo cual es importante recalcar que con esta simulación lo que se pretende es establecer una base de procesos para la integración al *Splunk* de todo componente de *Networing* utilizando *syslog* como protocolo de comunicación. Visto en la Figura 49.

Figura 49

Panel de configuración UDP

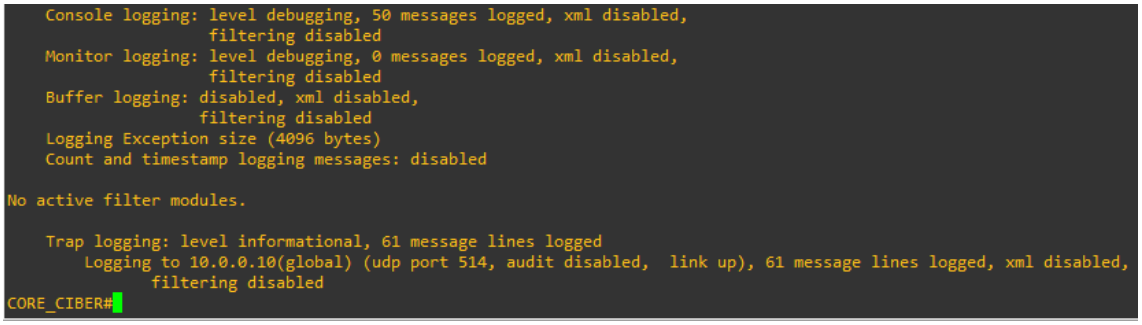


Nota. En la figura se observa la configuración vía web para la integración del Switch Core al SIEM mediante el protocolo Syslog.

Luego de realizar las configuraciones tanto en el SIEM como en el componente de red (Switch Core) se puede validar visto en la Figura 43 el envío correcto de los logs hacia el *Splunk* dando como respuesta positiva a la integración del lado del origen de proceso. El enlace con servidor está activo. Adicional 66 message lines logged (se han enviado 66 líneas de log a ese servidor).

Figura 50

Validación de envío de logs desde el Switch Core hacia el SIEM



Nota. En la figura se confirma el envío exitoso de logs al servidor SIEM desde el Switch Core, registrando 66 líneas de log.

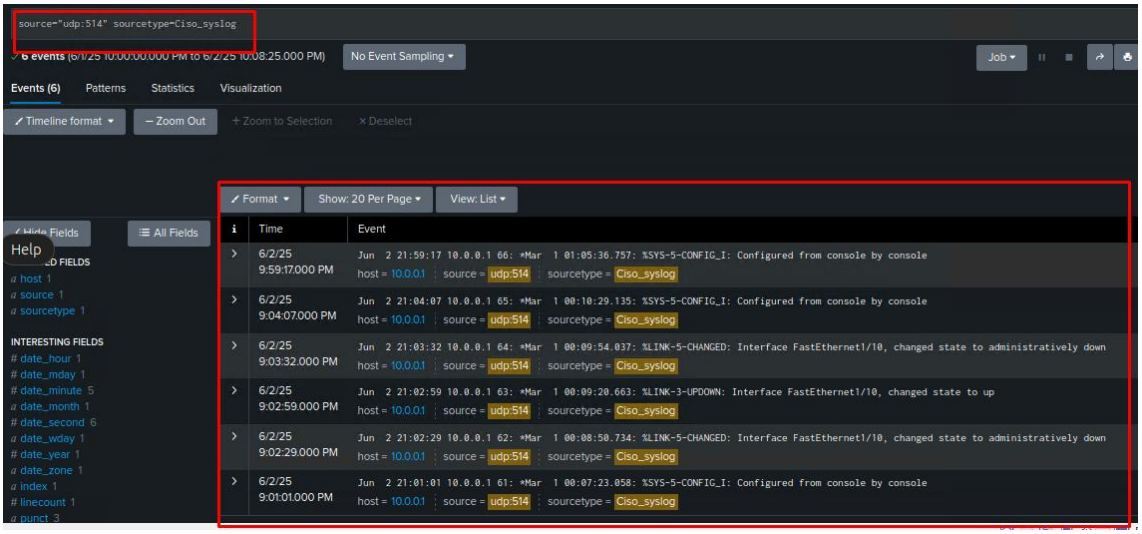
Como indicadores se tiene:

- El syslog remoto está activo.
- Que los mensajes han sido enviados.
- La severidad configurada
- Últimos logs generados exitosamente

En ese sentido como indica la Figura 51, se valida que el equipo de red está enviando logs al servidor *Splunk* en la IP 10.0.0.10, usando UDP puerto 514 y el SIEM está recibiéndolos y agrupando para su interpretación acorde a los lineamientos de análisis, visualización, correlación y acciones a tomar.

Figura 51

Recepción y agrupación de logs del Switch Core en el servidor SIEM



Nota. En la figura se valida la recepción de logs por parte del SIEM desde el Switch Core mediante UDP puerto 514, listos para su análisis y correlación.

3.3.3. Integración del protocolo MQTT al Splunk

Como parte de la integración de fuentes de datos heterogéneas en la plataforma *Splunk*, se implementó un entorno de simulación IoT que permite el envío de eventos generados por sensores virtuales directamente al servidor *Splunk* (IP 10.0.0.10) mediante el HTTP *Event Collector* (HEC).

Esta comunicación se habilita a través de un script puente desarrollado en Python, el cual se suscribe al tópico MQTT /iot/sensor01 y sensor02, donde el simulador iot_sim.py y iot_sim.py_sensor02 publican periódicamente datos representativos, como lecturas de temperatura y humedad. El broker Mosquitto, instalado en el servidor, actúa como intermediario de mensajes entre el simulador y el conector.

Una vez recibidos, los mensajes son estructurados en formato JSON y enviados al HEC de Splunk utilizando un token de autenticación previamente generado. Esta configuración permite

que los eventos IoT sean procesados, indexados y visualizados en tiempo real, facilitando tareas de monitoreo, correlación y análisis dentro de la interfaz de *Splunk*.

Los parámetros esenciales para esta integración se muestran en la siguiente tabla:

Tabla 11

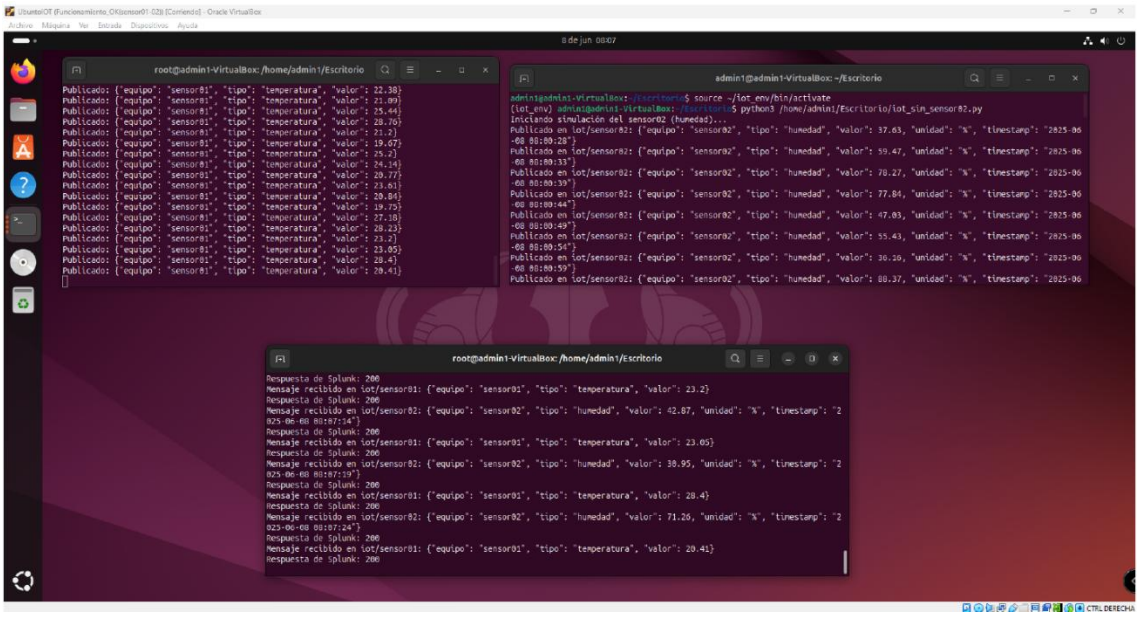
Parámetros de integración mediante script Python

Parámetro	Valor
Protocolo	HTTP (HEC)
Puerto	8088/TCP
Sourcetype	iot:simulado
Índice (Index)	main
Tópico MQTT	/iot/sensor01 - sensor02
Broker	Mosquitto

Nota. En la tabla se muestran los parámetros para enviar datos IoT desde MQTT al HEC de Splunk vía script Python.

Figura 52

Visual global de los scripts en funcionamiento

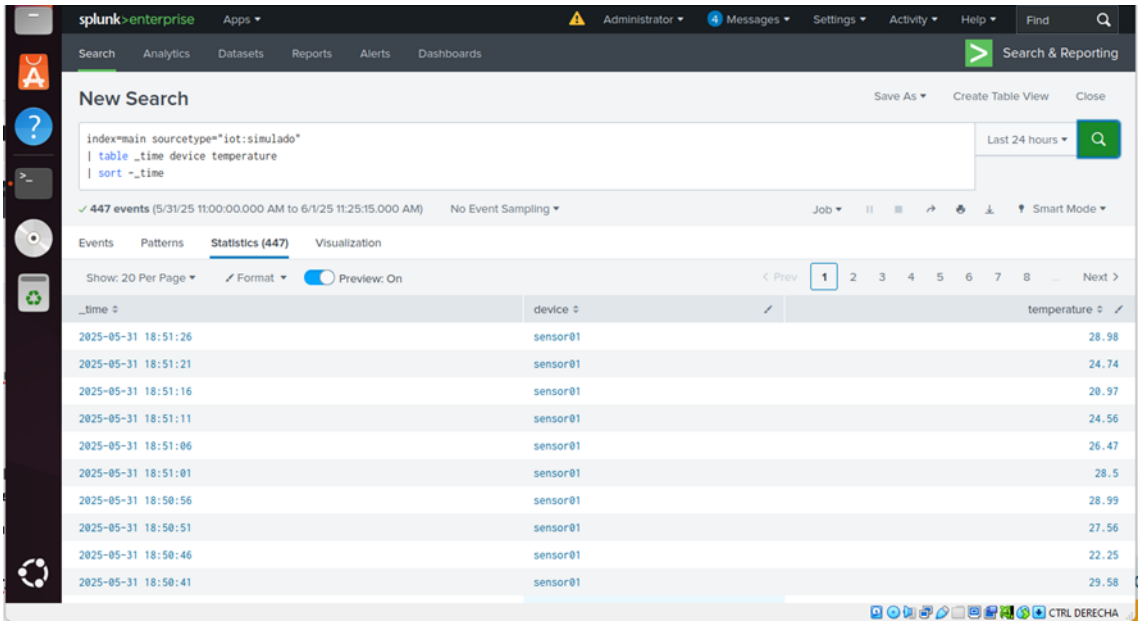


Nota. En la Figura se visualiza los 3 componentes funcionando al mismo tiempo

Los mensajes generados son transmitidos al broker MQTT Mosquitto, instalado en el servidor principal. Desde allí, un tercer script en el cliente actúa como puente de integración, capturando los mensajes publicados y reenviándolos de forma segura al sistema Splunk mediante su HTTP colector de Eventos de esta manera se obtiene una visualización individual de los mensajes con datos simulados desde los simuladores IoT y se observan de la siguiente manera en el caso del sensor de temperatura.

Figura 53

Visualización individual de datos de Temperatura



Nota. En la Figura se visualiza los datos recibidos del simulador de temperatura

3.3.4. Integración de eventos Cámara IoT al Splunk

Continuando con la integración de la cámara IoT a la plataforma *Splunk*, y en el contexto de la topología de red definida en este proyecto, se configura el dispositivo para redirigir sus

mensajes de registro locales mediante el protocolo *Syslog*. Esta configuración se implementa siguiendo los parámetros especificados en la Tabla 12.

Tabla 12

Parámetros de configuración para logs de cámara IoT

Parámetro	Valor
Configuración	Vía Consola
Puerto	514/UDP
Sourcetype	syslog
Index	Camera_iot

Nota. Parámetros utilizados para la recolección de logs de la cámara IoT en el SIEM

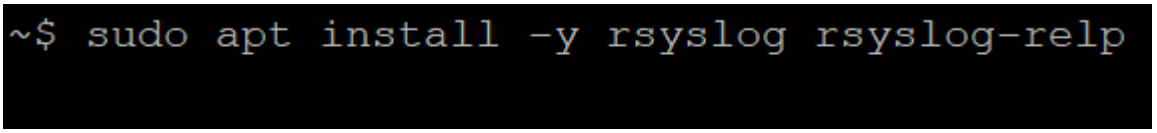
La configuración utiliza el módulo `imfile` para monitoreo de archivos y el protocolo TCP para enviar logs, asegurando compatibilidad con la configuración de Splunk

Instalación SysLog.

Sobre nuestra terminal, ejecutamos el siguiente comando para instalar syslog.

Figura 54

Instalación de Rsyslog con soporte RELP

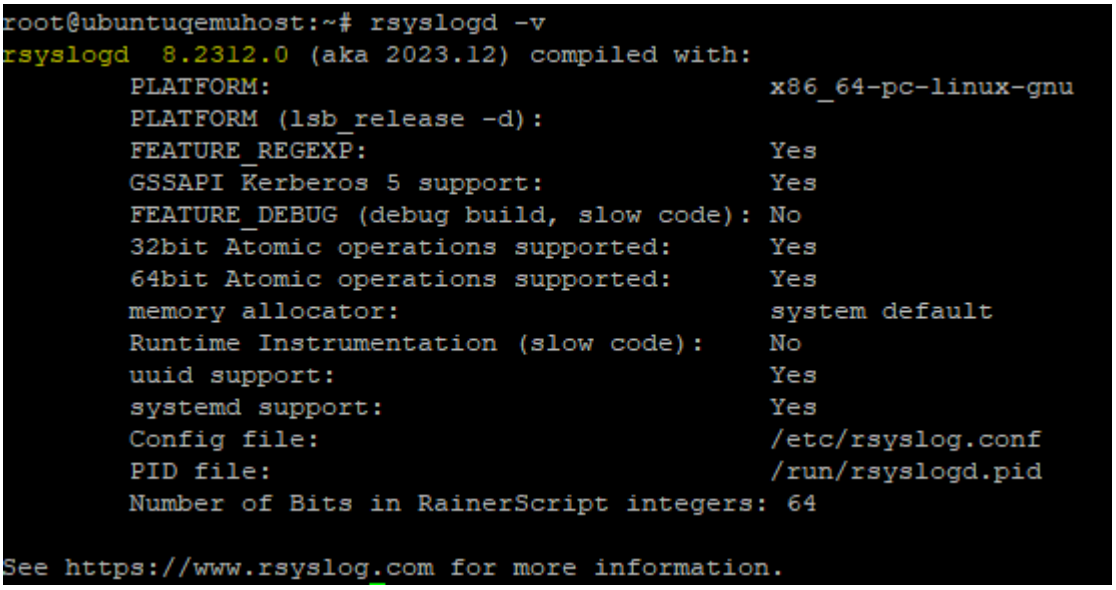


Nota: Se instala el servicio Rsyslog junto con el módulo rsyslog-relp, permitiendo el envío confiable de logs mediante el protocolo RELP (Reliable Event Logging Protocol).

Verificamos la versión instalada con el comando rsyslogd -v

Figura 55

Instalación y verificación de Syslog en el sistema



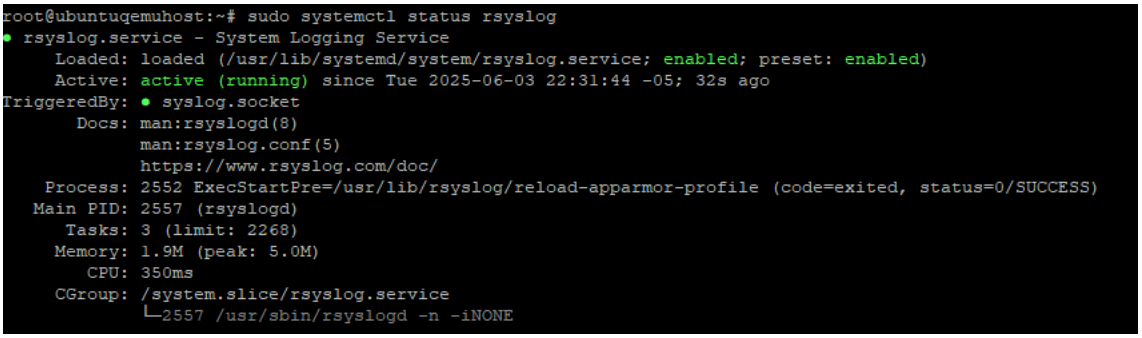
Nota. En la figura se muestra la ejecución del comando para instalar Syslog y la verificación de la versión instalada en la terminal.

- Servicio SysLog

  Validamos que el servicio syslog se encuentre levantado para continuar con la configuración

Figura 56

Validación del estado del servicio Syslog



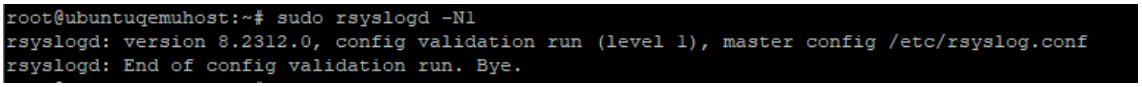
Nota. En la figura se observa la verificación del estado activo del servicio Syslog

- Configuración Syslog

  Abrimos el archivo de configuración principal (sudo nano /etc/rsyslog.conf) y parametrizamos para enviar logs al destino 10.0.0.10, validamos que la configuración se haya ejecutado correctamente.

Figura 57

Configuración del archivo rsyslog.conf para envío de logs



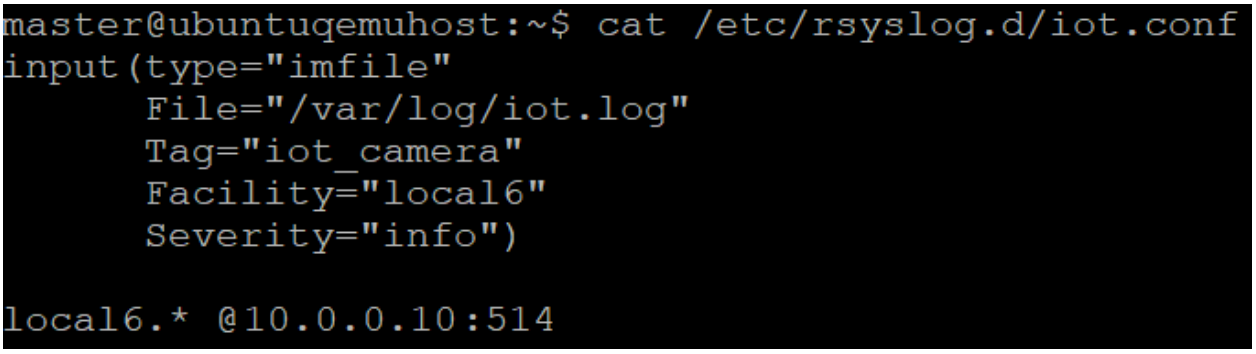
Nota. En la figura la validación de configuración del archivo rsyslogd

- Configuración Logs Cámara IoT

  Creamos un archivo de configuración específico con los parámetros del archivo de log generado por la cámara IoT y la dirección destino del servidor Splunk

Figura 58

Configuración de Rsyslog para el monitoreo del archivo de log del IoT



Nota: El archivo iot.conf permite capturar eventos desde /var/log/iot.log con la etiqueta iot_camera, clasificarlos como local6.info y reenviarlos al servidor remoto con IP 10.0.0.10 en el puerto estándar 514 mediante el protocolo UDP, asegurando la integración de eventos IoT con plataformas de monitoreo

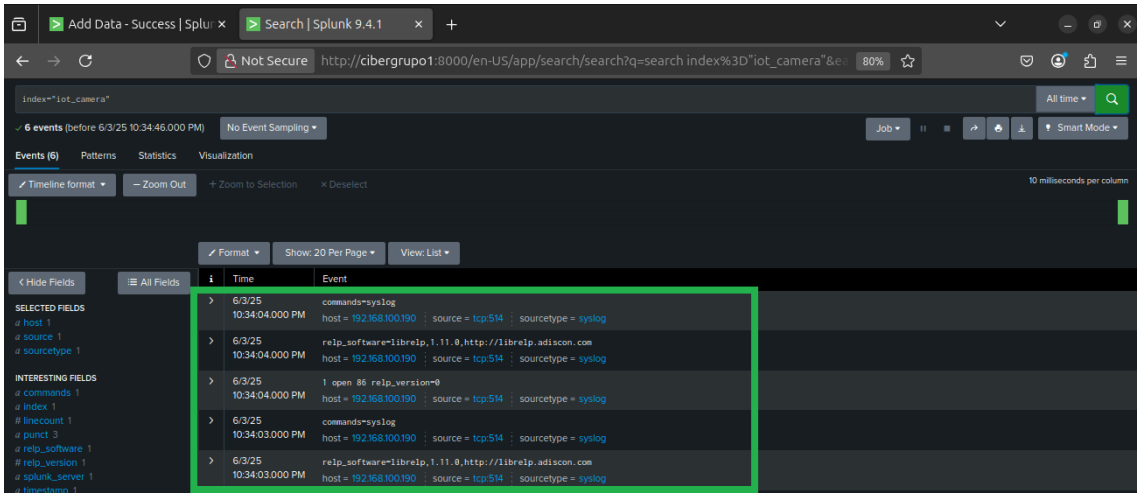
3.3.5. Integración a Splunk

Accedemos al servidor *Splunk* y, con base en los parámetros definidos en la tabla E, procedemos a crear un índice y un data input para habilitar la recepción de logs provenientes de la cámara IoT.

A continuación, en la barra de búsqueda de *Splunk*, ejecutamos una consulta global utilizando el índice creado (camera_iot) para verificar que los registros, anteriormente visibles solo de forma local, están siendo correctamente retransmitidos a la plataforma. Estos logs ahora se visualizan en la interfaz de búsqueda de Splunk, listos para su análisis, correlación y tratamiento dentro del entorno de monitoreo (Figura 59).

Figura 59

Verificación de logs de cámara IoT en la interfaz de búsqueda de Splunk



Nota. En la figura se confirma que los registros del índice camera_iot son recibidos y visualizados correctamente en Splunk.

3.3.5. Sourcetypes definidos

- Iot:simulacion para datos MQTT
- snmp_log para traps
- switch_syslog

- syslog para cámara IoT

CAPITULO 4

ANÁLISIS DE RESULTADOS

4.1. Pruebas de Concepto

4.1.1 Dashboards y Visualización

4.1.1.1 Paneles desarrollados:

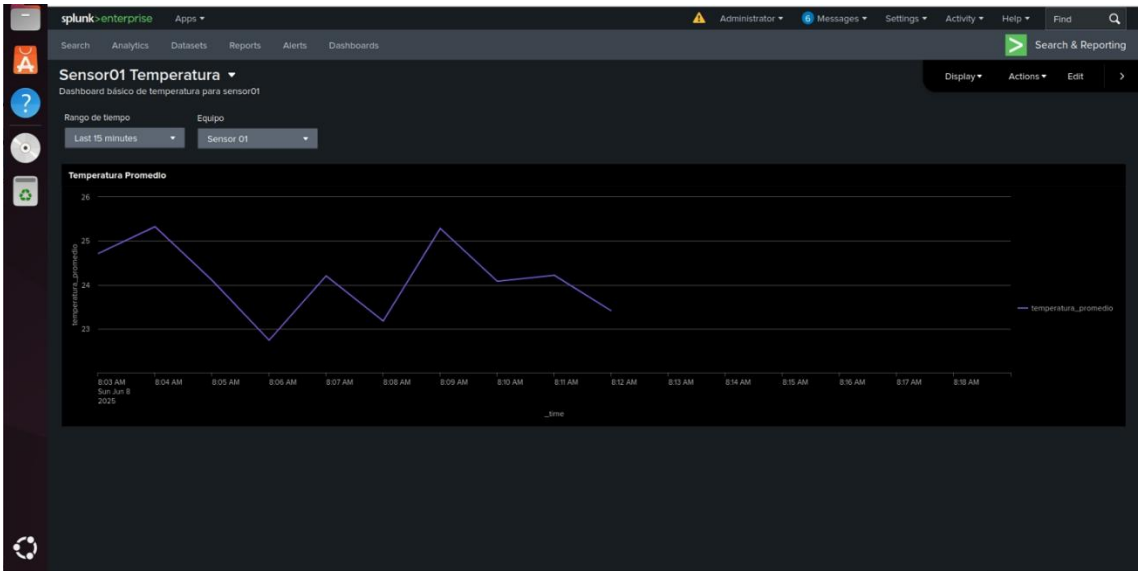
- **Estado de los sensores de temperatura y Humedad en tiempo real**

Como parte de la validación de la solución propuesta, se desarrollaron visualizaciones gráficas en *Splunk* que representan el flujo y la frecuencia de eventos generados por los sensores simulados. Estas gráficas, de tipo lineal, fueron construidas a partir de los datos enviados por los scripts iot_sim.py y iot_sim.py_sensor02, que simulan sensores de temperatura y humedad respectivamente, y publicados en tiempo real a través del protocolo MQTT.

El script iot_sim_reenviarMQTT.py interceptó estos mensajes y los reenvió al colector HTTP de *Splunk* (HEC), donde fueron indexados bajo un *sourcetype* personalizado (iot:simulado). A partir de estos datos, se generaron *dashboards* que muestran de forma clara la distribución temporal y temática de los eventos según su origen (sensor01 o sensor02), facilitando así el monitoreo del comportamiento del entorno IoT simulado.

Figura 60

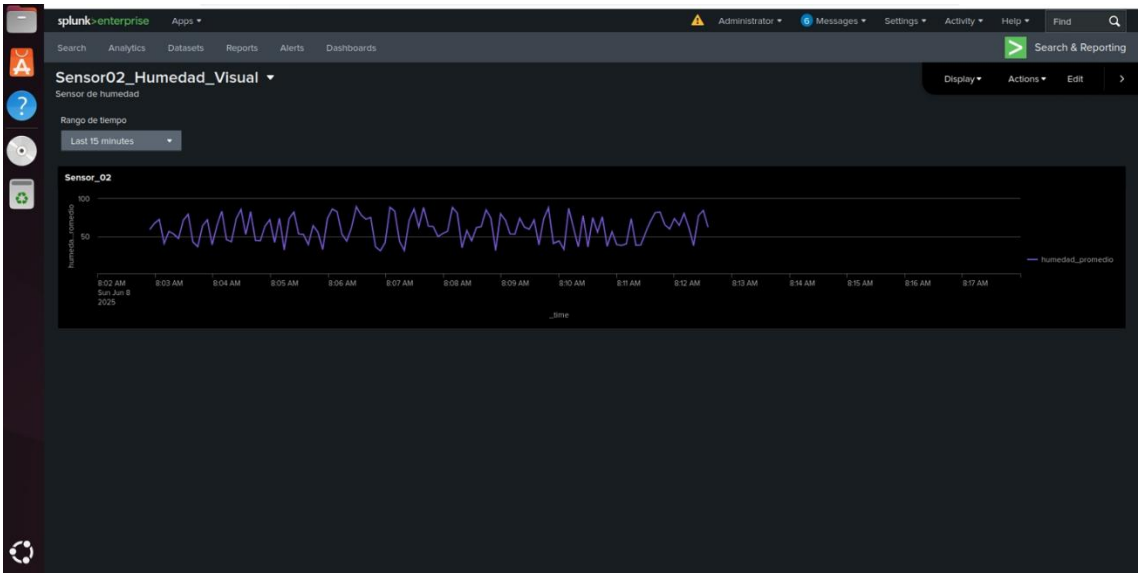
Visualización de eventos en tiempo real del sensor 01



Nota. En la Figura se muestra la gráfica de eventos simulados del sensor de temperatura en Splunk.

Figura 61

Visualización de eventos en tiempo real del sensor 02



Nota. En la Figura se muestra la gráfica de eventos simulados del sensor de humedad en Splunk.

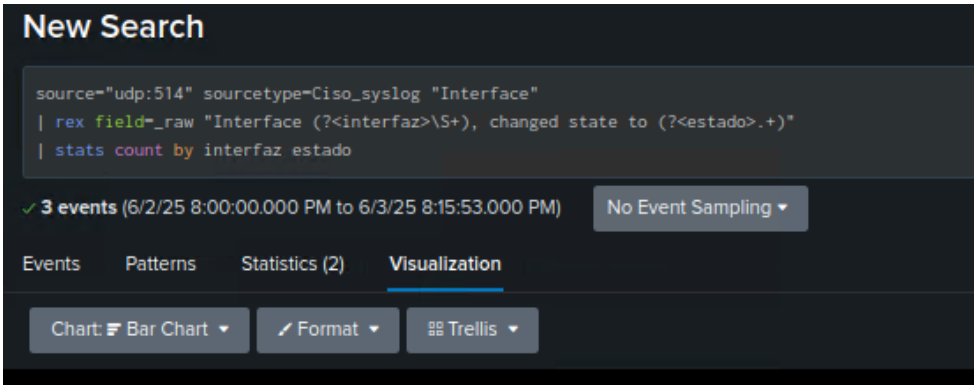
- **Estado de interfaces del switch**

Ya con la integración también del componente de red en la plataforma *Splunk* que recolecta los logs del switch para poder monitorearlo desde un panel (Dashboard/gráficas) se convierten en herramientas necesaria para observar el estado de las interfaces, así como la administración y recursos de manera intuitiva y en tiempo real.

Para lo cual se establece un panel personalizado para visualizar fácilmente eventos de cambios de estado, interfaces caídas o activas, facilitando la identificación rápida de problemas y el análisis continuo del rendimiento de la red. Como se identifica en la figura S, se tiene creado un filtro que permite agrupar los logs recibidos y los interpreta en una visualización centralizada base al contexto (“interfaz” y “estado”), con ello lo que se trata es mejora la eficiencia operativa para establecer alertas y acciones dependiendo el estado.

Figura 62

Filtro de eventos del switch en Splunk



Nota. Creación del filtro de búsqueda en Splunk para el monitoreo de interfaces y estados del switch, facilitando alertas y análisis en tiempo real.

La Figura 63 muestra el estado actual de cada interfaz del switch, distinguiendo claramente si hay interfaces activas (“up”) y las que presentan problemas o están deshabilitadas (“down”). Esto permite identificar rápidamente un incidente o problemas de conectividad

requieren atención inmediata, facilitando el monitoreo constante del rendimiento de la red esencial para una arquitectura PYME que en varias ocasiones no disponen de un SOC, adicional promueve la detección temprana de fallas que puedan afectar la conectividad o el funcionamiento de los dispositivos conectados.

Figura 63

Estado actual de interfaces del switch en Splunk

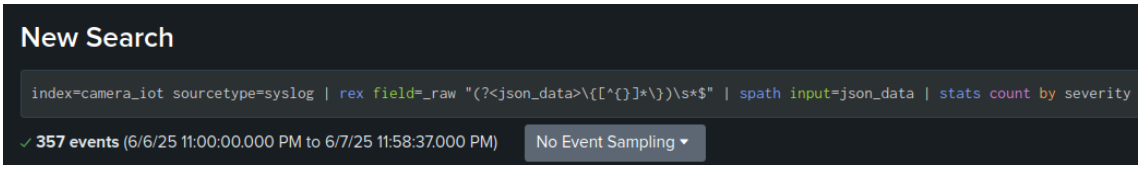
Nota. En la Figura se visualiza el estado “up” o “down” de las interfaces del switch, facilitando la detección rápida de fallas de conectividad en entornos PYME.

- **Eventos por criticidad**

  En base a los logs registrados por la cámara IoT, podemos generar un query para contabilizar los eventos por nivel de gravedad para identificar posibles incidentes de seguridad.

Figura 64

Filtro de eventos por criticidad en cámara IoT

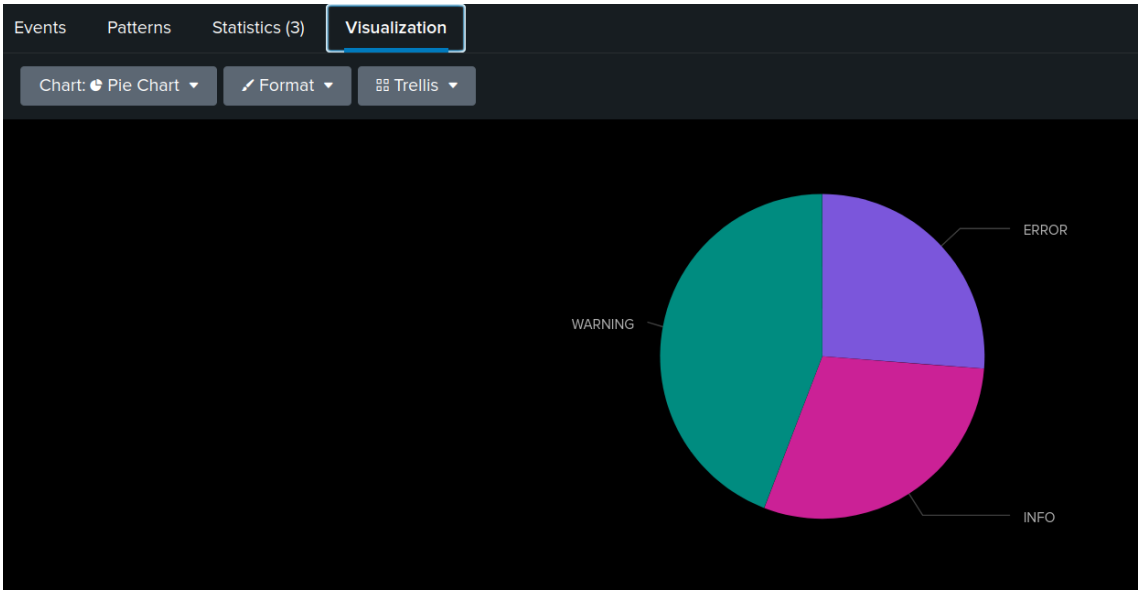


Nota. En la Figura se muestra la consulta en Splunk que clasifica los eventos según su nivel de gravedad, facilitando la detección de posibles incidentes.

Como se muestra en la Figura 65, se parametriza la búsqueda para contabilizar y agrupar los registros por nivel de severidad, utilizando la función *stats count by severity* sobre el índice correspondiente. Esta configuración permite obtener una visión consolidada de la cantidad de eventos clasificados según su criticidad, facilitando el análisis y priorización de incidentes dentro del entorno *Splunk*.

Figura 65

Conteo de eventos por severidad en Splunk



Nota. En la Figura se observa un gráfico tipo pastel de la agrupación de registros por nivel de severidad mediante stats count by severity, útil para priorizar incidentes.

Asimismo, en la Figura 66 se presenta una visualización en formato de gráfico de pastel basada en la consulta ejecutada. Esta representación ofrece una visión macro de la distribución de eventos según su nivel de criticidad, lo que facilita el análisis visual, la identificación de patrones y la priorización de incidentes dentro del entorno de monitoreo en *Splunk*.

4.1.2. Alertas y Correlación de Eventos

4.1.2.1. Condiciones de alerta:

- Temperatura > 10 °C: alerta crítica (Evento Controlado)
- Caída de interfaces del switch en simultáneo
- Alerta para eventos de error de Cámara IoT

Dadas las definiciones en el capítulo anterior, la creación de una alerta para detectar la caída simultánea de múltiples interfaces del Switch Core responde a la necesidad de principal de identificar eventos críticos que pueden comprometer la estabilidad de la red y afectar a la productividad en la empresa. Esto está definido como un incidente productivo y suele estar asociada a fallos en el hardware, errores de configuración, pérdida de conectividad troncal o incluso posibles vulnerabilidades de seguridad.

Con ello, al establecer una alerta específica para este patrón o proceso, se garantiza una respuesta rápida o estructurada ante determinadas situaciones que afectan varias zonas de una infraestructura de red al mismo tiempo. Como se observa en la figura 55 se ha establecido una alerta en base a lo explicado sobre la conectividad y estado de las interfaces con el filtro “ALERTAS_INTERFACES” up/down que puede ser usado para el envío de notificaciones por varios canales (*Splunk*/correo), permitiendo a los administradores actuar de forma inmediata para mitigar el impacto y restablecer la operación normal del entorno de red.

Figura 66

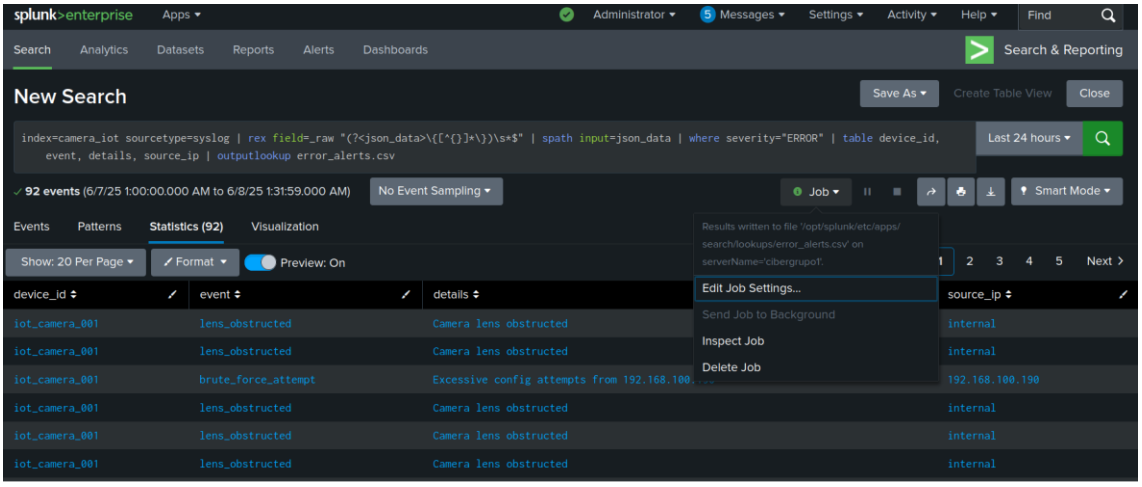
Alerta configurada para monitoreo de estado de interfaces

Nota. En la Figura se observa la alerta basada en el filtro "ALERTAS_INTERFACES" para detectar cambios up/down y activar notificaciones inmediatas.

Así mismo, se ejecuta una búsqueda en *Splunk* que filtra únicamente los mensajes más graves, aquellos clasificados como "ERROR". Los resultados, que incluyen detalles como el ID del dispositivo, el tipo de evento y la dirección IP de origen, se guardan en un archivo llamado error_alerts.csv. Este archivo actúa como un informe recursivo, la Figura 67 muestra como en el administrador de búsquedas de *Splunk* el archivo haya sido creado correctamente.

Figura 67

Panel de búsqueda de eventos



Nota. En la Figura se observa el panel de creación del filtro de búsqueda con registros filtrados por nivel “ERROR”, que incluye detalles clave para análisis y reporte con un archivo de salida csv

Una vez ejecutado el proceso, es posible transformar, analizar y revisar los datos contenidos en el archivo de salida, tal como se muestra en la Figura 68.

Figura 68

Archivo csv de log

	A	B	C	D
1	device_id	event	details	source_ip
2	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
3	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
4	iot_camera_001	brute_force_attempt	Excessive config attempts from 192.168.100.190	192168100190
5	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
6	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
7	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
8	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
9	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
10	iot_camera_001	brute_force_attempt	Excessive config attempts from 192.168.100.190	192168100190
11	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
12	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
13	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
14	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
15	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
16	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
17	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
18	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
19	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
20	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
21	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
22	iot_camera_001	brute_force_attempt	Excessive config attempts from 192.168.100.190	192168100190
23	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
24	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
25	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
26	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
27	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
28	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
29	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
30	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
31	iot_camera_001	lens_obstructed	Camera lens obstructed	internal
32	iot_camera_001	lens_obstructed	Camera lens obstructed	internal

Nota. En la Figura se muestra el archivo generado y descargado de eventos para un análisis

4.1.2.1.1. Acciones.

- **Envió de correo electrónico**

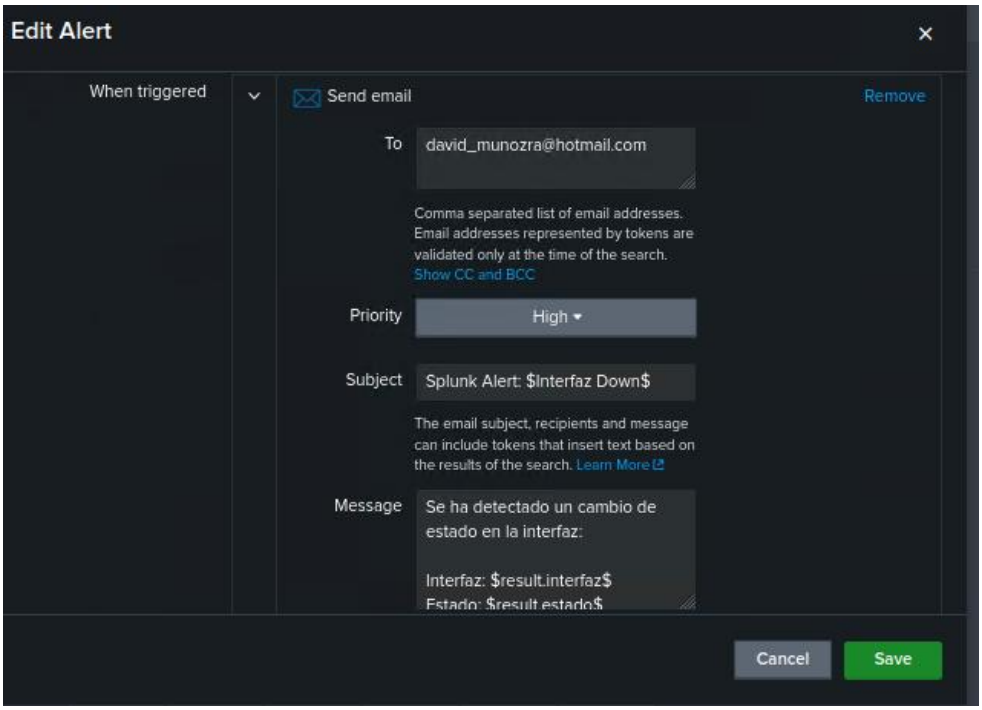
  -Notificación del estado Interfaces y recursos componentes de red

  Dado que ya se encuentra creado el filtro que identifica las interfaces y su estado (“up” o “down”), se continua con la creación de la alerta por correo en *Splunk*, le cual se centró

únicamente en ajustar los parámetros necesarios para su activación y notificación. Esto incluyó definir la condición de las interfaces en estado “down”, se estableció el umbral mínimo de ocurrencias para considerar el evento como crítico, y configurar la acción de envío de correo electrónico con los destinatarios y el mensaje personalizado, como se observa en la figura FDW. Al aprovechar la búsqueda existente, el proceso de creación de la alerta fue directo, permitiendo implementar una notificación automatizada eficiente sin necesidad de modificar la lógica de filtrado previa.

Figura 69

Configuración de parámetros de alerta por correo

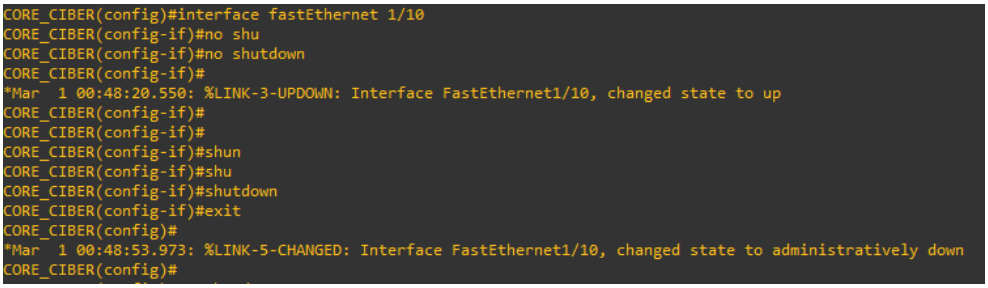


Nota. En la Figura se muestra los parámetros de correo creados

Como se observa en la Figura 61, se realizó una prueba de concepto donde desde el Switch Core se deshabilitó a la interfaz FastEthernet 1/10 haciendo el proceso cuando en un ambiente real se tenga problemas de conexión o fallos en las interfaces.

Figura 70

Deshabilitación de interfaz FastEthernet



Nota. En la Figura se observa la simulación de falla en la interfaz FastEthernet 1/10 para validar la detección de problemas de conectividad en el entorno de red.

Luego de aplicar esta prueba se identificó como el SIEM centralizó los eventos, visto en la figura 71, que por medio de la alerta lanzó la notificación del incidente producido en la interfaz FastEthernet 1/10 “Down”.

Figura 71

Panel de notificación de alerta

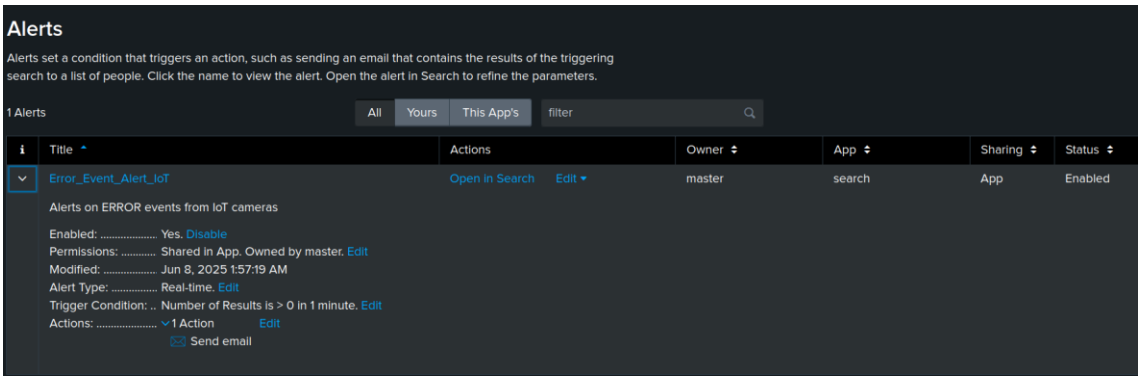
Nota. La Figura visualiza el panel de monitoreo de alertas basadas en la prueba de desconexión

En el caso de la cámara IoT, una vez configurada la búsqueda y guardada la alerta correspondiente, se puede aprovechar esa misma consulta para implementar una notificación automatizada de manera eficiente (Figura 72). Esto permite reutilizar la lógica de filtrado

previamente definida, sin necesidad de modificarla, lo que garantiza consistencia en la detección de eventos críticos.

Figura 72

Descripción de la alerta



Nota. En la Figura se observa la descripción general de la alerta creada para los eventos.

- **Notificación en *Splunk***

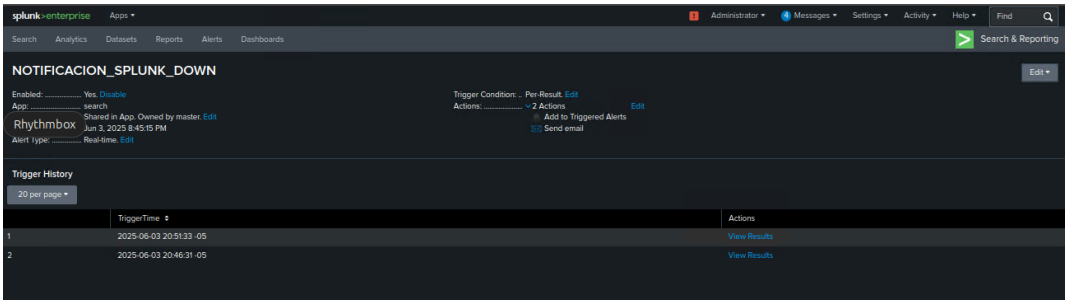
  -Notificación del estado Interfaces y recursos componentes de red

Dado que ya se encuentra creado el filtro que identifica las interfaces y su estado ("*up*" o "*down*"), se continua con la creación de la alerta por notificación en *Splunk*, el cual se centró únicamente en ajustar los parámetros necesarios para su activación y notificación.

Esto incluyó definir la condición de las interfaces en estado "*down*", se estableció el umbral mínimo de ocurrencias para considerar el evento como crítico, y configurar la acción de notificación en la plataforma *Splunk*, como se observa en la figura FzW. Al aprovechar la búsqueda existente, el proceso de creación de la alerta fue directo, permitiendo implementar una notificación automatizada eficiente sin necesidad de modificar la lógica de filtrado previa.

Figura 73

Panel de Alertas

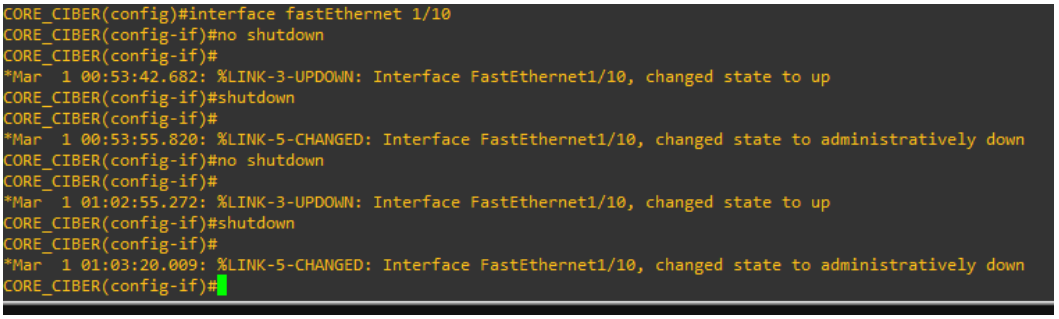


Nota. En la Figura se observa la definición de umbral y notificación automática en Splunk basada en búsqueda existente para detección eficiente de eventos críticos.

Como se observa en la figura 65, se realizó una prueba de concepto donde desde el Switch Core se deshabilitó a la interfaz FastEthernet 1/10 haciendo el proceso cuando en un ambiente real se tenga problemas de conexión o fallos en las interfaces.

Figura 74

Prueba de desconexión de interfaz FastEthernet 1/10



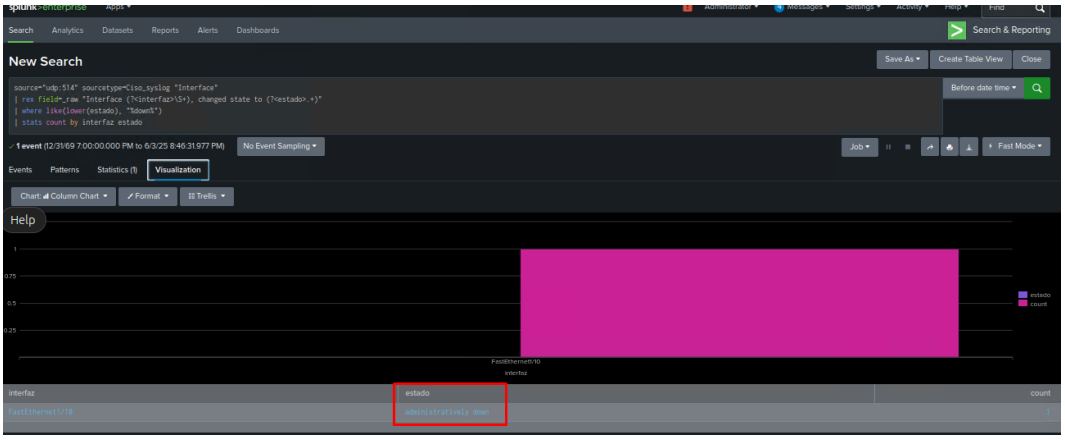
Nota. En la Figura se muestra la simulación de falla en la interfaz para validar la detección de problemas de conectividad en el Switch Core.

Luego de aplicar esta prueba se identificó como el SIEM centralizó los eventos, visto en la Figura 75, y por medio de la alerta lanzó la notificación directa en la plataforma del incidente

producido en la interfaz FastEthernet 1/10 “Down”. Esto es primordial si se tiene un monitoreo constante de los equipos de red en una infraestructura empresarial IoT.

Figura 75

Panel de búsqueda de eventos



Nota. En la Figura se muestra la centralización de eventos y notificación automática en el SIEM tras la caída de la interfaz, facilitando el monitoreo continuo en entornos IoT.

En el caso de la cámara IoT, es posible realizar un análisis de los intentos de inicio de sesión para identificar posibles eventos relacionados con ataques cibernéticos. Una técnica común consiste en detectar múltiples intentos de acceso fallidos desde una misma dirección IP en un corto período de tiempo, lo cual puede ser indicio de un ataque de fuerza bruta.

Para ello, se configura una búsqueda en *Splunk* Figura 76 que contabiliza los eventos de *login* agrupados por dirección IP de origen dentro de una ventana temporal de 5 minutos. Esta estrategia permite identificar patrones de comportamiento repetitivo que podrían representar intentos automatizados de acceso no autorizado.

Figura 76

Búsquedas y filtros de control

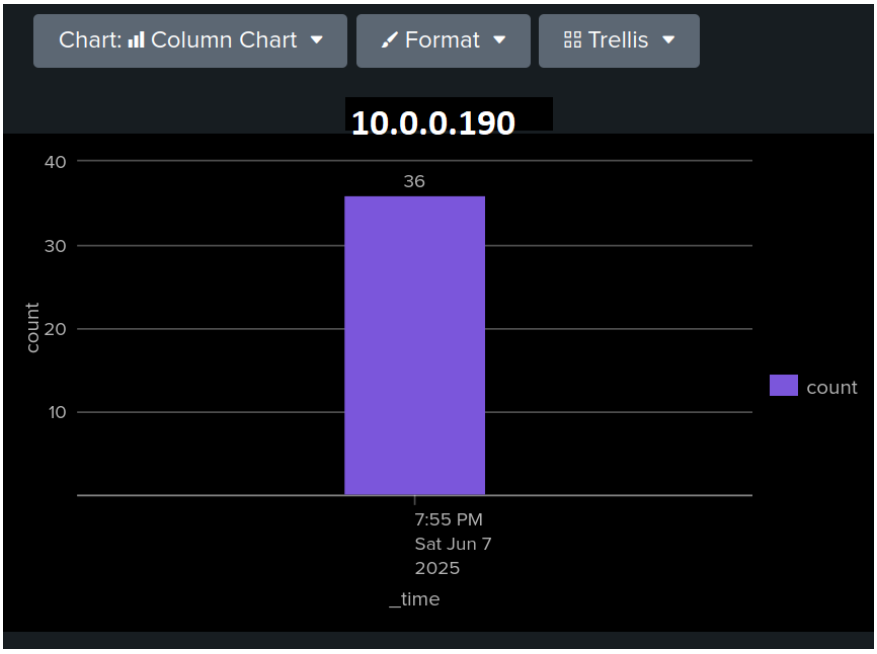
Nota. En la Figura se observa la creación del filtro configurado para detectar patrones repetitivos de acceso potencialmente no autorizado mediante agrupación por IP.

Se puede observar cómo una misma dirección IP genera múltiples eventos relacionados con intentos de inicio de sesión. Este comportamiento refuerza la hipótesis de un posible ataque de fuerza bruta, ya que evidencia una actividad repetitiva y sospechosa desde un único origen.

Para obtener una perspectiva más clara sobre la frecuencia y distribución temporal de estos intentos, se ha creado un gráfico de líneas que representa los eventos de inicio de sesión a lo largo del tiempo. Esta visualización permite identificar picos de actividad anómala y facilita el análisis de patrones que podrían indicar un comportamiento malicioso persistente. Figura 68.

Figura 77

Panel de monitoreo de eventos



Nota. En la Figura se visualización la gráfica de barras que muestra picos y patrones temporales para detectar posibles actividades maliciosas persistentes en determinado tiempo

Ahora, considerando la alerta generada previamente, se ha diseñado una nueva condición de monitoreo orientada a detectar valores de temperatura que superen un umbral crítico en una zona específica. Esta alerta está configurada para activarse automáticamente cuando los datos de sensores simulados reporten temperaturas superiores a los 10 °C, lo cual representa un valor de control en este entorno de simulación, esta alerta llegara a un correo electrónico definido por el administrador del sistema.

Para que *Splunk* pueda enviar correos electrónicos como mecanismo de notificación ante este tipo de eventos, fue necesario habilitar la funcionalidad de envío de correos desde la interfaz administrativa del sistema. Este proceso incluyó los siguientes pasos:

- **Configuración del servidor SMTP**:

  Se definió el servidor saliente de Gmail (smtp.gmail.com) y el puerto 587, activando el uso de TLS para una conexión segura.

- **Autenticación y seguridad**:

  Se utilizó una cuenta de Gmail personal para el proyecto. Debido a las políticas de seguridad de Google, se generó una contraseña de aplicación iqocgctgceysrvot de App Password, lo cual es necesario para que servicios externos como Splunk puedan autenticarse sin comprometer la contraseña principal del correo.

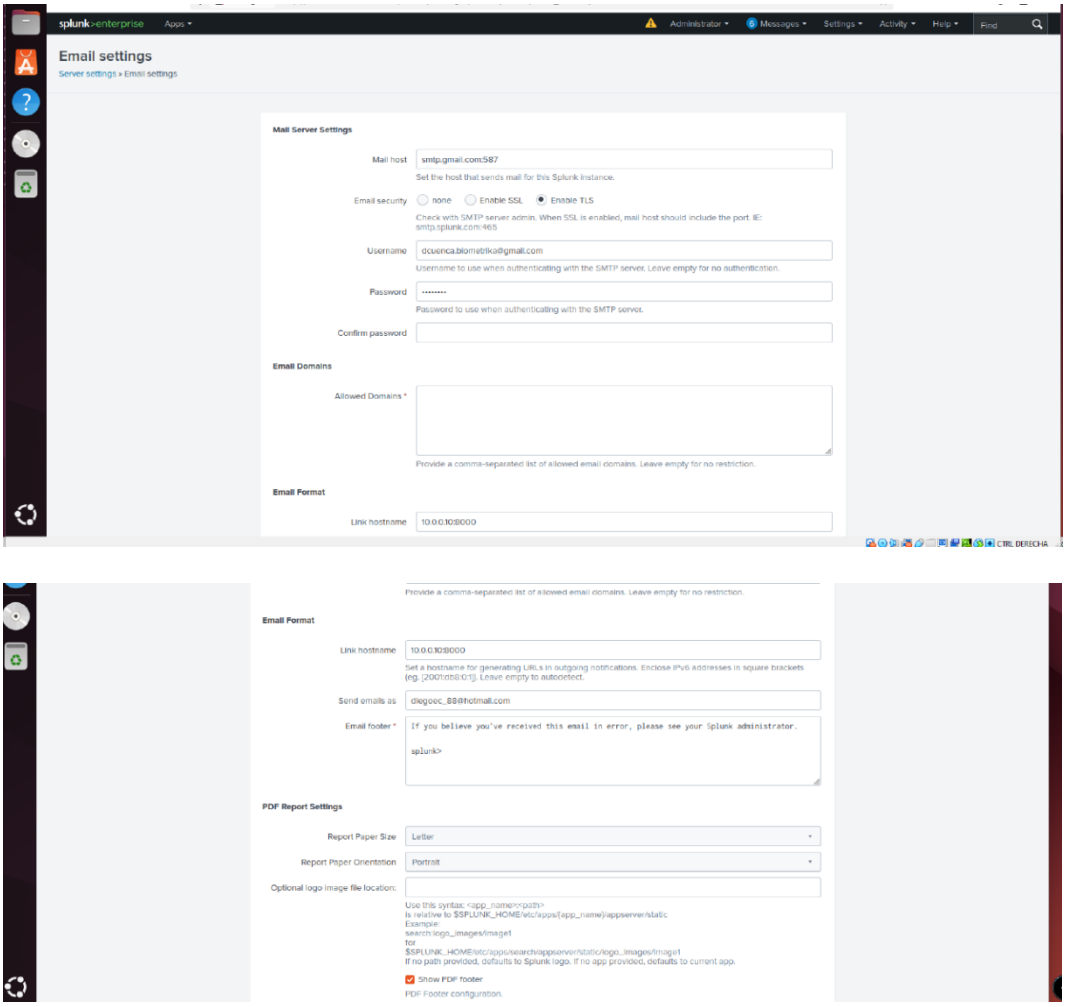
- **Configuración en Splunk**:

  En Settings > Server Settings > Email Settings, se ingresaron:

  a. El servidor SMTP de Gmail.
  b. El correo remitente.
  c. La contraseña de aplicación.
  d. El destinatario del correo que recibirá las notificaciones de alerta.

Figura 78

Configuración General de Email



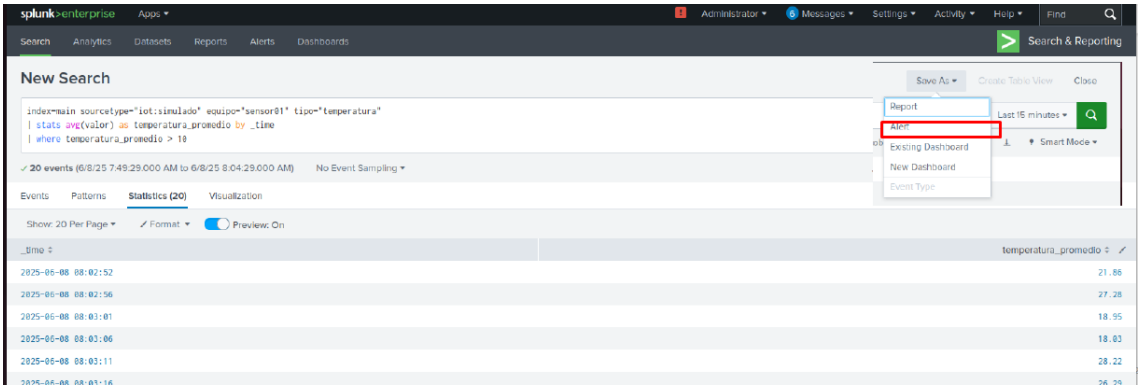
Nota: La figura muestra el panel de configuracion del envio de email desde Splunk para las notificacion via correo

- **Definición de la alerta**:

En la sección *Search & Reporting*, se creó una búsqueda programada que filtra eventos donde el campo temperatura supera los 10 °C. Esta búsqueda se guarda como alerta y se configura con una acción de envío de correo electrónico con asunto y cuerpo personalizados, incluyendo los resultados que dispararon la condición.

Figura 79

Generación de filtro de temperatura



Nota. La Figura muestra el filtro de búsqueda aplicado y los datos recibidos en *Splunk* para la creación de la alerta

Con ello, *Splunk* puede enviar automáticamente notificaciones por correo cuando se detectan valores de temperatura anómalos en tiempo real, mejorando la capacidad de respuesta y visibilidad ante eventos críticos, incluso en un entorno de simulación como el utilizado en este proyecto

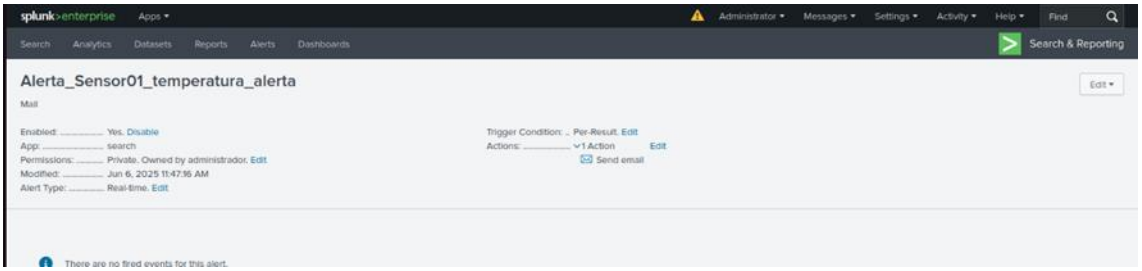
- **Ejecución de la alerta automática**

Esta alerta de tipo critico está relacionada a un sistema de notificación por correo electrónico, permitiendo que el equipo técnico o de gestión reciba en tiempo real una advertencia ante condiciones anómalas.

La visualización demuestra la capacidad para detectar desviaciones críticas y reaccionar de forma proactiva, fortaleciendo la respuesta temprana ante eventos que puedan afectar la operación o la seguridad de los entornos monitoreados.

Figura 80

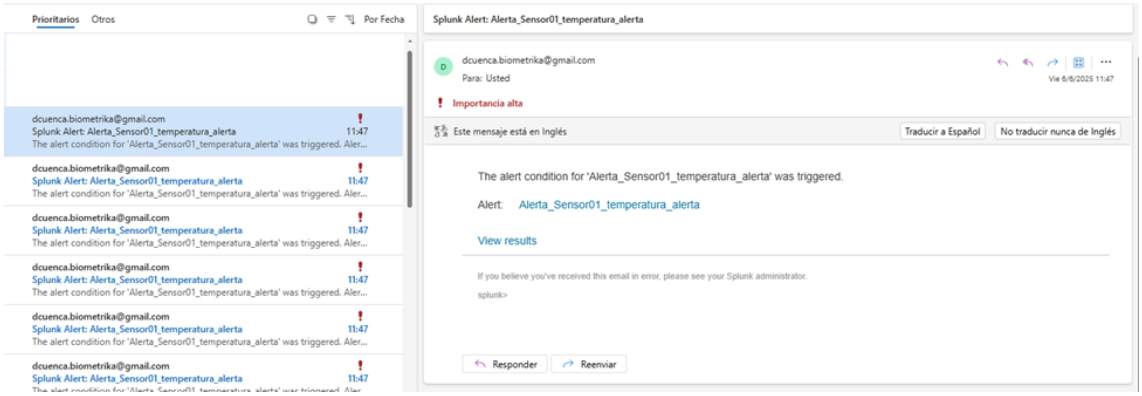
Descripción de Alerta



Nota. En la Figura muestra la creación y el estado de la alerta de correo creada para eventos de temperatura que sobrepase los 10°C

Figura 81

Visualización de correo electrónico en bandeja de entrada



Nota. La Figura muestra la recepción correcta del correo enviado desde *Splunk* cuando se genera la alerta de temperatura

Gracias a la flexibilidad del *Splunk* y al uso de condiciones personalizadas basadas en búsquedas, es posible definir alertas altamente específicas que se adapten a las necesidades de la empresa o al contexto de seguridad de una organización. En el caso de este proyecto, las alertas

se diseñaron para monitorear los datos simulados de sensores y otros dispositivos IoT, demostrando la utilidad del sistema para anticipar condiciones fuera de parámetros y facilitar una acción temprana.

4.2. Análisis de resultados

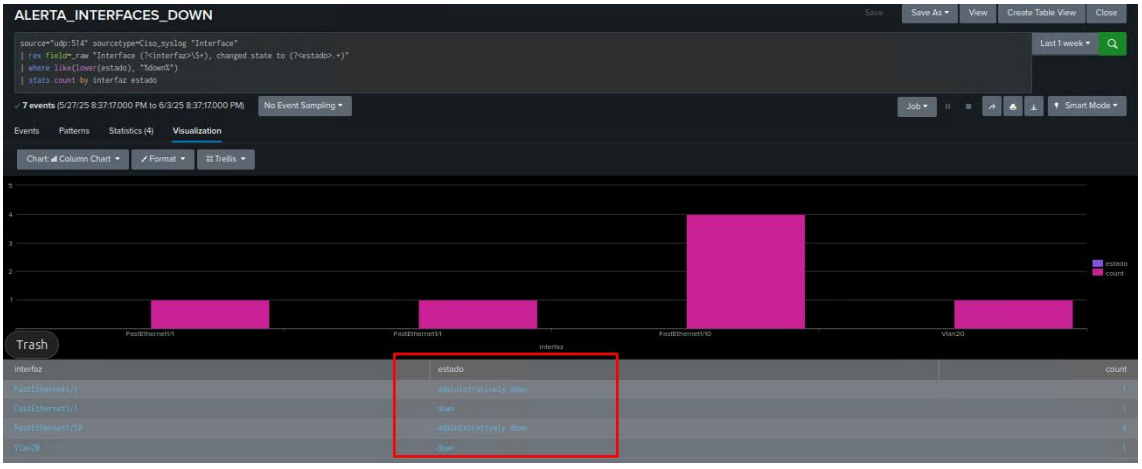
4.2.1. Pruebas y Resultados

- Se simularon eventos de temperatura crítica
- Se simularon eventos sobre el dispositivo IoT que simula una cámara
- Se mostraron los resultados en *dashboards* y logs en tiempo real

Durante la fase de validación, se realizaron pruebas controladas simulando fallas en interfaces de red y condiciones críticas para las conexiones con dispositivos IoT conectados. Los datos recopilados fueron visualizados en tiempo real a través de *dashboards* configurados en *Splunk*, permitiendo observar los cambios en el estado de las interfaces y los eventos generados por este comportamiento. Como se observa en la Figura 82 que muestra en resumen como el SIEM va interpretando e integrando los eventos de los componentes de red (Switch Core) en la infraestructura de red diseñada para este proyecto.

Figura 82

Panel de Monitoreo de Interfaces

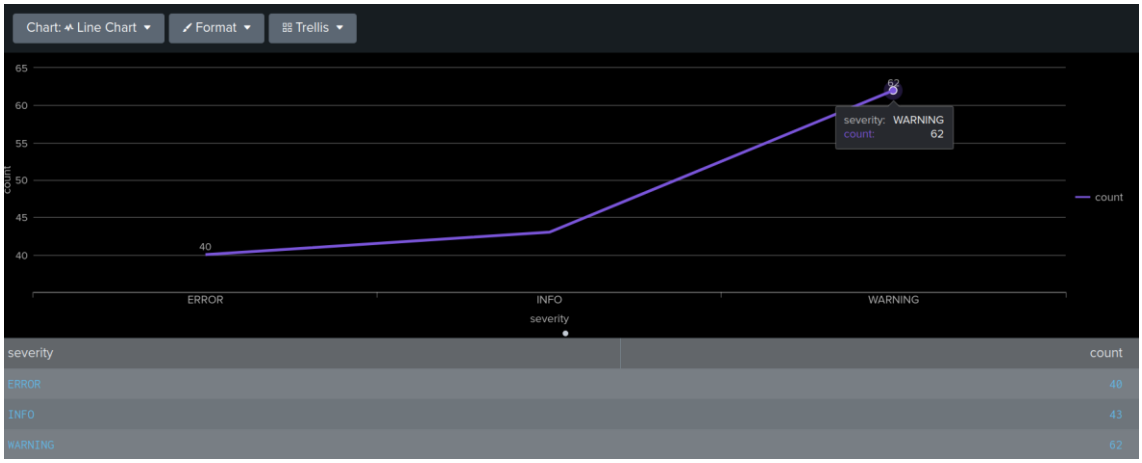


Nota. La Figura muestra el consolidado de eventos por medio del panel de alertas creado para una fácil visualización y control de incidentes

Asimismo, un dispositivo IoT como una cámara puede correlacionar eventos dentro de un *dashboard* en *Splunk*, lo que permite visualizar e interpretar su comportamiento operativo en tiempo real. Esta capacidad de correlación facilita la evaluación del correcto funcionamiento del dispositivo, mejora la visibilidad del entorno y respalda la toma de decisiones informadas. El objetivo principal de esta funcionalidad es demostrar el alcance y la utilidad de la herramienta para el monitoreo inteligente y proactivo de dispositivos IoT conectados de manera centralizada. Figura 83.

Figura 83

Dashboard de correlación y monitoreo en tiempo real de cámara IoT



Nota. La Figura expone la visualización centralizada que facilita la evaluación y seguimiento operativo del dispositivo para un monitoreo proactivo.

Uno de los principales objetivos de la prueba fue verificar el correcto funcionamiento del filtro que identifica interfaces en estado "*down*", la cámara IoT que evidencia eventos. Al cumplirse esta condición, se comprobó que el sistema generó alertas automáticas tal como estaba

previsto. Las notificaciones por correo electrónico llegaron a los destinatarios definidos, con el mensaje personalizado y los detalles del evento. Asimismo, las notificaciones internas de Splunk se activaron de forma oportuna, registrando los incidentes en la plataforma.

Estos resultados confirman que la lógica de correlación, las condiciones de activación y los mecanismos de alerta del sistema funcionan adecuadamente, cumpliendo con los objetivos del monitoreo automatizado de seguridad en el entorno simulado. Contextualizado en la Tabla 13.

Tabla 13

Resultados de la integración y monitoreo del entorno IoT y red

Característica	Resultado
Integración Componentes de red (Switch Core)	Integración completa (Envío y Recepción eventos Netwoking) SIEM red IOT
Centralización de logs	Filtración y agrupación completa en la Plataforma Splunk
Visualización de eventos	Construcción de dashboard interfaces y recursos completada
Alertas y notificaciones	Gestión por correo y notificación Splunk completada

Nota. La tabla se resumen los resultados alcanzados en la integración de componentes de red y IoT con Splunk, destacando la centralización, visualización y gestión de alertas para un monitoreo efectivo.

Finalmente, para ir comprobando la viabilidad de la integración entre dispositivos IoT y el sistema SIEM utilizado que es *Splunk*, se diseñó un entorno de pruebas controlado que simula

las condiciones de una red heterogénea real. Este entorno no solo permite generar eventos desde sensores virtuales mediante el protocolo MQTT, sino también observar su tránsito hacia *Splunk* utilizando un puente programado en Python. La finalidad de esta prueba es verificar que los eventos generados por dispositivos distribuidos pueden ser detectados, procesados y visualizados en tiempo real, fortaleciendo la visibilidad de seguridad. Entonces, se demuestra que el modelo propuesto puede aplicarse como base para futuras implementaciones en organizaciones que requieran monitoreo proactivo y adaptable de sus activos IoT.

Para una mejor visualización de datos se crea un *dashboard* consolidado que permite tener una visión completa y comparativa de los datos y sensores que se integran en la red en donde se valida los siguientes puntos:

- Identificar patrones normales de funcionamiento.
- Detectar variaciones anómalas o fuera de rango.
- Analizar condiciones ambientales simuladas desde múltiples fuentes.

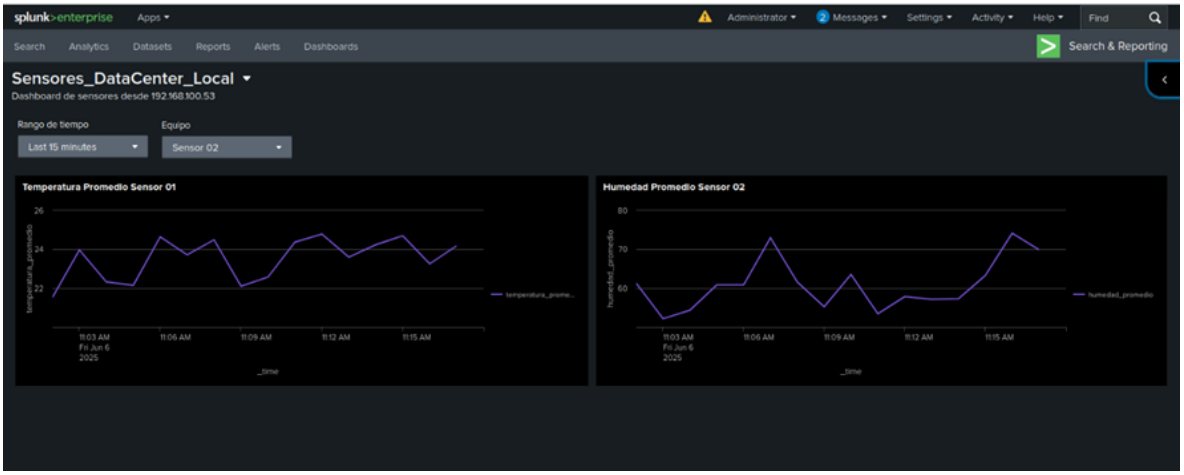
Observaciones identificadas con la implementación:

- Se evidenció flujo constante de eventos cada 5 segundos, tal como fue programado en el simulador.
- Los datos llegaron estructurados correctamente en formato JSON, lo cual permitió su análisis y visualización sin necesidad de *parsers* adicionales.
- La gráfica tipo lineal generada en el *dashboard* permitió observar de forma clara el flujo de datos.
- La simulación se comportó de manera consistente durante el tiempo de prueba.

- No se registraron pérdidas de datos ni errores de conexión entre los scripts y el HEC, lo cual demuestra una correcta configuración del entorno.

Figura 84

DashBoard Consolidado de Monitoreo en tiempo real



Nota. La Figura muestra el monitoreo simultaneo de los sensores de temperatura y humedad en una zona critica

En la Figura se verifica la eficacia de la integración técnica entre los sensores, el *broker* MQTT y la plataforma *Splunk*, demostrando la capacidad del sistema para consolidar y analizar datos multisensoriales en entornos empresariales, como el de las *PYMEs* ecuatorianas.

Las figuras desarrolladas por el equipo evidencian una correlación efectiva y estable entre los sensores y el sistema SIEM, demostrando la viabilidad de la solución propuesta para mejorar la visibilidad de seguridad en entornos con dispositivos IoT.

Se realizó un análisis comparativo entre cuatro soluciones de monitoreo de seguridad con capacidad de integrar dispositivos IoT como muestra la Tabla 14.

- Diseño propuesto (Splunk + MQTT + SNMP)
- Wazuh + MQTT
- ELK Stack (Elastic + Logstash + Kibana)
- Zabbix IoT

Tabla 14

Cuadro Comparativo: Sistemas de Monitoreo de Seguridad IoT para PYMEs en Ecuador

Criterio	Propuesta: Splunk + MQTT + SNMP	Wazuh + MQTT	ELK Stack (Elastic + Logstash + Kibana)	Zabbix IoT
Costos de Licenciamiento	Medio (Splunk free limitado a 500 MB/día, versión Enterprise requiere licencia por GB/día)	Bajo (Open source)	Bajo (Open source, pero requiere configuración manual y recursos propios)	Bajo (Open source)
Tiempo de Implementación	Medio: ~1-2 meses para integración completa, dashboards, alertas y pruebas	Medio: ~1 mes para integración básica, dashboards y alertas	Alto: ~2-3 meses por integración manual de pipelines y dashboards	Medio-Bajo: ~1 mes para integración básica de SNMP y plantillas IoT
Compatibilidad IoT (MQTT, SNMP)	Alta: MQTT broker + SNMP integrados, dashboards	Alta: Soporte MQTT con módulos externos, SNMP	Media: Requiere parsers y pipelines	Alta: SNMP nativo, plantillas IoT, soporte limitado

	personalizados, puente Python para eventos IoT en tiempo real	nativo, integración manual de brokers	manuales para MQTT/SNMP	a MQTT (requiere scripts externos)
Visualización y Correlación	Alta: Dashboards en Splunk con lógica de correlación y alertas automáticas en tiempo real	Media-Alta: Dashboards básicos en Kibana o Wazuh app, correlación limitada	Media-Alta: Visualización potente en Kibana, pero pipelines manuales para correlación	Media: Dashboards predefinidos, correlación limitada, reportes básicos
Escalabilidad	Alta: Soporta grandes volúmenes de logs y múltiples fuentes IoT sin pérdida de datos	Media-Alta: Depende de hardware propio, pero flexible para escalar	Alta: Muy escalable, pero requiere recursos y administración avanzada	Media: Escalabilidad limitada en entornos grandes
Alertas y Notificaciones	Completo: Correo, alertas internas Splunk, triggers automáticos	Completo: Alertas email, dashboards, integración SIEM	Completo: Alertas, notificaciones, pipelines custom	Completo: Alertas email/SMS, acciones automatizadas básicas
Ventajas	Visualización en tiempo real robusta Alta compatibilidad con IoT	Gratis, comunidad activa Fácil integración con syslog/SNMP SIEM integrado	Potente motor de búsqueda y visualización Escalable Comunidad grande	Fácil de instalar Plantillas SNMP listas Buen soporte a dispositivos de red

	Filtro de interfaces “down” probado Notificaciones automáticas confiables Soporte técnico disponible			
Desventajas	Costo de licencia para grandes volúmenes de datos Requiere conocimientos de configuración avanzada	Curva de aprendizaje en reglas personalizadas Requiere tuning manual No contiene un enfoque de monitoreo snmp nativo Soporte y visualización limitada externa	Complejidad en configuración de pipelines Mayor esfuerzo de mantenimiento	Visualización menos robusta Limitada correlación de eventos IoT

Nota. Tabla muestra la comparativa de tecnologías para el monitoreo de sistemas de comunicación IoT con sus pros y contras en beneficio de encontrar un punto de análisis de este diseño propuesto.

Los criterios principales fueron: costos, tiempo de implementación, compatibilidad con protocolos IoT, visualización, escalabilidad, alertas y adaptabilidad a redes IoT de PYMEs en Ecuador como se muestra en la Tabla 15.

Tabla 15

Resultados principales de las tecnologías que más se adaptan el giro de negocio

Aspecto Clave	Mejor Evaluado	Resumen
Costos	Wazuh, ELK, Zabbix	Son gratuitos, pero requieren más tiempo de configuración manual.
Tiempo de implementación	Zabbix, Splunk	Instalación rápida para monitoreo básico, pero limitado para IoT avanzado.
Compatibilidad IoT	Splunk + MQTT + SNMP	Soporte nativo y robusto para MQTT, SNMP y puente IoT con scripts Python.
Visualización y Correlación	Splunk	Dashboards avanzados, correlación real de eventos IoT + red en tiempo real.
Escalabilidad IoT	Splunk, ELK	Escalan bien con grandes volúmenes de datos IoT; Zabbix y Wazuh se limitan según hardware.
Alertas Automáticas IoT	Splunk	Alertas en tiempo real, correo y triggers automáticos configurados y probados.
Ventajas Clave	Splunk	Integración completa IoT-red, monitoreo proactivo, sin pérdidas de datos, flujo estable y dashboards personalizados.
Desventajas	Splunk	Requiere licencia pagada si se superan límites de datos; demanda conocimiento técnico avanzado.

Nota. Tabla muestra los resultados de manera específica apuntando a los principales entes de análisis hacia el costo/beneficio en donde Splunk se destaca por tener más factores técnicos para un sistema centralizado, escalable y estable.

De acuerdo con los resultados de las pruebas y esta comparación:

El diseño (Splunk + MQTT + SNMP) demuestra ser la solución más robusta para PYMEs en ecuador sabiendo que a nivel general no se tiene una inversión adecuada para la ciberseguridad empresarial, en este contexto:

- Ofrece alta visibilidad y correlación de eventos IoT y red en tiempo real.
- Cumple con alertas automáticas confiables y notificaciones por correo interno.
- Es adaptable y escalable para crecer con la infraestructura de una PyME.
- Valida eventos multisensoriales y estados de red, mejorando la capacidad de respuesta proactiva.
- Aunque su costo de licencia puede ser superior para altos volúmenes, se compensa con su fiabilidad y soporte técnico.

Por tanto, si una Pyme busca un monitoreo de seguridad IoT robusto y centralizado, para garantizar la integración, visualización en tiempo real, correlación de eventos y generación de alertas en entornos IoT, adaptándose de manera efectiva a las necesidades de seguridad de las PYMEs en Ecuador. Se puede enfatizar que este diseño cumple con los objetivos de análisis y si se pretende implementar se recomendada la estabilización tras 1 año de análisis de la red y pruebas piloto.

CAPITULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1.Conclusiones

Con base en los resultados obtenidos y la integración técnica realizada, se concluye que el sistema diseñado cumple con los objetivos establecidos para el monitoreo y la seguridad de redes IoT en pequeñas y medianas empresas (PYMEs) del Ecuador. *La integración del switch core dentro del entorno simulado permitió una visibilidad completa sobre el comportamiento del tráfico y los eventos de red, lo cual es fundamental para identificar posibles incidentes o vulnerabilidades.*

Así mismo, la incorporación de dispositivos IoT como la simulación de una cámara y sensores de temperatura y humedad permitió una visibilidad detallada de los eventos generados por el dispositivo, facilitando la detección de comportamientos anómalos, intentos de acceso no autorizados y fallos operativos. Esta visibilidad es importante para fortalecer la postura de seguridad, optimizar la gestión de incidentes y respaldar decisiones informadas en tiempo real.

La centralización de logs fue exitosa, permitiendo filtrar, clasificar y agrupar los eventos relevantes provenientes de múltiples fuentes IoT y de infraestructura. Gracias a esta consolidación, fue posible construir *dashboards* que ofrecen una visualización clara y dinámica del estado de las interfaces de red, eventos dispositivos IoT, recursos del sistema y comportamiento de los sensores.

El sistema de alertas y notificaciones demostró ser eficaz, generando respuestas automáticas ante condiciones críticas, tanto por correo electrónico como dentro de la plataforma *Splunk*. Esto permite no solo detectar fallos o comportamientos anómalos de zonas críticas u otras que dependen de la entidad, sino también actuar con rapidez, fortaleciendo la capacidad de

respuesta ante incidentes de seguridad en entornos de red IoT, especialmente relevantes para pequeñas y medianas empresas.

El entorno simulado, que permitió probar todo el ciclo de generación, transmisión y monitoreo de datos, sienta así una base sólida para futuras implementaciones reales en redes heterogéneas, facilitando escenarios más robustos de análisis y correlación de eventos para una ciberseguridad proactiva.

Finalmente se indica que este trabajo de fin de maestría presenta el diseño de un sistema virtualizado de monitoreo de seguridad orientado a *PYMEs* del Ecuador, integrando la plataforma *Splunk* con los protocolos MQTT y SNMP. La solución, desarrollada en entornos simulados, permite evaluar de una forma más rápida la implementación de este tipo de soluciones y la supervisión en tiempo real de dispositivos IoT, la detección de eventos anómalos y la generación automatizada de alertas de seguridad y lo que llevaría a una presentación de resultados con los factores a considerar en una empresa y a la toma de decisiones sobre este tipo de soluciones con las altas gerencias.

El enfoque del proyecto demuestra que, sin requerir grandes inversiones en infraestructura, es posible construir entornos funcionales y escalables que mejoran la visibilidad de seguridad en redes heterogéneas, fortaleciendo las capacidades de respuesta ante incidentes en empresas con recursos limitados.

5.2.Recomendaciones

Como recomendación clave derivada de este diseño realizado, se propone que, si el sistema es implementado en un entorno real, se acompañe de un plan de análisis continuo durante al menos un año, para poder compararlo, orientado a recopilar, evaluar y refinar los datos generados por todos los componentes que conforman la red de comunicación IoT. Este periodo

de tiempo permitirá comprender a fondo los patrones de comportamiento normales y anómalos que surgieron en ese año y optimizar las reglas de correlación, ajustar los umbrales de alerta y validar la cobertura de seguridad frente a eventos reales para este nuevo periodo minimizando riesgo y vulnerabilidades de seguridad. Bajo este enfoque, al término del primer año, la solución basada en *Splunk* no solo habrá madurado operativamente, sino que estará afinada para actuar como un SIEM optimizado, alineado con las necesidades específicas de la infraestructura de cada PYME. Este proceso de adaptación continua permitirá que el sistema evolucione de una solución (sistema) de monitoreo a una herramienta robusta de ciberseguridad, capaz de ofrecer una visibilidad centralizada, mejorar la toma de decisiones y brindar una protección más eficaz frente a amenazas que puedan afectar la continuidad operativa de las PYMEs.

Muchas PYMEs en Ecuador no cuentan con conocimientos suficientes en ciberseguridad, por lo que es fundamental acompañar las soluciones técnicas con capacitaciones prácticas y materiales fáciles de entender. Se sugiere crear guías paso a paso para interpretar paneles y alertas en *Splunk*, dirigidas a personas sin formación técnica. Estas guías deben incluir ejemplos claros, como qué hacer si se detecta una caída en la red, intentos de acceso no autorizados a cámaras de seguridad o dispositivos sospechosos. También deben explicar acciones básicas recomendadas, como aislar equipos comprometidos. Además, se puede mejorar la detección de amenazas integrando *Splunk* con herramientas que identifiquen patrones peligrosos en protocolos como MQTT, comunes en dispositivos IoT. Esto permitiría actuar antes de que un ataque cause daños.

Se recomienda que, en caso de llevarse a cabo una implementación en un entorno productivo, esta se acompañe de un plan detallado de análisis y dimensionamiento qué contemple inventario de recursos, componentes, estimación del tiempo de uso de licencias y

evaluación de capacidades técnicas del entorno, esto con el objetivo de maximizar el aprovechamiento de las funcionalidades que ofrece *Splunk*, alineadas con las necesidades reales del negocio.

Se sugiere extender el periodo de observación más allá del tiempo de evaluación inicial con la finalidad de comprender en profundidad los patrones de comportamiento normales y anómalos con la finalidad de garantizar una transición más segura y eficiente hacia una implementación productiva.

El laboratorio se ha construido sobre una arquitectura de red simplificada, que incorpora dispositivos perimetrales y controles de seguridad básicos, representativos de un entorno típico en pequeñas y medianas empresas. Esta elección responde a la necesidad de simular un escenario realista y alcanzable, donde se puedan evaluar capacidades de monitoreo y respuesta sin requerir una infraestructura compleja o costosa, no obstante, se recomienda controles adicionales como segmentación, controles de acceso a la red y mecanismos de detección de intrusos (IPS/IDS) para elevar el nivel de protección y fortalecer la seguridad.

Referencias Bibliográficas

Alaisecure. (s. f.). ¿Cuáles son los dispositivos IoT?: Ejemplos. AlaiSecure - Internacional. https://www.alaisecure.com/blog/cuales-son-los-dispositivos-iot-ejemplos/

Amazon Web Services. (s. f.). ¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/mqtt/

Bueno, G., & Haz, L. (2022). Ciberseguridad post Covid-19 y su impacto en las pymes del Ecuador. Revista de Producción Ciencia e Investigación Pro Sciences, 6(46), 103-120. doi:https://doi.org/10.29018/issn.2588-1000vol6iss46.2022pp103-120

Carrión, M. (2024). ¿Qué es IoT? El Internet de las Cosas. ITSQMET. https://itsqmet.edu.ec/iot/

CyberArrow team. (2025). ¿Qué es Splunk? ¿Para qué se utiliza Splunk? | CyberArrow. https://www.cyberarrow.io/blog/what-is-splunk-what-is-splunk-used-for/

Digi. (s. f.). ¿Qué es Zigbee? Más información sobre la tecnología de malla inalámbrica Zigbee. https://es.digi.com/solutions/by-technology/zigbee-wireless-standard

Del Rio, L. (2023, enero 10). Splunk, la herramienta para anticiparse a los ataques. Computing. https://www.computing.es/seguridad/splunk-la-herramienta-para-anticiparse-a-los-ataques/

Eclipse Foundation. (2018). Eclipse Mosquitto. Eclipse Mosquitto. https://mosquitto.org/

Fortinet. (s. f.). ¿Qué es el IoT (IoT)? Definición, beneficios y desafíos. Fortinet. https://www.fortinet.com/lat/resources/cyberglossary/iot.html

Fortinet. (s. f.). ¿Qué es el Protocolo simple de administración de red (SNMP)? ¿Es seguro? Fortinet. https://www.fortinet.com/lat/resources/cyberglossary/simple-network-management-protocol.html

Fortinet. (s. f.). ¿Qué es la calidad de servicio (QoS) en las redes? Fortinet. https://www.fortinet.com/lat/resources/cyberglossary/qos-quality-of-service.html

García de Zúñiga, F. (s. f.). Python: ¿qué es y para qué sirve? | Blog de Arsys. Arsys. https://www.arsys.es/blog/python-que-es-y-para-que-sirve

GNS3. (s. f.). Getting Started with GNS3 | GNS3 Documentation. https://mother.github.io/docs/

Homey. (s. f.). ¿Qué es Zigbee? La tecnología de red eléctrica inteligente más popular del mundo. https://homey.app/es-es/wiki/que-es-zigbee/

International Business Machines. (2023). ¿Qué es el Internet de las cosas (IoT)? | IBM. https://www.ibm.com/mx-es/topics/internet-of-things

Jude, A. X. (2025). MQTT Protocol in IoT: A Guide to Reliable IoT Communication. cavliwireless.com. https://uploads-hubblethings.s3.eu-west-1.amazonaws.com/strapi/mqtt_thumb_a407b341ff.png

Kaspersky (2025). ¿Qué es la Internet de las cosas? Definición y explicación. Obtenido de https://latam.kaspersky.com/resource-center/definitions/what-is-iot?srsltid=AfmBOorWavfV_W3zB3V7Fn7VstxZDd_luNYj2I-3bauJyBNFwFGpzMzo

Kaspersky (2025). ¿Qué es la Internet de las cosas? Definición y explicación. Obtenido de https://latam.kaspersky.com/resource-center/definitions/what-is-iot?srsltid=AfmBOorWavfV_W3zB3V7Fn7VstxZDd_luNYj2I-3bauJyBNFwFGpzMzo

Kriesche, P. (2017). Humans vs. Machines – who will manage the factory of the future? Technology and Operations Management. https://d3.harvard.edu/platform-rctom/submission/humans-vs-machines-who-will-manage-the-factory-of-the-future/

López, B. (2024). Obtenido de https://digibuo.uniovi.es/dspace/bitstream/handle/10651/74071/TFM_TonnySocorrodelPozo.pdf?sequence=7&isAllowed=y

Muylinux. (2024, abril 25). *Disponible Ubuntu 24.04 LTS, novedades y descarga*. https://www.muylinux.com/2024/04/25/ubuntu-24-04-lts/

Omnitron Systems. (s. f.). What Is SNMP? How Does It Work? Omnitron Systems. https://www.omnitron-systems.com/blog/what-is-snmp-how-does-it-work

Paessler The Monitoring Experts. (s.f.). ¿Qué es SNMP? - Definición y detalles. https://www.paessler.com/es/it-explained/snmp

Piensa Solutions. (s. f.). ¿Qué es Ubuntu? Guía sobre este sistema operativo. https://www.piensasolutions.com/blog/que-es-ubuntu-guia-sobre-este-sistema-operativo

Romero, R. (2021). Obtenido de https://openaccess.uoc.edu/handle/10609/132147

Splunk Cisco. (s. f.). What Is Splunk & What Does It Do? A Splunk Intro. Splunk. https://www.splunk.com/en_us/blog/learn/what-splunk-does.html

Splunk Inc. (2025). *The CISO report 2025. https://www.splunk.com/en_us/campaigns/ciso-report.html*

Summer, W. (s. f.). ¿Qué es Python? Todo lo que necesitas saber para empezar. ¿Qué es Python? Todo lo que necesitas saber para empezar. https://www.datacamp.com/blog/all-about-python-the-most-versatile-programming-language

Techtarget. (s. f.). What is Simple Network Management Protocol (SNMP)? | Definition from TechTarget. Search Networking. https://www.techtarget.com/searchnetworking/definition/SNMP

Venco. (s. f.). Qué es ZigBee, cómo funciona y características principales—Venco Electrónica. https://www.vencoel.com/que-es-zigbee-como-funciona-y-caracteristicas-principales/

Walton, A. (2024). ¿Qué es GNS3? Para qué Sirve y Cómo Instalarlo» CCNA desde Cero. CCNA desde Cero. https://ccnadesdecero.es/que-es-gns3-como-usarlo/

Apéndice

a. Documentación

Desarrollo del Trabajo

Descripción General

Este documento describe los pasos para configurar un entorno de simulación IoT usando *Splunk* como sistema SIEM, Mosquitto como *broker* MQTT, y *scripts Python* que simulan dispositivos IoT desde una VM cliente. Se incluye:

Verificación de servicios

Creación de HEC en *Splunk*.

Configuración del *broker* MQTT.

Despliegue del script simulador en la VM cliente.

Visualización de eventos en *Splunk*.

Infraestructura

Servidor (10.0.0.10):

Splunk.

Broker MQTT (Mosquitto).

Cliente (10.0.0.12):

Simulador de dispositivo IoT (*Python Script*).

Script puente MQTT-*Splunk.*

Instalación y Configuración en el Servidor

Los siguientes pasos se ejecutan luego de la instalación completa de *Splunk*.

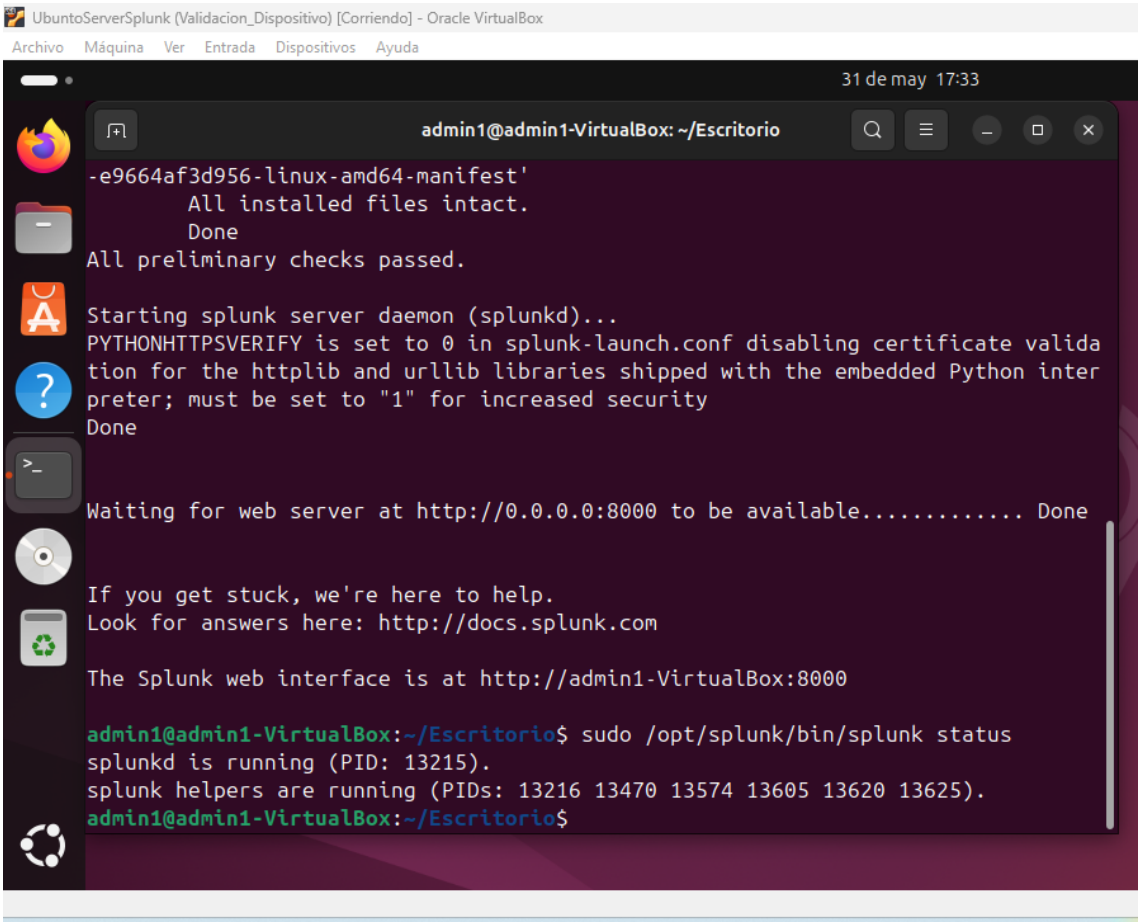
Verificar e iniciar *Splunk*

sudo /opt/splunk/bin/splunk status

sudo /opt/splunk/bin/splunk start

Figura 85

Verificar e iniciar Splunk



Nota. En la Figura se observa el visual que indica que *Splunk* está ejecutándose de manera correcta en el servidor.

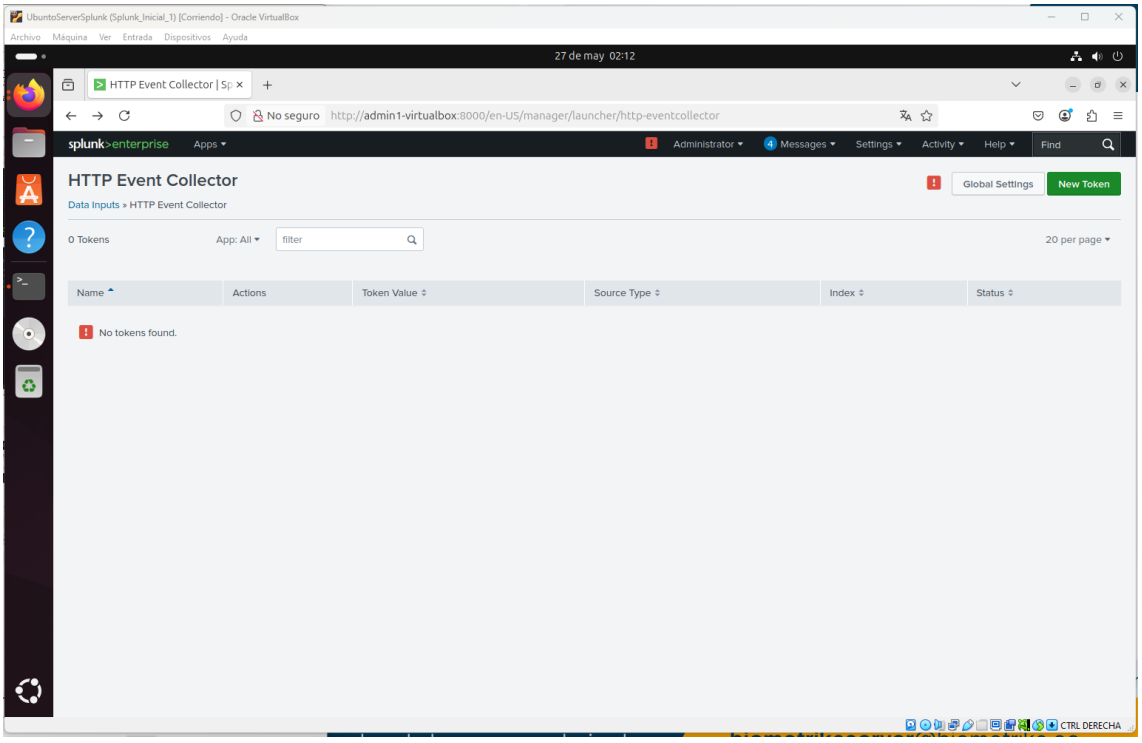
Habilitar HTTP Event Collector (HEC)

Accede a *Splunk* en http://10.0.0.10:8000 como administrador (cuenta local creada).

Ir a Settings > Data inputs > HTTP Event Collector.

Figura 86

Panel de HEC

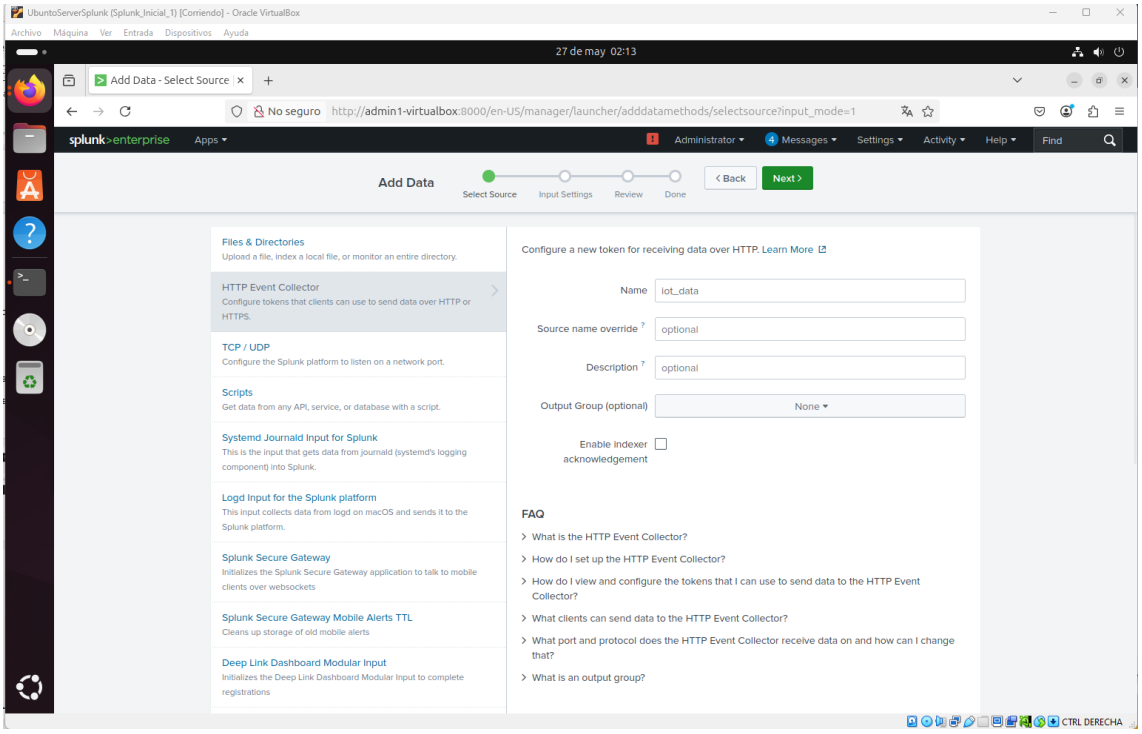


Nota. En la Figura se observa la ventana 1 de gestión de HEC.

Crea un nuevo *Token* (ej. Iot_data, iot_simulacion o cualquier nombre identificativo). De igual forma se debe incluir la opción MAIN para la búsqueda de eventos.

Figura 87

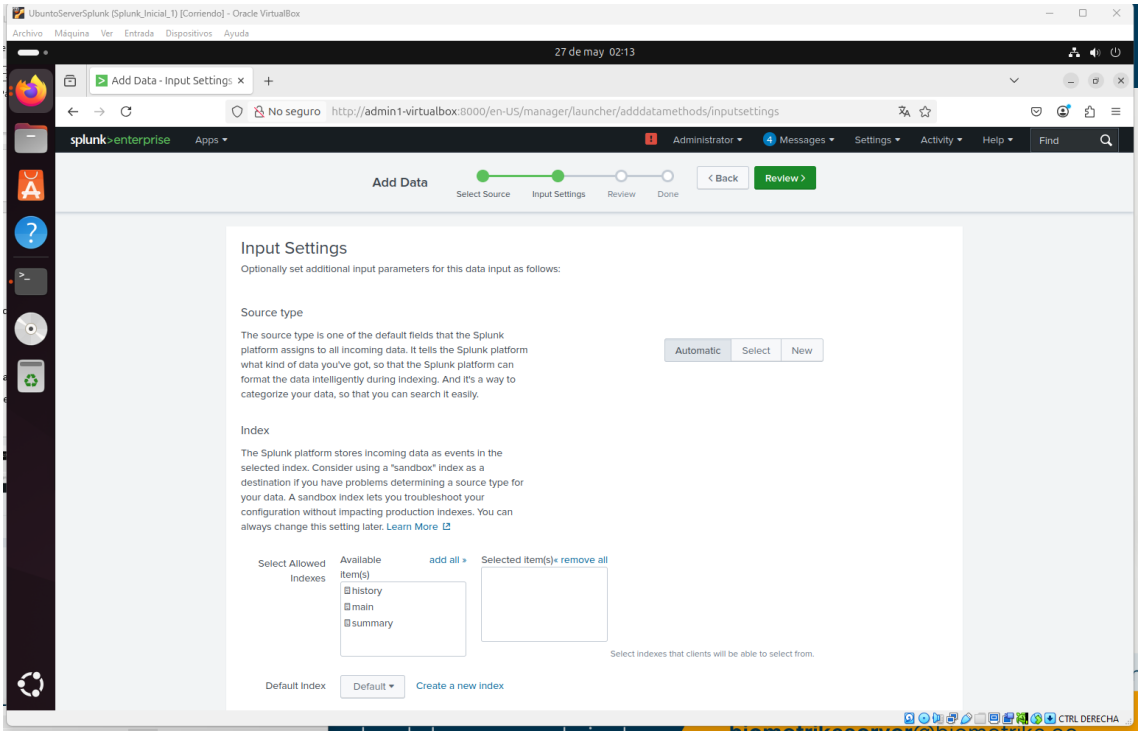
Creación del token para la integración



Nota. En la Figura se mira la ventana 2 de creación de la nueva descripción del token.

Figura 88

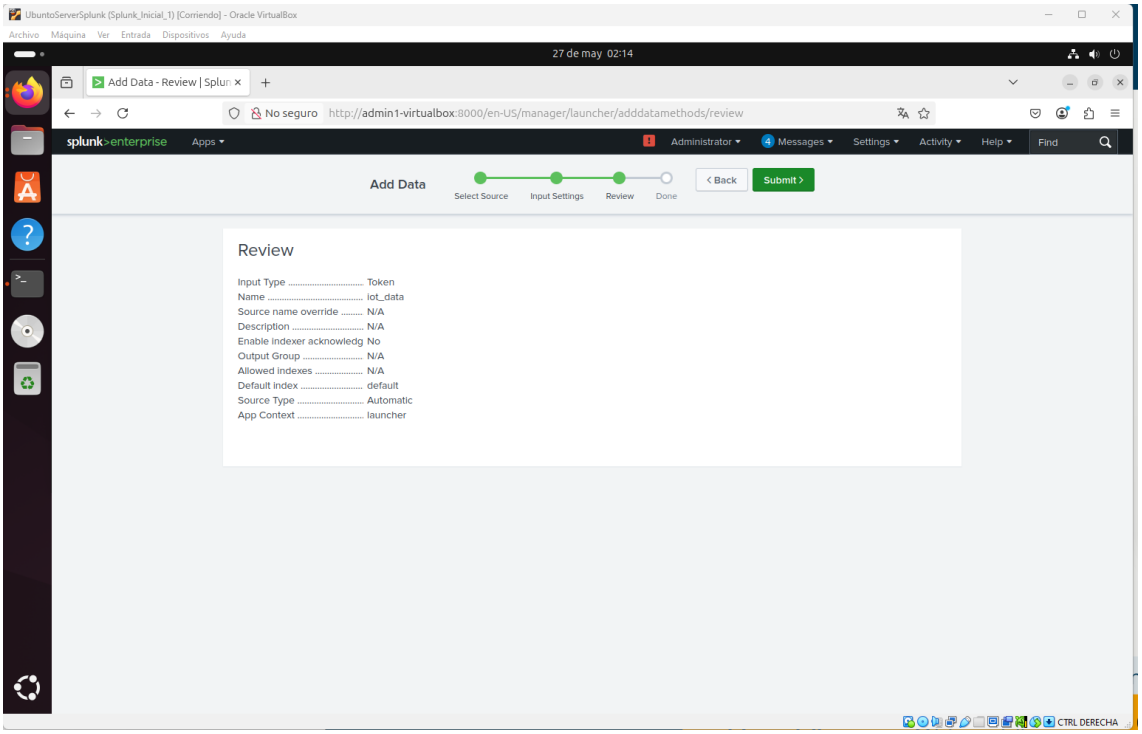
Configuración de insumos para el token



Nota. En la Figura se observa la ventana 3 que es la selección del índice de búsqueda.

Figura 89

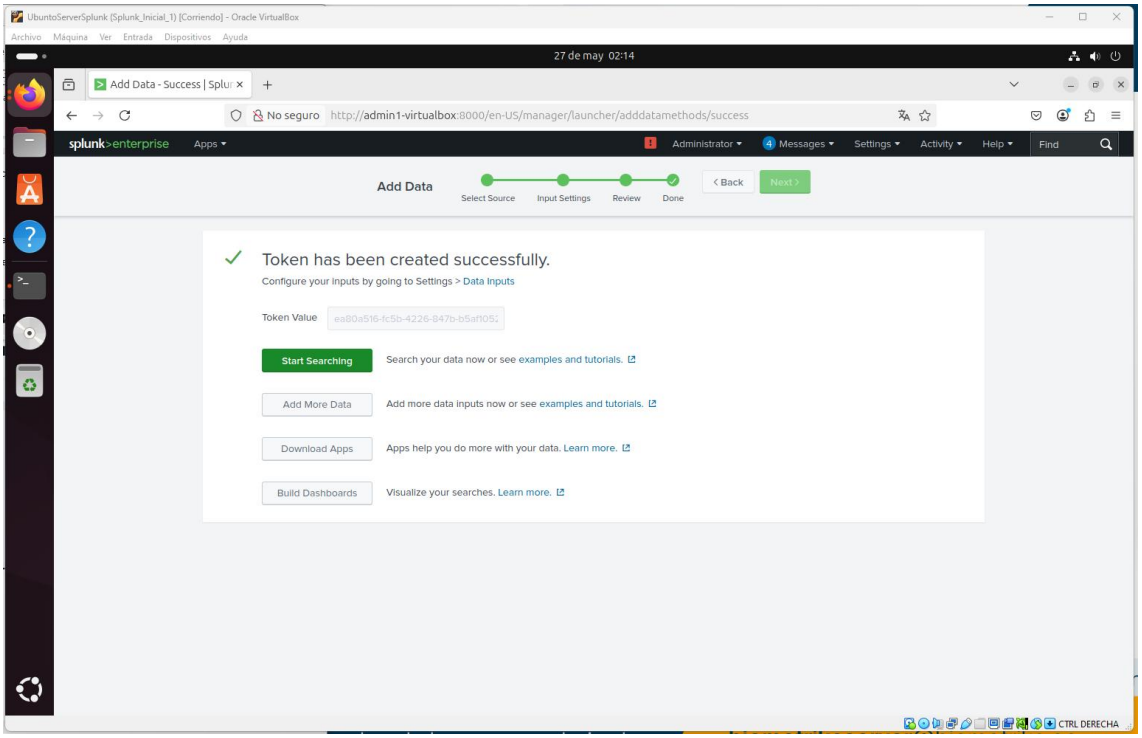
Revisión de la descripción del HEC



Nota. En la Figura se mira la ventana 4 que es la revisión general de la nueva entrada creada.

Figura 90

Creación del Token



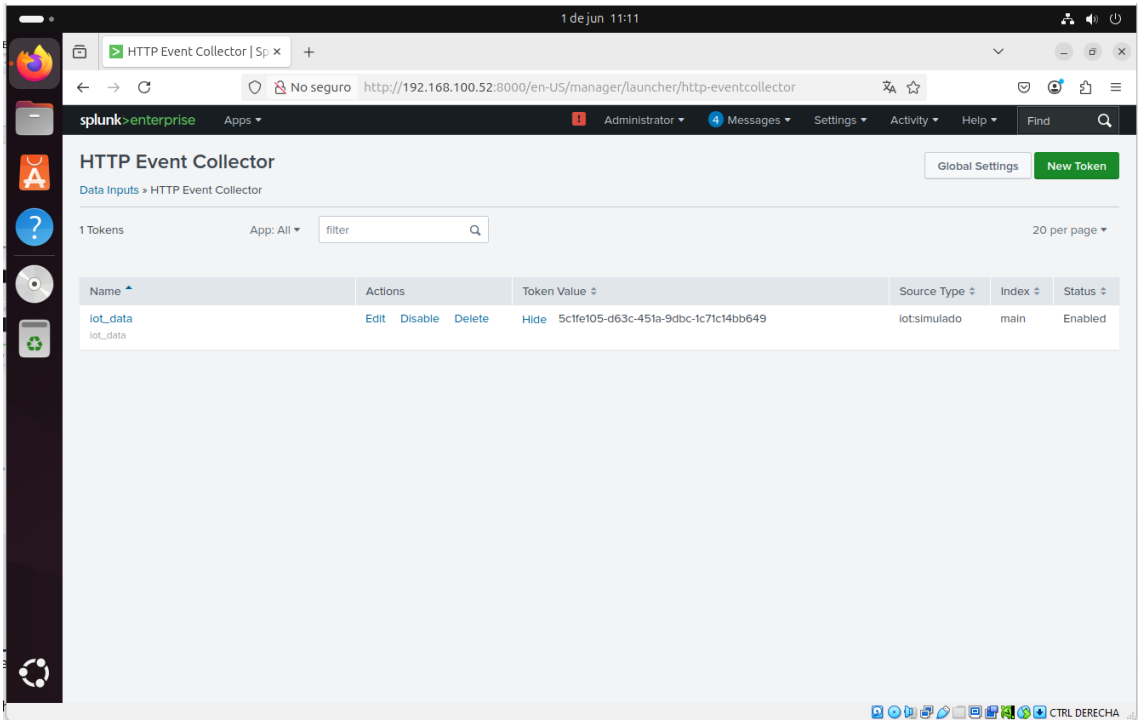
Nota: En la Figura se observa la ventana 5 que es la creación de Token de forma correcta.

Guarda el token generado para usarlo en el *Script Python*.

Validar de que *HEC* esté habilitado y asignado al índice *main*.

Figura 91

Panel del HEC creado



Nota. En la Figura se evidencia la creación de datos de Token.

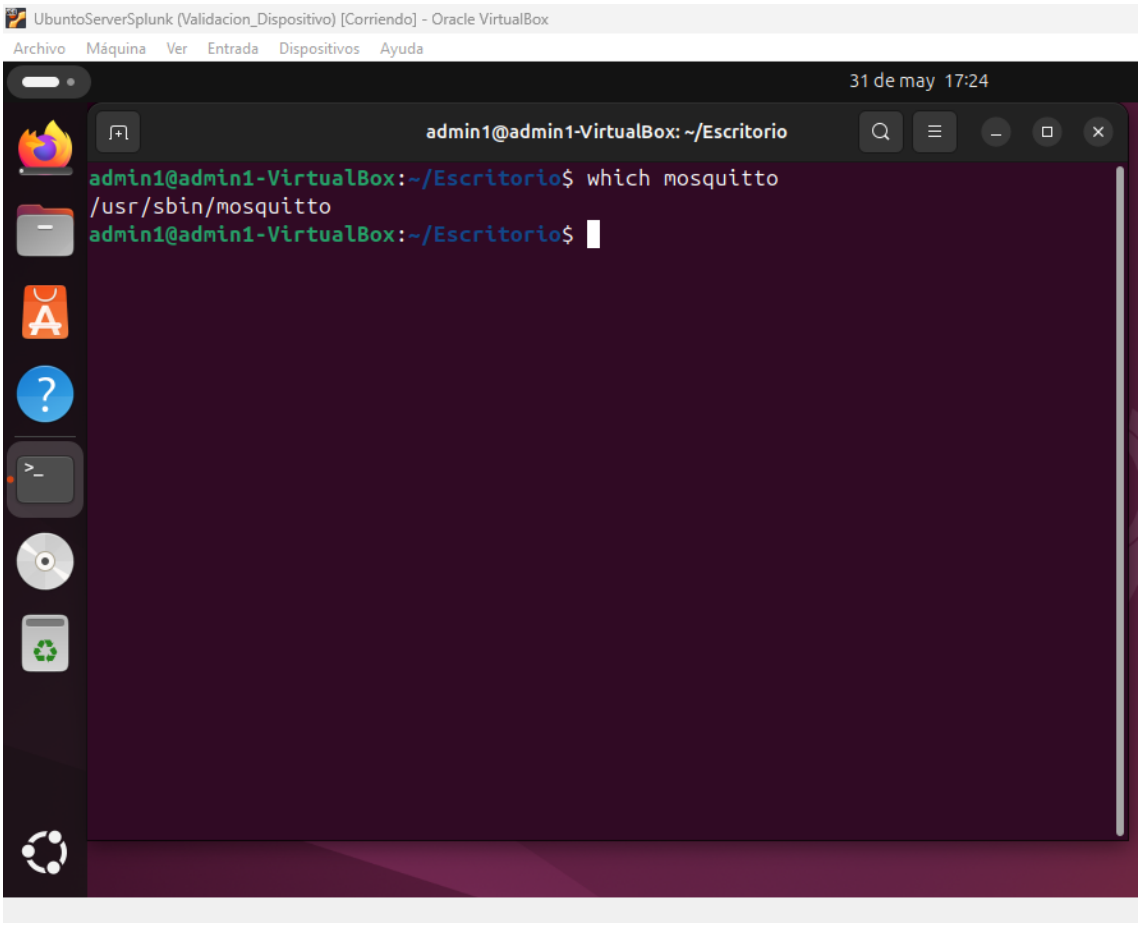
Instalar Mosquitto

Sudo apt update.

Sudo apt install *mosquitto mosquitto-clients*.

Figura 92

Verificación de Mosquitto



Nota. En la Figura se mira la verificación de Mosquitto instalado en el servidor.

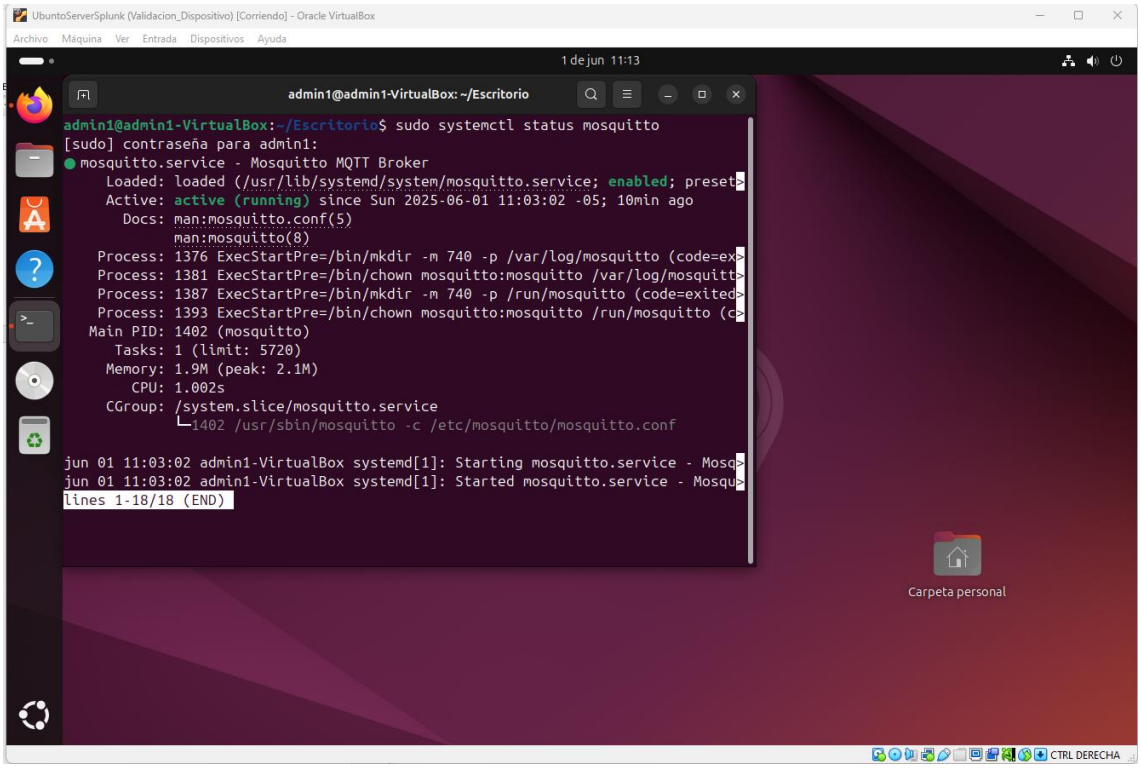
sudo systemctl enable mosquitto

sudo systemctl start mosquito

sudo systemctl status mosquito

Figura 93

Comprobación de funcionamiento del Mosquitto



Nota. En la figura se evidencia la verificación de estado del servicio de Mosquitto.

Configuración en el VM Cliente (100.53)

Preparar entorno virtual Python

> *sudo apt install python3-venv* -y

Instala el módulo necesario (*venv*) para crear entornos virtuales con Python en sistemas basados en Debian/Ubuntu.

> python3 -m venv ~/iot_env

Crea un entorno virtual llamado iot_env en el directorio del usuario. Este entorno funciona como un espacio independiente para gestionar paquetes Python.

> source ~/iot_env/bin/activate

Activa el entorno virtual. A partir de aquí, todos los paquetes que se instalen con pip se alojarán solo dentro de este entorno.

> pip install paho-mqtt requests

Instala las bibliotecas necesarias:

paho-mqtt: Permite conectarse y comunicarse con un broker MQTT.

requests: Permite hacer solicitudes HTTP, necesario para enviar datos a Splunk vía HEC.

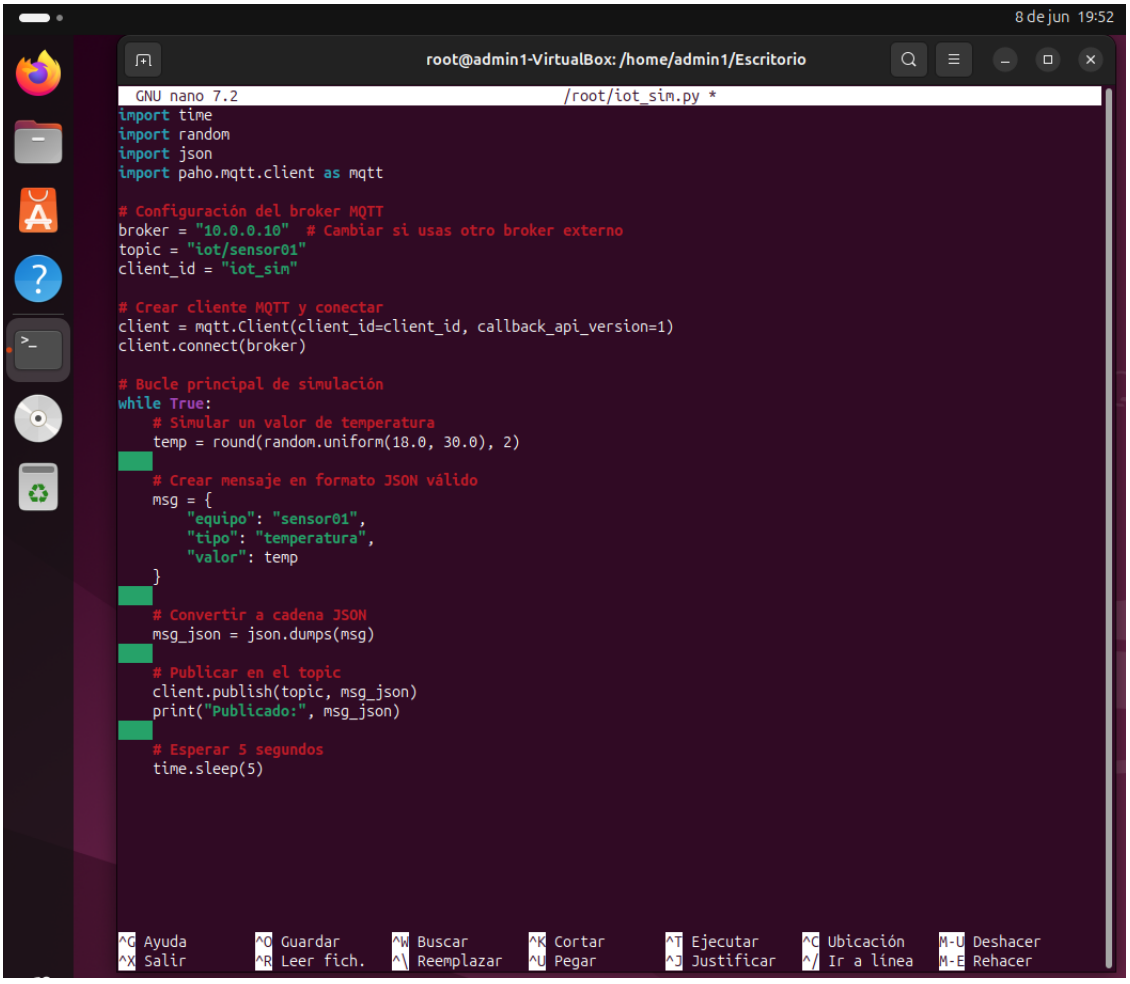
3.1.8. Script de simulación de IoT (iot_sim.py)

Se desarrolló un simulador en Python que genera datos de temperatura cada 5 segundos. Este script utiliza la librería paho-mqtt para publicar los datos al *broker* MQTT en un *topic* específico. Los datos se estructuran en formato JSON, representando el comportamiento de un sensor IoT real. Esta información es posteriormente interceptada por un script puente que reenvía los eventos a *Splunk* mediante el colector HEC.

sudo nano /root/iot_sim.py

Figura 94

Instrucciones de Script

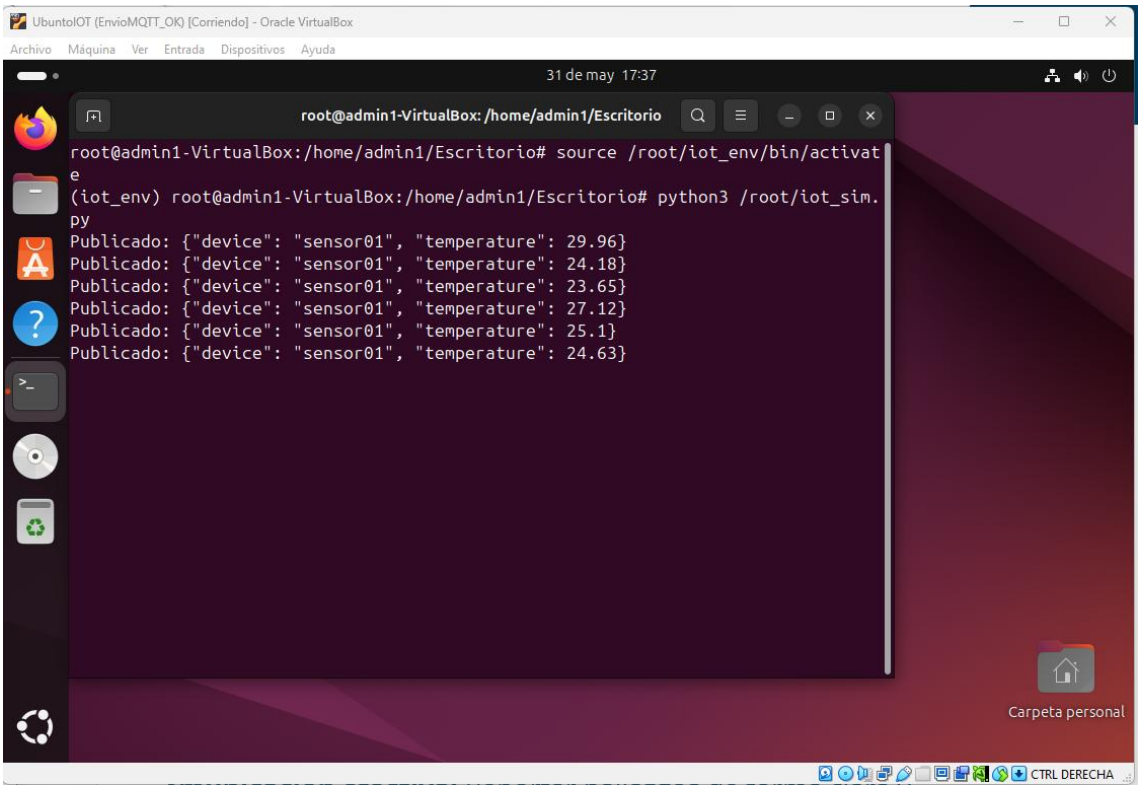


Nota. En la Figura se visualización de las instrucciones de Script iot_sim.py.

Ejecución del Script de simulación de IoT (iot_sim.py)

Figura 95

Visualización de ejecución del simulador IoT



Nota. En la Figura se visualiza la correcta ejecución del script simulador de IoT.

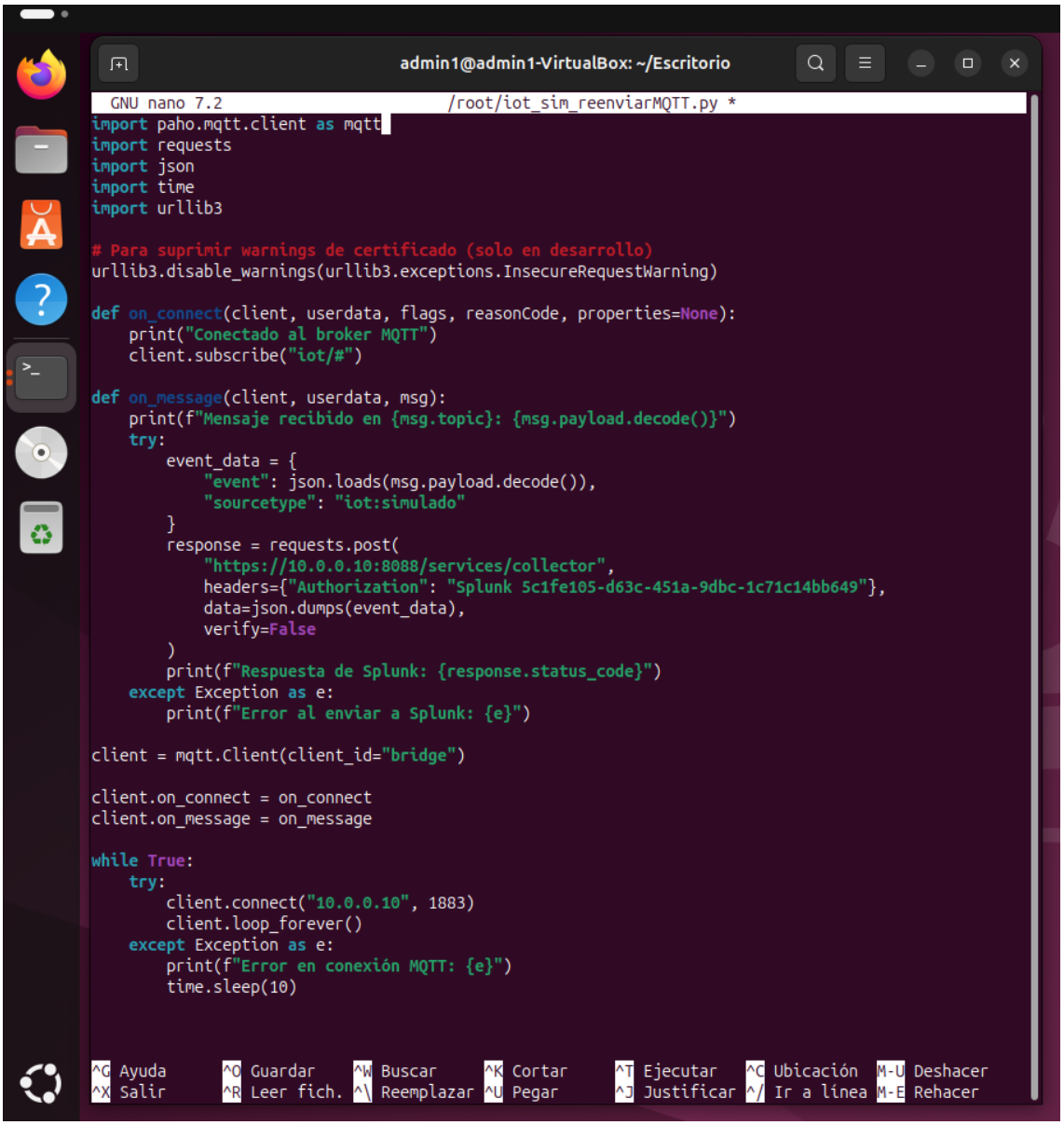
Script puente a Splunk (iot_sim_reenviarMQTT.py)

Este script funciona como un conector entre un *broker* MQTT (Mosquitto) y el sistema de monitoreo *Splunk*. Escucha mensajes en tiempo real publicados en un tópico MQTT (por ejemplo, iot/sensor01), decodifica su contenido (como valores de sensores), y luego los envía como eventos estructurados al HTTP *Event Collector* (HEC) de *Splunk* usando el protocolo HTTP. Esto permite que *Splunk* reciba y visualice los datos generados por dispositivos IoT simulados, facilitando la supervisión, análisis de seguridad y generación de alertas dentro de entornos heterogéneos.

sudo nano /root/iot_sim_reenviarMQTT.py

Figura 96

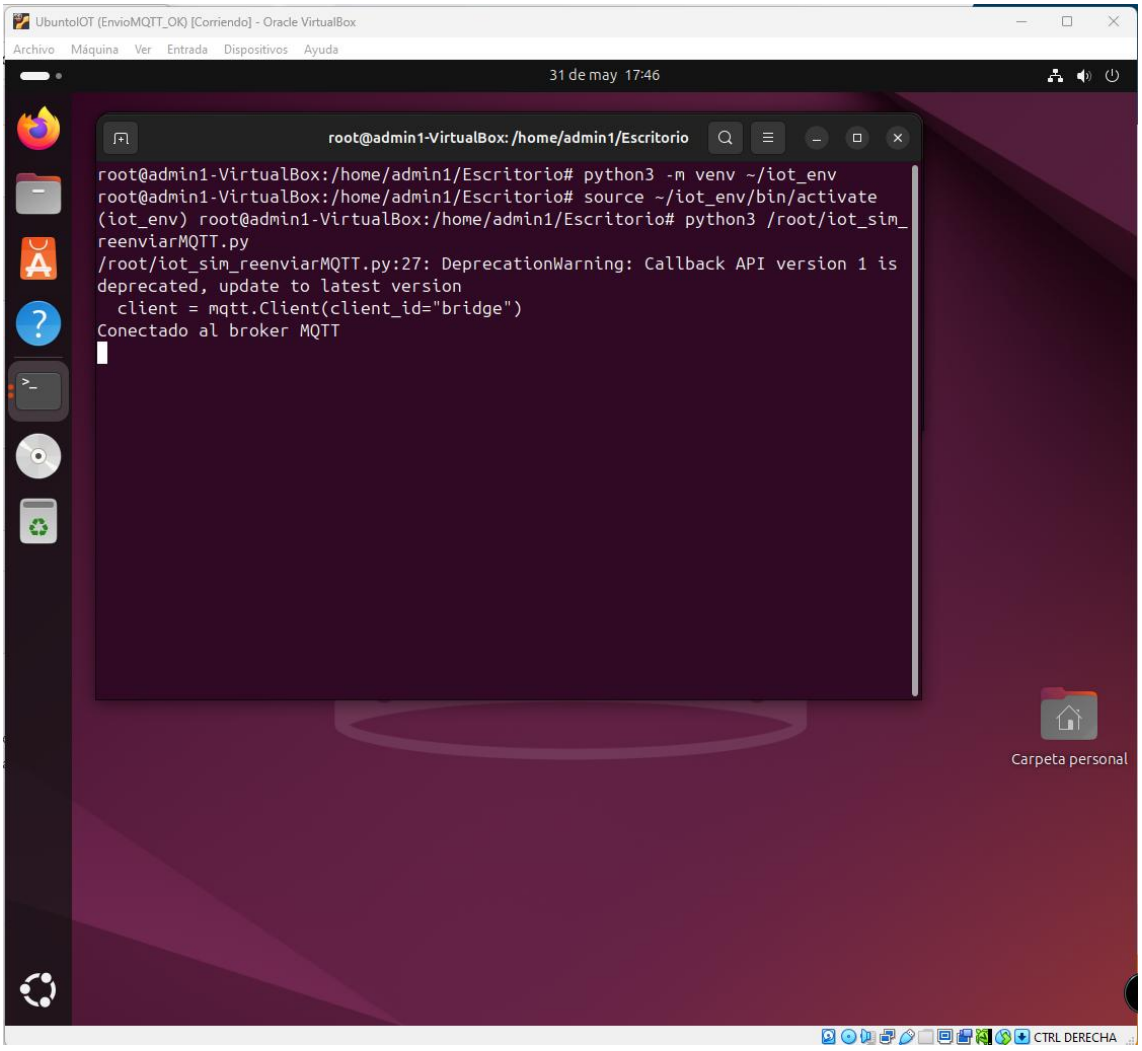
Visualización de las instrucciones de Script



Nota. En la Figura se visualiza las instrucciones de script iot_sim_reenviarMQTT.py

Figura 97

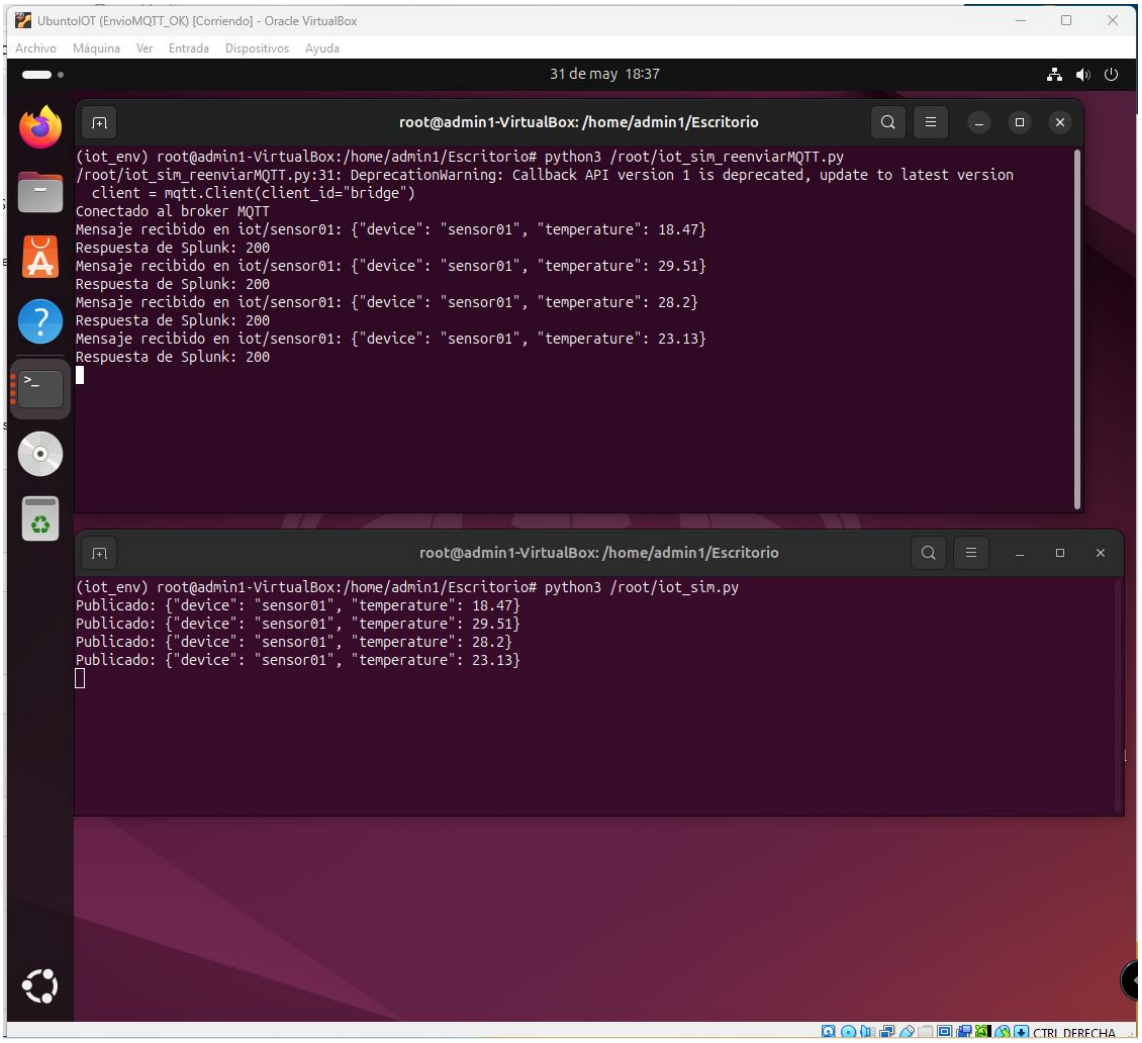
Ejecución de conexión al broker Mosquitto



Nota. En la Figura se evidencia la visualización de ejecución del Conector en línea.

Figura 98

Ejecución en línea simulador y conector



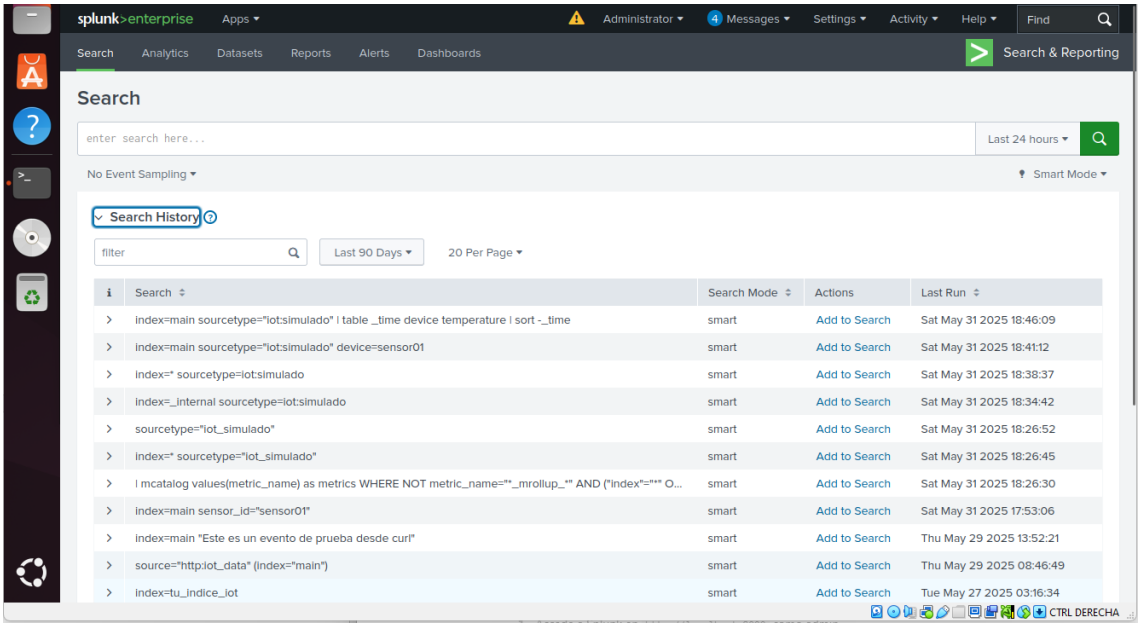
Nota. En la Figura se visualiza la ejecución en línea del simulador y el conector.

Verificación en Splunk

Ingresar a Search & Reporting en Splunk, se visualiza los índices de búsqueda realizados y si es primera vez de la búsqueda se debe ingresar el comando básico: index=main sourcetype="iot:simulado"

Figura 99

Panel de búsqueda general de eventos

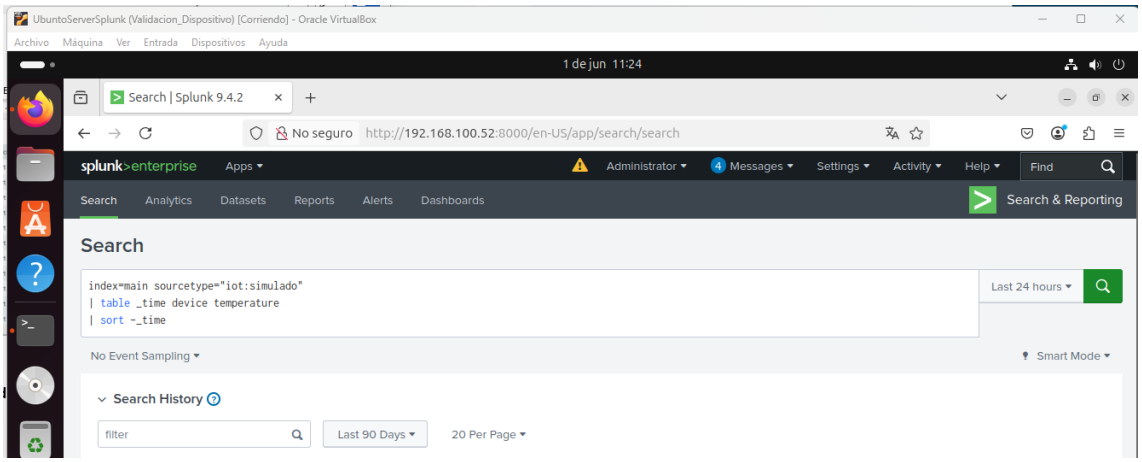


Nota. En la Figura se observa la ventana de búsqueda de eventos.

Si se desea ver con mayor fluidez se crea una vista de datos.

Figura 100

Creación de vista de filtro resumen de IoT simulado

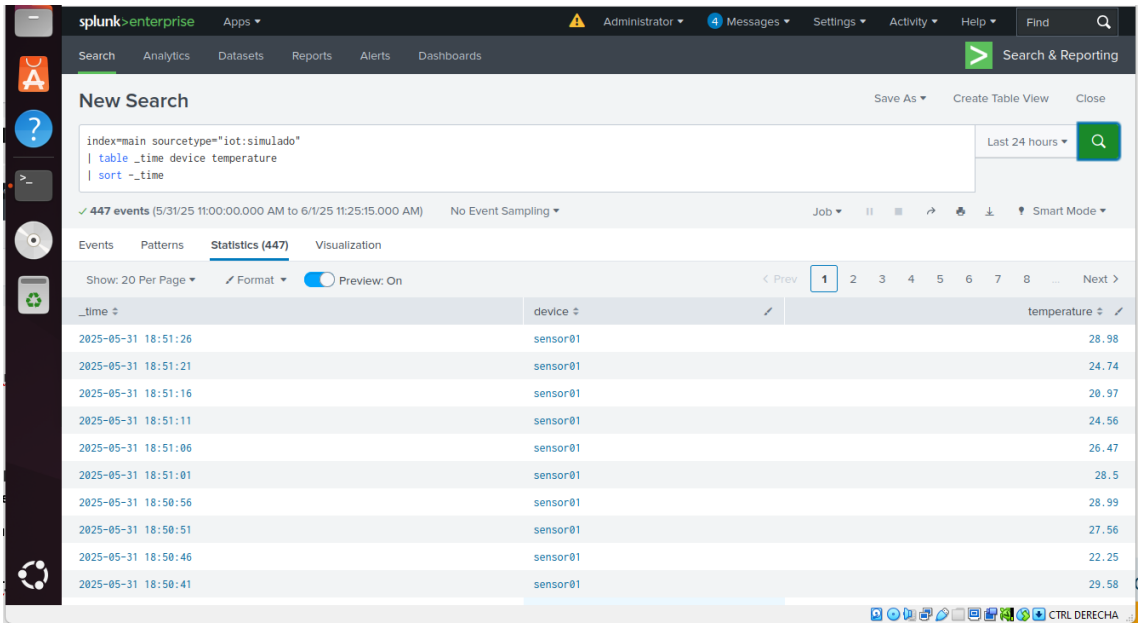


Nota. En la Figura se mira la creación de la vista resumen de los datos de IoT simulado.

Visualización de eventos generados desde la PC cliente, dispositivo IOT

Figura 101

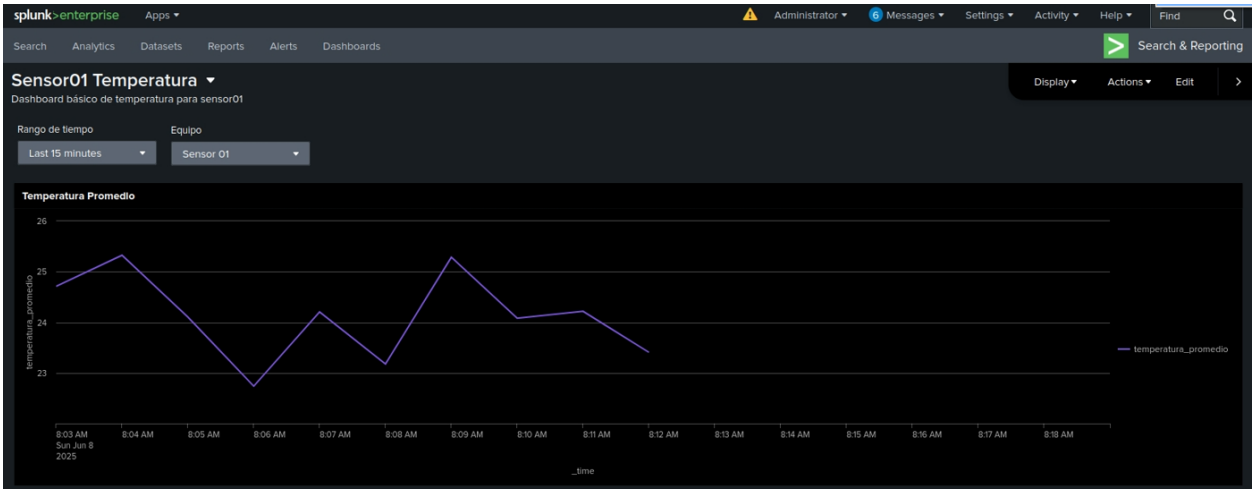
Visualización en eventos del Sensor_01



Nota. En la Figura se evidencia los eventos generados para el Sensor_01 en formato JSON

Figura 102

Dashboard sensor 1



Nota: En el Figura se evidencia las gráficas con resultados de los eventos del sensor 01.

Diagrama de Conexión

>[IoT Simulador]

En esta máquina virtual cliente se ejecuta un script (iot_sim.py) que simula un sensor IoT. Este genera periódicamente datos (como temperatura) y los publica en un tópico MQTT (iot/sensor01).

g> MQTT

El protocolo MQTT se utiliza como canal de comunicación ligero y eficiente para enviar los datos simulados. Es ideal para entornos con recursos limitados y se encarga de transmitir los mensajes desde el simulador al *broker*.

>[Broker Mosquitto]

Actúa como intermediario en el intercambio de mensajes. Recibe los datos publicados por el simulador y los pone a disposición de los suscriptores del tópico correspondiente (iot/sensor01). Está instalado en el servidor que también aloja *Splunk*.

> Script Puente

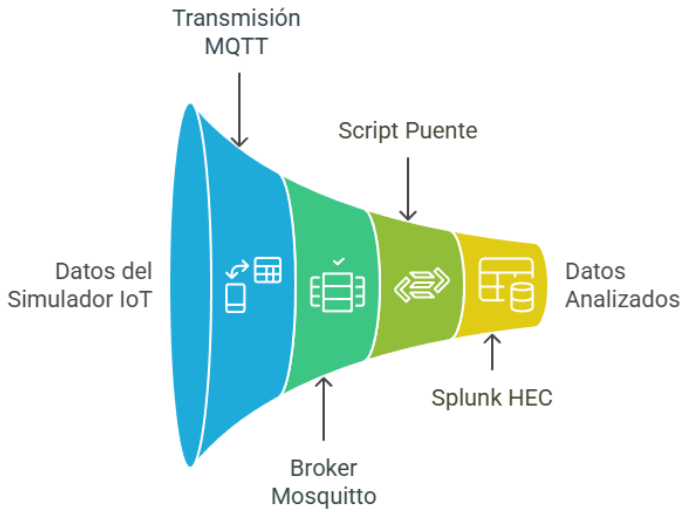
Este script (iot_sim_reenviarMQTT.py), también ejecutado en la máquina cliente, se suscribe al tópico MQTT y recibe todos los mensajes emitidos. Luego convierte cada mensaje en un evento y lo envía mediante una solicitud HTTP POST al HEC de Splunk.

>[Splunk HEC]

El HTTP *Event Collector de Splunk* recibe los eventos enviados por el script puente, los indexa en tiempo real y los pone disponibles para visualización, monitoreo y análisis desde la interfaz de Splunk (Search & Reporting).

Figura 103

Estructura de conexión y flujo de la simulación



Nota. La Figura muestra de forma visual la forma y orden desde el inicio de simulación y todos los componentes que interactúan hasta llegar a los datos centralizados en Splunk