UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

INFORME FINAL DEL TRABAJO DE INTEGRACIÓN CURRICULAR, PROPUESTA TECNOLÓGICA

TEMA:

"IMPLEMENTACIÓN DE SISTEMA DE DETECCIÓN DE ATAQUES PHISHING POR MEDIO DE INTELIGENCIA ARTIFICIAL EN ENTORNOS IOT"

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

Línea de investigación: Innovación Tecnológica y de productos, conectividad e integración de sistema

AUTOR:

Edison Fernando Picuasi Flores

DIRECTOR:

Ing. Fabián Geovanny Cuzme Rodríguez, Msc.

Ibarra, 2025

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003964986		
APELLIDOS Y NOMBRES:	Picuasi Flores Edison Fernando		
DIRECCIÓN:	Peguche, Barrio Tahuantinsuyo Vía Imbabura		
EMAIL:	efpicuasif@utn.edu.ec, efpicuasif@gmail.com		
TELÉFONO FIJO:		**TELF. MOVIL**	0988748184

DATOS DE LA OBRA	
TÍTULO:	Implementación de sistema de detección de ataques phishing por medio de inteligencia artificial en entornos IoT.
AUTOR (ES):	Picuasi Flores Edison Fernando
FECHA: AAAAMMDD	18 de febrero de 2025
SOLO PARA TRABAJOS DE INTEGRACIÓN CURRICULAR	
CARRERA/PROGRAMA:	■ **GRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Telecomunicaciones
DIRECTOR:	Msc. Fabián Geovanny Cuzme Rodríguez
ASESOR:	Msc. Henry Patricio Farinango Endara

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Edison Fernando Picuasi Flores, con cédula de identidad Nro. 1003964986, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de integración curricular descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

Ibarra, a los 18 días del mes de febrero de 2025.

EL AUTOR:

.....

Edison Fernando Picuasi Flores

CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 18 días, del mes de febrero de 2025

EL AUTOR:

.....

Edison Fernando Picuasi Flores

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 18 de febrero de 2025

MSC. FABIÁN GEOVANNY CUZME RODRÍGUEZ

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

(f) .....

Msc. Fabián Geovanny Cuzme Rodríguez.
C.C.: 1311527012

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificado del trabajo de Integración Curricular "IMPLEMENTACIÓN DE SISTEMA DE DETECCIÓN DE ATAQUES PHISHING POR MEDIO DE INTELIGENCIA ARTIFICIAL EN ENTORNOS IOT" elaborado por EDISON FERNANDO PICUASI FLORES, previo a la obtención del título de Ingeniero en Telecomunicaciones, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

(f):.....
Msc. Fabián Geóvanny Cuzme Rodríguez
C.C.: 1311527012

(f):.....
Msc. Henry Patricio Farinango Endara
C.C.: 1003423710

DEDICATORIA

A mis queridos padres, José Picuasi y María Flores, por su amor, paciencia y sacrificios realizados, me han dado la oportunidad de alcanzar este logro.

A mis hermanas, Gina y Saida, por su apoyo en la búsqueda de nuevas aspiraciones y motivación para no rendirse.

Edison Fernando Picuasi Flores

AGRADECIMIENTO

Expreso, un profundo agradecimiento a mis padres, José Picuasi y María Flores por la paciencia, el sacrifico, la confianza, y el apoyo durante toda esta etapa.

A mis hermanas, Gina y Saida, por su apoyo constante y la confianza depositada en mí para cumplir mis metas.

Al Ing. Fabián Cuzme, director de mi trabajo de titulación y al Ing. Henry Farinango, asesor, por su tiempo, apoyo y guía en la realización de este trabajo.

Finalmente, a todos mis amigos con quienes compartí aulas y experiencias durante mi vida universitaria.

Edison Fernando Picuasi Flores

RESUMEN EJECUTIVO

El presente trabajo de titulación aborda la creciente problemática que representan los ataques phishing en la seguridad informática de entorno IoT. El objetivo general se centra en implementar un sistema de detección de ataques phishing que opere en conjunto con un sistema de detección de intrusos (IDS) y utilice un modelo de inteligencia artificial basado en procesamiento de lenguaje natural. El desarrollo sigue los lineamientos de la metodología en cascada, y se estructura en 4 fases principales. En la primera fase se determinan los requerimientos del sistema haciendo uso del estándar ISO/IEC/IEEE 29148:2018. En la siguiente fase correspondiente al de diseño, se establece la arquitectura del sistema con un enfoque hacia el Internet de las Cosas, definiendo tanto la red IoT, así como del modelo de detección de phishing. En la fase de implementación, se despliegan los servicios necesarios para la comunicación por medio de CoAP, se enlazan los servicios de almacenamiento y monitorización de datos, además de que se realiza el entrenamiento del modelo para su posterior integración en la red simulada. La fase final se centra en la evaluación del sistema en general, por medio de un plan de pruebas. En base al plan se evalúa la funcionalidad de los segmentos y procesos que forman parte de la red IoT, así como de las funciones asociadas al análisis de phishing en correos electrónicos. Los resultados obtenidos demostraron que la integración de técnicas de inteligencia artificial con sistemas de detección de intrusos mejora significativamente la capacidad de detección de amenazas, permitiendo una detección más precisa de elementos relacionados con el phishing. En conclusión, la solución propuesta muestra evidencias de mejora de la seguridad en redes IoT, además que agregan una capa extra de seguridad en las interacciones de los usuarios. Si bien se recomienda su evaluación en entornos reales y la adición de más parámetros de análisis al modelo, en general se logra

establecer un modelo operativo que permite afrontar ataques phishing por medio de correos electrónicos.

Palabras clave: Detección de phishing, Internet de las cosas (IoT), Sistemas de Detección de Intrusos (IDS), Procesamiento de Lenguaje Natural (PLN), Seguridad Redes, Ciberseguridad, Arquitectura IoT, Mininet-WiFi.

ABSTRACT

This thesis addresses the growing problem of phishing attacks in computer security within IoT environments. The general objective focuses on implementing a phishing attack detection system that operates in conjunction with an intrusion detection system (IDS) and uses an artificial intelligence model based on natural language processing. The development follows the waterfall methodology guidelines and is structured in 4 main phases. In the first phase, the system requirements are determined using the ISO/IEC/IEEE 29148:2018 standard. In the next phase corresponding to design, the system architecture is established with a focus on the Internet of Things, defining both the IoT network and the phishing detection model. In the implementation phase, the necessary services for communication through CoAP are deployed, storage and data monitoring services are linked, and the model is trained for its subsequent integration into the simulated network. The final phase focuses on the evaluation of the overall system through a test plan. Based on the plan, the functionality of the segments and processes that are part of the IoT network is evaluated, as well as the functions associated with phishing analysis in emails. The results obtained demonstrated that the integration of artificial intelligence techniques with intrusion detection systems significantly improves the capability to detect threats, allowing for more precise detection of phishing-related elements. In conclusion, the proposed solution shows evidence of improved security in IoT networks, while also adding an extra layer of security in user interactions. Although its evaluation in real environments and the addition of more analysis parameters to the model is recommended, in general, an operational model is established that allows addressing phishing attacks through emails.

Keywords: Phishing Detection, Internet of Things (IoT), Intrusion Detection System (IDS), Natural Language Processing (NLP), Network Security, Cybersecurity, IoT Architecture, Mininet-WiFi.

LISTA DE SIGLAS

NLP. Natural Language Processing

IDS. Sistema de Detección de Intrusos

CoAP. Constrained Application Protocol

IoT. Internet de las Cosas

IA. Inteligencia Artificial

GenAI. Inteligencia Artificial Generativa

DNS. Sistema de nombres de dominio

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I. Antecedentes

En este capítulo se presentan los antecedentes, en el que se da una breve introducción acerca del proyecto de titulación. A continuación, se presenta el tema, la problemática, los objetivos planteados, tanto general como específicos, el alcance y la justificación que fundamenta este proyecto.

1.1 Problema de investigación.

La creciente presencia de los dispositivos de Internet de las cosas (IoT) han mejorado las formas de interacción con el entorno a la vez que se ha convertido en una tecnología de cambio a nivel industrial. Pese a los inherentes beneficios que brinda los dispositivos IoT, también se han sumado el incremento en los índices de amenazas de seguridad y privacidad producto de la gran diversidad de entornos y la falta de estándares (Kim, 2017), sumado a esto la existencia de un amplio tipo de ataques que operan a nivel físico, de red, software y encriptación (Andrea et al., 2015) dificultan los procesos de detección y mitigación de ataques.

En el entorno IoT los ataques como el phishing representan un gran impacto debido a que ponen en riesgo la credibilidad e imagen de la marca, así como en un marco económico estos incurren en pérdidas, a su vez del lado de la víctima generan desconfianza en los métodos de transacción que se reflejan en los índices de adopción de tecnología orientada al comercio (Srivastava & Purcell, 2007).

La inteligencia artificial es una rama de las ciencias computacionales que hacen referencias a los sistemas o maquinas que cuentan con características de inteligencia humana y ejecutan tareas en base a reglas o algoritmos (Jakhar & Kaur, 2020). Una disciplina de la inteligencia artificial de gran impacto en el análisis de información es el aprendizaje de maquina o machine learning la cual recientemente se encamina como una

solución prometedora en el desarrollo de nuevas IDS debido a su capacidad para descubrir, examinar y extraer patrones de datos para la creación de modelos de predicción más precisos (Yang & Shami, 2022).

Para abordar los problemas de seguridad en relación con los ataques de phishing en entornos IoT se propone un sistema que incorpore características de inteligencia artificial, así como otras tecnologías emergentes centradas en el análisis de tráfico y datos generados en la red, con el fin de mejorar la eficiencia en los procesos de detección de ataques.

1.2 Justificación

El crecimiento constante en la adopción de dispositivos IoT, los cuales según índices presentados por (Dahlqvist et al., 2019) para 2023 se prevé un total de 43 millones de dispositivos conectados que en comparación a 2018 es 3 veces mayor, dichas cifras representan un aspecto a considerar ya que pese al crecimiento en el número de dispositivos el aspecto de seguridad no experimentado el mismo desarrollo.

La globalización digital ha creado nuevas formas ilícitas de apropiación de datos personales que no se centran solo en métodos de estafas tradicionales, sino que incorpora herramientas como buscadores de internet y servicios basados en contenido (Aguilar Martínez et al., 2023). Según (Gupta et al., 2018) el phishing es el ataque número uno en materia de robo de información a nivel empresarial, este tipo de ataque estuvo involucrado en el 32% de robo de datos del año 2019 (Abbas et al., 2021). La presencia de ataques más sofisticados en torno al IoT requieren el uso de nuevas tecnologías para solventar los riesgos de seguridad (Andrea et al., 2015).

Bajo el contexto anterior la seguridad de los datos personales se vuelve esencial en cualquier sistema y a su vez este se convierte en un requerimiento necesario en

cualquier clase de servicio, la Ley Orgánica De Protección De Datos Personales (Ley Orgánica de Protección de Datos Personales, 2021) en su artículo 37 establece que “El responsable o encargado del tratamiento de datos personales según sea el caso, deberá sujetarse al principio de seguridad de datos personales, para lo cual deberá tomar en cuenta las categorías y volumen de datos personales, el estado de la técnica, mejores prácticas de seguridad integral y los costos de aplicación de acuerdo a la naturaleza, alcance, contexto y los fines del tratamiento, así como identificar la probabilidad de riesgos.” De esta manera la normativa exige la implementación de mecanismos que salvaguarden la información personal, a su vez que se debe evidenciar que las medidas adoptadas e implementadas mitiguen los riesgos identificados.

Otro punto de clave es el nivel de seguridad con el que cuentan los dispositivos IoT, un nivel aceptable de seguridad para un objeto inteligente IoT gira en torno a varias consideraciones que son: Diseño, Conciencia sobre la seguridad, Seguridad en transporte, Seguridad en el almacenamiento, seguridad en las plataformas de software, Seguridad en las plataformas de hardware, Seguridad funcional, Autenticación, Auditabilidad, Registro forense, Gestión y Monitoreo, Sostenibilidad y Capacidad de actualización (Cuzme Rodríguez, 2015), dichos aspectos no solo se centran en la parte física si no que se extiende a diversas áreas denotando que el grado de seguridad va en función de que aspectos abarca o satisface el objeto inteligente.

Ningún tipo de red está exenta de vulnerabilidades ya sea a nivel de diseño, arquitectura, protocolos o tecnología, en el caso de IoT dichas vulnerabilidades se exponen de manera gradual a medida que el número de dispositivos aumenta. Uno de los métodos para afrontar el análisis del gran volumen de datos generado en las redes IoT es el uso de la inteligencia artificial, que, a su vez junto con otras técnicas, tienen la capacidad de mitigar (Wu et al., 2020).

Así es como, el presente proyecto propone el diseño de un sistema de detección de ataques enfocados en phishing basado en procesamiento de lenguaje natural, la cual se implementará por medio de Mininet para posteriormente realizar la verificación del sistema a través de un data set de evaluación de detección de intrusos (CIC-IDS2017) y URLs maliciosas (ISCX-URL2016).

1.3 Objetivos

1.3.1 Objetivo General

Implementar un sistema de detección ataques de phishing aplicando IDS y modelos de inteligencia artificial basado en procesamiento de lenguaje natural.

1.3.2 Objetivos Específicos

- Investigar y comparar las características, así como los porcentajes de efectividad de los mecanismos de seguridad típicos en relación con los mecanismos que empleen inteligencia artificial.
- Implementar un IDS que opere en conjunto con un modelo de procesamiento de lenguaje natural (NLP).
- Realizar pruebas de funcionamiento del sistema de detección de ataques phishing basado en procesamiento de lenguaje natural.

1.4 Alcance

El presente proyecto propone cubrir el área de la seguridad de la información a través de un sistema que emplee métodos de inteligencia artificial, al igual que tecnologías emergentes como el procesamiento de lenguaje natural, para la detección de ataques phishing en entornos IoT. La recopilación de información resultante de la investigación encaminará el proceso de desarrollo del sistema, cuya finalidad es ampliar el grado de seguridad en un entorno IoT.

Para el proyecto se toma a consideración el tipo de investigación aplicada que, sumado al modelo de cascada, el cual se basa en una secuencia de etapas interconectadas, permitirá desarrollar el sistema propuesta de manera secuencial y ordenada. Las etapas consisten en definición de los requisitos, diseño e implementación, integración y pruebas del sistema, operación y mantenimiento.

En primera instancia se realizará una investigación en torno a los diferentes tipos de ataques que se suscitan en las redes IoT, con el objetivo de comprender y recopilar datos sobre qué áreas o en que segmentos específicos se producen dichos ataques, además de un estado del arte que denote la relación entre la inteligencia artificial y procesamiento de lenguaje natural, con los sistemas de detección de ataques phishing en entornos IoT, para de esta manera determinar los requerimientos para el diseño del sistema.

La segunda etapa se centra en el diseño del modelo con base en la investigación que cumpla con los requerimientos necesarios de un modelo de procesamiento de lenguaje natural, el cual debe deber ser capaz de aplicar los procesos de limpieza, normalización de datos y tokenización, en referencia a los ataques phishing.

La etapa de implementación del sistema consiste en incorporar el modelo generado a una simulación realizada de Mininet, que contenga las características de una red IoT, en la cual se realizará actividades como el preprocesamiento de datos, la extracción de características, la detección de anomalías y la clasificación del ataque.

En la etapa final se realizan las pruebas de validación del sistema, para la cual se pretende realizar ataques simulados mediante herramientas de software y datasets, de esta forma se podrá validar que el modelo desarrollado es capaz de detectar el ataque de phishing.

CAPÍTULO II. Marco Teórico

En el segundo capítulo, se abordan las definiciones clave relacionadas con el Internet de las Cosas (IoT), phishing, inteligencia artificial y procesamiento de lenguaje natural (NLP). Se examinan a detalle los principios de IoT, la naturaleza del phishing, las aplicaciones de la inteligencia artificial y el papel del PLN en la interacción entre humanos y máquinas. Este capítulo establece las bases conceptuales esenciales para comprender el contexto y los elementos fundamentales de la investigación.

2.1 Seguridad Informática

Según la Organización Internacional de Normalización (ISO, 2018), la seguridad de la información se define como el conjunto de procesos, métodos y procedimientos orientados a preservar la información, así como los sistemas de información de acciones en torno al uso, difusión, modificación o supresión no autorizado. La seguridad de la información contempla la conservación de las 3 propiedades de la información; confidencialidad, integridad y disponibilidad.

La confidencialidad se refiere a la protección y restricción del acceso a la información solo a aquellos usuarios autorizados. La información confidencial no debe estar disponible o revelada a personas no autorizadas, ya sea de manera intencional o accidental.

La integridad se centra en la precisión y la fiabilidad de la información. Garantiza que la información no ha sido alterada de manera no autorizada o accidental. En un contexto de seguridad de la información, la integridad asegura que los datos mantengan su exactitud y consistencia durante todo su ciclo de vida.

La disponibilidad se refiere a la accesibilidad y utilización de la información cuando sea necesario por parte de usuarios autorizados. Implica que los sistemas y

servicios estén operativos y accesibles, minimizando el tiempo de inactividad y asegurando que los recursos estén disponibles para cumplir con los requisitos operativos.

2.2 Phishing

Es un método de ingeniería social en la que un atacante conocido como phishier emplea diversas estrategias que emulen comunicaciones electrónicas de una organización (Alkhalil et al., 2021) con el objetivo de persuadir al objetivo del ataque para que revele información personal o confidencial tales como dirección email, nombre de usuario, contraseñas e información financiera (Alabdan, 2020). El ataque phishing consta de varias etapas en las que el ciberdelincuente sigue para llevar a cabo el engaño. La Figura 1 muestra el ciclo de vida del ataque Phishing que consta de 5 fases principales:

Planificación

Es un proceso en el cual se delinea el objetivo ya sea un individuo u organización y la información de interés (Alkhalil et al., 2021). Primordialmente, esta fase consiste en extraer información esencial de la víctima por medio de análisis de tráfico. Posterior al reconocimiento se establecen herramientas y técnicas que se ejecutaran en el ataque a través de medios viables tales como sitios web fraudulentos, emails que contengan enlaces maliciosos, etc (Benavides et al., 2020).

Phishing

En esta fase se despliega la actividad del phishing dirigida a los objetivos previamente identificados, los atacantes implementan tácticas engañosas al enviar emails adulterados disfrazados como comunicaciones legítimas proveniente de instituciones y entidades reales, que solicitan información confidencial bajo la premisa de una actualización de registros. La urgencia de la actualización de la información se enfatiza, instando a la víctima a acceder a los enlaces maliciosos presentes en el correo (Benavides et al., 2020).

Infiltración

Varía en función del método empleado, pero esencialmente en esta fase se define por la respuesta del objetivo (Alkhalil et al., 2021) e inicia al momento que accede al enlace malicioso, este simple acto desencadena la instalación automática de malware y a su vez brinda al atacante la capacidad de acceder al sistema y comprometerlo. Dependiendo del caso el acceso al enlace puede redireccionar al usuario a un sitio web falso (Benavides et al., 2020).

Recopilación de datos

En esta fase el phisher despliega diversas estrategias para obtener la información requerida y alcanzar los objetivos planteados en la fase de planificación (Alkhalil et al., 2021). Durante el proceso de recopilación de información el atacante simula la identidad de la víctima de esta manera se accede a sus cuentas y al sistema por lo que eventualmente puede repercutir en pérdidas financieras. Los sistemas comprometidos abren puertas a su uso en ataques más amplios, como son los de denegación de servicios (DOS) (Benavides et al., 2020).

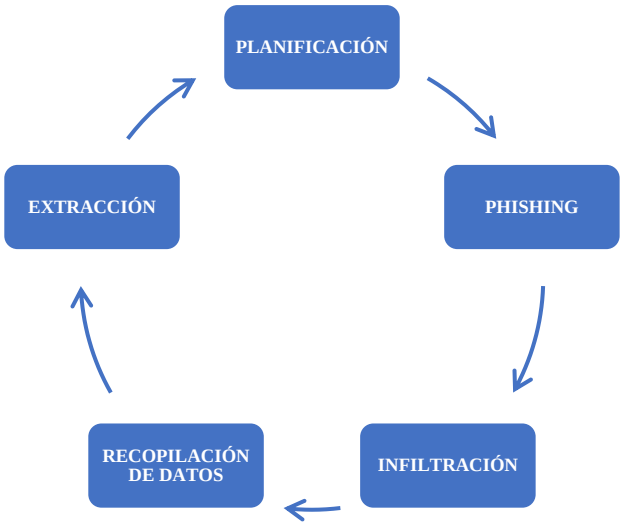
Extracción

Es la última fase del ataque, se ejecuta posterior a la obtención del acceso y la información requerida (Benavides et al., 2020). En ella el phisher intenta eliminar de manera meticulosa cualquier rastro del ataque tales como sitios web falsos con el fin de eludir la detección y preservar el anonimato (Alkhalil et al., 2021). Finalmente, se realiza un análisis del éxito alcanzado durante el ataque para ser aplicado en futuros ataques.

La Figura 1 engloba los procesos correspondientes al ciclo de vida del ataque phishing.

Figura 1

Ciclo de Vida del Phishing



Fuente: Autoría.

2.2.1 Tipos y Técnicas de Ataques Phishing

Los phishers ejecutan sus ataques a través de la manipulación psicológica y métodos técnicos, con el propósito de obtener información personal. Generalmente, se emplean aquellos ataques que se centran en el engaño, para de esta manera aprovechar la psicología humana y obtener información relevante (Alkhalil et al., 2021). La Figura 2 muestra los diversos tipos y técnicas en las que se divide el ataque phishing.

2.2.1.1 Phishing Engañoso

Es el tipo de ataque de más habitual, en este enfoque, el phisher inventa situaciones o aplica métodos técnicos para atraer y persuadir a la víctima sobre la autenticidad del correo electrónico fraudulento. Al caer en estas artimañas, el usuario se convierte en víctima al acceder al enlace proporcionado, revelando así su información personal al phisher. Este tipo de phishing se lleva a cabo a través de correos electrónicos fraudulentos, sitios web falsos, llamadas telefónicas de phishing, plataformas de redes sociales y diversos otros medios (Alkhalil et al., 2021).

Phishing por Email

Consiste en enviar correos electrónicos fraudulentos de manera aleatoria desde una fuente no confiable a miles de destinatarios. Estos emails falsos pretenden ser de una persona o entidad financiera, con el objetivo de persuadirlos para que realicen acciones que conduzcan a la revelación de información confidencial. Existen tres variantes de phishing por correo electrónico: spear phishing, whale phishing y clone phishing, cada una adaptada según el objetivo del ataque (Alkhalil et al., 2021).

- **Spear Phishing:** Busca obtener información detallada de la víctima para crear correos electrónicos maliciosos que parezcan provenir de fuentes fiables.
- **Whale Phishing:** Emplea tácticas sofisticadas para engañar a personas de alto nivel con el objetivo de obtener datos confidenciales.
- **Clone Phishing:** Replica correos electrónicos legítimos haciendo uso de información del destinatario, como direcciones de correo reales que contiene enlaces modificados o archivos adjuntos maliciosos.

Sitio Web Falsificado

En este tipo de ataque en donde los phishers crean una página web que aparenta ser auténtica y se asemeja al sitio web legítimo. Los estafadores dirigen a usuarios desprevenidos hacia estos sitios por medio de enlaces incluidos en emails, anuncios o cualquier otra forma de redirección, de este modo buscan revelar y recopilar información confidencial (Alkhalil et al., 2021).

Phishing Telefónico

Es una forma de ataque en donde el phisher se comunica con la víctima por medio de llamadas telefónicas o SMS. Durante la comunicación el atacante adopta la identidad de una fuente conocida y confiable con la que la víctima trata. El ataque consiste en el envío de alertas de seguridad convincentes de fuentes tales como bancos, y de esta manera persuadir a la víctima de que se comunique a un número en específico, con el objetivo de

que este comparta información de identificación personal (PII) o números de PIN. El phishing telefónico puede hacer uso de técnicas como la suplantación de identidad de llamadas (CID), donde el atacante engaña a la víctima sobre la fuente de la llamada, o aprovechar herramientas de (IP PBX) para potenciar los ataques de voz por IP (VoIP) (Alkhalil et al., 2021).

Social Media Attack (Soshing, Social Media Phishing)

Este tipo de ataque utiliza plataformas de redes sociales como medio principal para llevar a cabo las acciones maliciosas. Abarca diversas amenazas, como el secuestro de cuentas, la suplantación de identidad, estafas y la distribución de malware. La detección y mitigación de estas amenazas requieren más tiempo en comparación con los métodos tradicionales, debido a que estas plataformas operan fuera de la red convencional.

2.2.1.2 Subterfugio Técnico

Implica engañar a individuos para que divulguen información confidencial mediante artimañas técnicas, como la descarga de código malicioso en el sistema de la víctima (Alkhalil et al., 2021). Los subterfugios técnicos pueden categorizarse en los siguientes tipos:

Phishing basado en Malware

Este método de ataque implica la ejecución de software malicioso en el computador de la víctima, con el objetivo de llevar a cabo actividades fraudulentas o robo de información confidencial. El malware se descarga a través de técnicas de ingeniería social o aprovechando vulnerabilidades en el sistema de seguridad, como la presente en los navegadores web (Alkhalil et al., 2021). Existen varias formas de phishing basado en malware, entre ellas se destacan las siguientes:

- Registradores de teclas y de pantalla (Keyloggers).
- Virus y Gusanos.
- Spyware
- Adware
- Ransomware
- Rootkits
- Secuestradores de sesión
- Troyanos web
- Envenenamiento por archivos de hosts
- Ataque de reconfiguración del sistema
- Robo de datos

Phishing basado en DNS (Pharming)

Denominado también como Pharming es un ataque que manipula el DNS para redirigir a las víctimas a sitios web maliciosos, contaminando la cache DNS con información incorrecta. Esta técnica incluye la alteración del servidor y del archivo de hosts, al comprometer el servidor se modifican las direcciones IP legítimas por otras en donde llevan al usuario a sitios falsos de manera involuntaria (Alkhalil et al., 2021).

Phishing por Inyección de Contenido

Esta práctica consiste en insertar información falsa en un sitio web legítimo, el contenido malicioso tiene el potencial de redirigir a la víctima hacia sitios webs fraudulentos con el objetivo de inducir al usuario a revelar información confidencial o provocar la descarga de malware en el dispositivo (Alkhalil et al., 2021).

Phishing de Hombre en el Medio

En esta forma de phishing, el atacante interviene la comunicación entre el usuario y el sitio web legítimo con el objetivo de obtener información de ambas partes. En este tipo de ataque, el mensaje original destinado a los usuarios legítimos es desviado hacia el

atacante en lugar de llegar al destinatario previsto, a través del uso de diversas técnicas como envenenamiento ARP, suplantación de DNS, troyanos y ofuscación de URL. El atacante registra la información interceptada para su uso indebido en futuros casos (Alkhalil et al., 2021).

Phishing en motores de búsqueda

Esta técnica de phishing genera sitios webs maliciosos que ofrecen ofertas atractivas mediante técnicas de optimización de motores de búsqueda para lograr su indexación legítima. De esta manera se posiciona sitios web maliciosos aprovechando la confianza de los usuarios que buscan información legítima (Alkhalil et al., 2021).

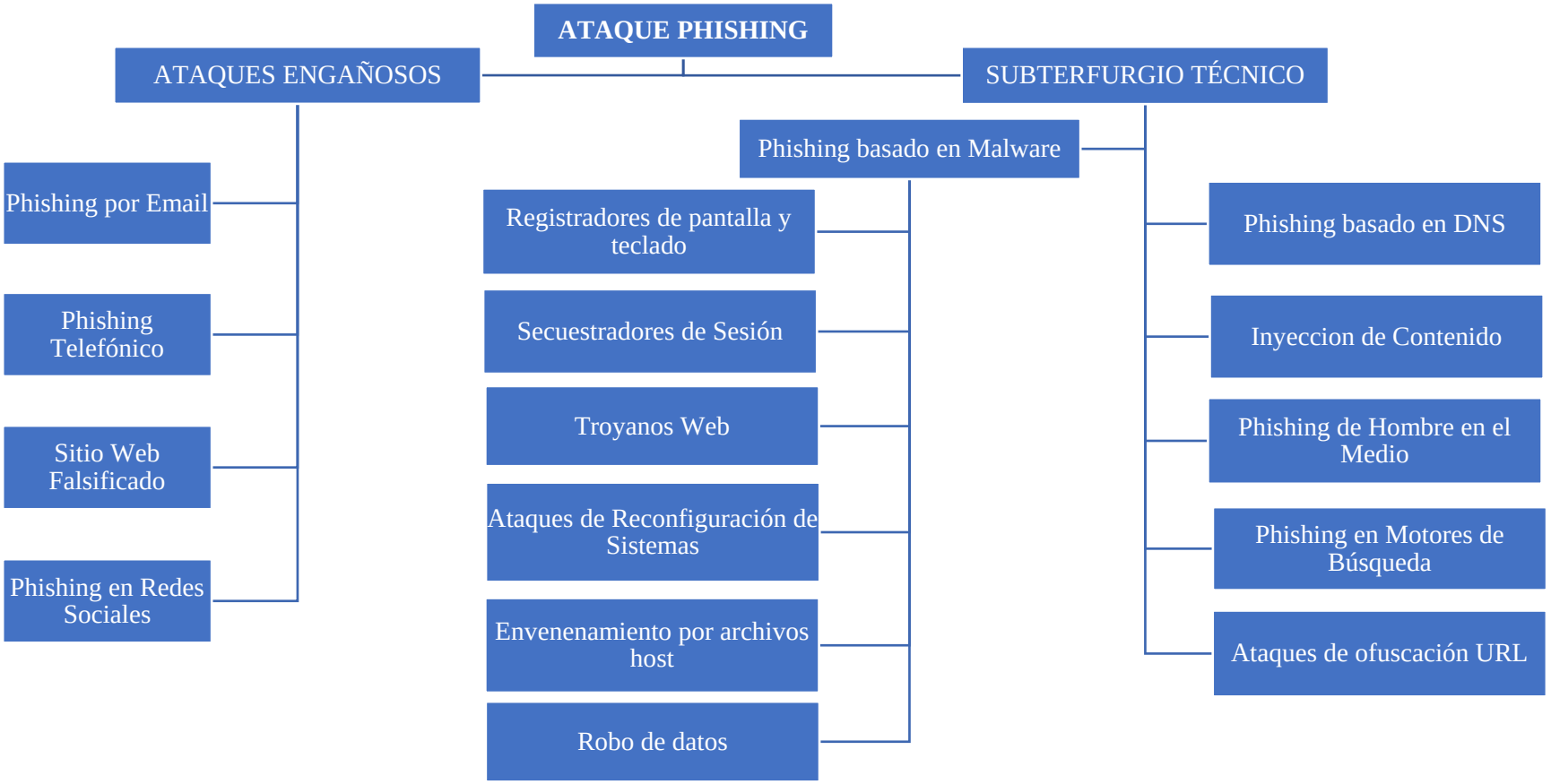
Ataques de ofuscación de URL y HTML

Este ataque busca persuadir al usuario para que acceda a un enlace en específico, el cual redirige a la víctima hacia un servidor malicioso en lugar del destino legítimo. Esta técnica, ampliamente usada, oculta la verdadera dirección web a la que el usuario intenta acceder, para que de esta manera el enlace parezca auténtico. Los métodos comunes utilizados en este tipo de ataque incluyen el uso de nombres de dominio incorrectos y la ofuscación de nombres de host para falsificar la apariencia de la dirección (Alkhalil et al., 2021).

La Figura 2 muestra una recopilación del compendio de información sobre la diversidad de los ataques phishing.

Figura 2

Tipos de Ataques Phishing



Fuente: Adaptado de (Alkhalil et al., 2021)

2.3 Internet de las Cosas (IoT)

El término IoT se lo atribuye a Kevin Ahton. Se define como un sistema compuesto por sensores y actuadores interconectados que posibilita la administración, recopilación e intercambio de información proveniente de sensores. La capacidad de administrar la información implica la presencia de algún sistema de procesamiento, como microprocesadores o microcontroladores (Gualberto, 2021).

El Internet de las Cosas (IoT) se destaca por la conectividad entre dispositivos mediante el internet y diversas tecnologías. Entre las principales características en torno al IoT se destaca: Conectividad, Protocolos de Transmisión de Mensajes y Aplicaciones IoT.

- **Conectividad:** Las aplicaciones de IoT generalmente envían los datos recolectados por un dispositivo a un sistema o subsistemas externos, para el almacenamiento y compartición de la información. La transferencia de datos entre dispositivos y sensores de IoT pueden llevarse a cabo a través de distintos tipos de redes, tales como cableadas, inalámbricas, de máquina a máquina (M2M), como 3G, 4G, 5G, NB-IoT, entre otras opciones (Tseng et al., 2020).

- **Protocolos de Transmisión de Mensajes**: La complejidad de diseñar redes de IoT, con múltiples sensores y limitaciones en el esquema de direcciones, demanda una gestión avanzada de recursos por lo que se acentúa la importancia de estrategias efectivas para la transferencia de datos, que influya de manera significativamente en el ancho de banda de la red (Tseng et al., 2020). Para garantizar seguridad y eficiencia, se han desarrollado varios protocolos de

mensajería específicos dentro de los cuales se destacan CoAP, MQTT, XMPP y AMQP.

- **Aplicaciones IoT:** La expansión del uso de la Internet de las Cosas (IoT) en diversos campos respaldan y explican el crecimiento de esta nueva tendencia, llevando a una adopción más generalizada de la IoT en la sociedad actual. La exploración de las aplicaciones específicas de IoT contribuye a una comprensión más profunda y al perfeccionamiento continuo de esta tecnología, facilitando así el diseño de sistemas innovadores adaptados a situaciones recientemente desarrolladas (Tseng et al., 2020). Dentro de las áreas de aplicación se sitúan:

    - Aplicaciones Médicas
    - Aplicaciones Ambientales
    - Aplicaciones de Infraestructura.
    - Aplicaciones Industriales
    - Aplicaciones Comerciales
    - Aplicaciones para Ciudades Inteligentes

2.4 IDS

Un Sistema de Detección de Intrusos (IDS) hace referencia a un sistema de hardware o software que realiza monitoreos de manera automática con el objetivo de identificar ataques o intrusiones. Posteriormente, se generan y envían alertas a la red o computador (Saranya et al., 2020). Las funciones fundamentales de los Sistemas de Detección de Intrusos incluyen (Vasilomanolakis et al., 2015):

- Monitorear las actividades de red.
- Analizar los datos recolectados.
- Verificar las configuraciones del sistema y buscar vulnerabilidades.
- Detectar patrones o firmas.

- Almacenar patrones o firmas en una base de datos.
- Emitir alertas en casos de coincidencia con algún patrón o firma detectado.

El sistema de Detección de Intrusos integra los siguientes componentes:

- **Red de monitoreo:** Consiste en la recopilación de paquetes de red que contienen datos relevantes para la detección de intrusos. Los paquetes de red se conforman de un encabezado y una carga útil, la información anexada a ambos elementos puede ser explotada para ejecutar un ataque. Además, se analiza el flujo de red para identificar patrones que podrían ser empleados con fines maliciosos. Los conjuntos de datos de detección de intrusiones incluyen características de nivel de paquete y de flujo para clasificar eficazmente los ataques (Thakkar & Lohiya, 2022).
- **Recopilación de datos:** Recopila datos detallados sobre el sistema objetivo para planificar el ataque. Este proceso se lleva a cabo mediante consultas empleando comandos como para determinar detalles del servidor y host o a través de herramientas de red que permitan rastrear la actividad de red (Thakkar & Lohiya, 2022).
- **Análisis de los detalles del paquete:** Es el proceso de escaneo del paquete de red que busca robar información confidencial, como ataques del tipo Remoto a Local (R2L) que comprometen el sistema para obtener acceso no autorizado. Entre los ataques que pueden obtener acceso se sitúan el rastreo de paquetes, robo de credenciales o inyección de malware (Thakkar & Lohiya, 2022).
- **Identificar y almacenar la firma/patrones de ataque:** Consiste en identificar los patrones de ataque, tanto de los modelos establecidos como de los nuevos, así como las firmas de los programas que aprovechan alguna vulnerabilidad (exploits) para provocar comportamientos anómalos y ataques internos. Los modelos y firmas se registran en una base de datos para su posterior consulta, lo que

simplifica la detección de comportamientos anómalos y la implementación de medidas adecuadas (Thakkar & Lohiya, 2022).

- **Generación de alerta:** Posterior al reconocimiento del ataque se emite una advertencia y notifica al administrador de seguridad. La alerta se activa con base a la coincidencia de la firma/patrón (Thakkar & Lohiya, 2022).

2.4.1 Clasificación de los Sistemas de Detección de Intrusos (IDS)

Los Sistemas de Detección de Intrusos se clasifican según el tipo de implementación, técnica de detección, frecuencia de uso, tipo de respuesta, estructura y conocimiento (Tidjon et al., 2019).

Clasificación por Tipo

- IDS basado en Red

Se ubica en la Zona Desmilitarizada (DMZ) después del firewall (Tidjon et al., 2019). Monitorea y analiza los segmentos y capas de la red para identificar tanto tráfico como comportamiento malicioso (Torres Ruiz, 2019).

- IDS basado en Host:

Protege únicamente el equipo en el que fue instalado y su funcionamiento radica en la inspección de los archivos, servicios y software que contiene el equipo anfitrión. Una vez se detecte el comportamiento malicioso se generan y envían alertas al administrador (Torres Ruiz, 2019).

Clasificación según la Estructura

- IDS Centralizado

Se concibe en base al posicionamiento de los agentes del sistema de detección dentro del núcleo de la red, por ejemplo, en un enrutador. El punto central o agente IDS conserva un registro de las peticiones entre los diferentes nodos de la red, además, que analiza la información transmitida entre los mismos. A nivel estructural este tipo de IDS

presenta desafíos, debido a que el análisis del tráfico en un solo nodo no es suficiente y en circunstancias en las cuales el nodo central se ve comprometido (Thakkar & Lohiya, 2021).

- IDS Distribuido

En este mecanismo de detección de intrusos los agentes IDS se implementa en cada dispositivo de la red (Thakkar & Lohiya, 2021) de manera que el procesamiento y el análisis de tráfico, así como la detección de eventos anómalos se ejecutan en cada uno de los agentes. La distribución del sistema resulta de la segmentación de la inteligencia entre los gestores y agentes de la red (Pathan, 2014). El monitoreo se limita solo al nodo en sí, de manera que, no se puede aprovechar el resto de los nodos de la red para tomar decisiones más precisas (Thakkar & Lohiya, 2021).

Clasificación según las Técnicas de Detección

- IDS basado en Firmas

El método de detección se basa en la coincidencia de firmas del ataque del tráfico, con la firma alojada en la base de datos. La detención opera de mejor manera con ataques conocidos, además, este tipo de detección es más sencilla de implementar. La fiabilidad de detección decrece con ataques nuevos o variantes de ataques conocidos, debido a que la firma de estas no se encuentra presente en la base de datos (Thakkar & Lohiya, 2021).

- IDS basado en anomalías:

Este sistema de detección identifica las intrusiones por medio de la coincidencia de patrones. El IDS basado en anomalías genera un perfil de usuario a través del análisis de tráfico normal del sistema (Tidjon et al., 2019). En el proceso de monitoreo se analiza si el tráfico actual coincide con los patrones de un comportamiento normal; en caso de presentarse desviaciones en relación con el perfil, se genera una alerta de comportamiento anómalo (Torres Ruiz, 2019).

Clasificación por tipo de respuesta

- IDS Pasivo

Su modo de configuración se basa en el monitoreo y análisis del tráfico en la red. Por medio de notificaciones, alerta al operador de posibles vulnerabilidades y ataques. Este tipo de IDS carece de la facultad de generar acciones de protección o corrección de manera autónoma. Entre sus ventajas se caracteriza su fácil y rápida implementación además de su tolerancia a ataques directos (Miller & Gregory, 2022).

- IDS Activo

También conocido como Sistema de Prevención de Intrusos IPS (Achary, 2021). Este tipo de sistema está programado para contener de manera automática conexiones o tráfico sospechoso en curso, sin que algún operador intervenga. El IPS ofrece la ventaja de ejecutar medidas correctivas en tiempo real, sin embargo, al implementarse en el límite de la red, también se expone a ataques directos como los del tipo de Denegación de Servicios (Miller & Gregory, 2022).

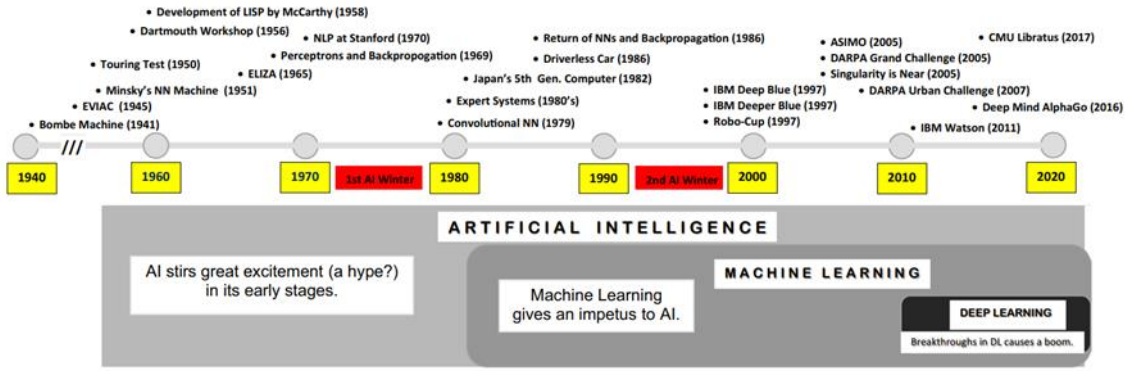
2.5 Inteligencia Artificial

La Inteligencia Artificial o AI es una tecnología que permite a las computadoras reproducir comportamientos inteligentes como el aprendizaje, juicio y la toma de decisiones (Oestriecher & Beasley, 2020). La IA trata a la información como un recurso, para el análisis y estudio de los diferentes métodos de expresión del conocimiento. La recopilación de diversas disciplinas como las ciencias de la computación, biología, psicología y filosofía ha permitido el desarrollo de aplicaciones orientadas al reconocimiento de voz, procesamiento de imagen, procesamiento de lenguaje natural, robótica y automatización de procesos (Zhang & Lu, 2021).

La Figura 3 presenta una línea de tiempo en la que se destacan los principales hitos en la evolución de la inteligencia artificial.

Figura 3

Evolución de la Inteligencia Artificial



Fuente: Tomado de (Kaynak, 2021).

Desde los primeros intentos de automatización durante la Segunda Guerra Mundial con la Máquina Bombe en 1941, hasta los avances más recientes como el desarrollo de vehículos autónomos y sistemas de procesamiento de lenguaje natural, la evolución de la inteligencia artificial ha sido una historia de hitos significativos. La prueba de Turing en 1950 planteó la pregunta inicial sobre si una máquina podría imitar el comportamiento humano. La década de 1950 vio el nacimiento oficial de la IA en el Dartmouth Workshop, mientras que, en 1958, John McCarthy desarrolló el lenguaje de programación Lisp, crucial para la investigación en IA. En años posteriores, avances como la Máquina de Redes Neuronales de Marvin Minsky en 1951 y la creación de ELIZA en 1965 destacaron la exploración de la simulación de la inteligencia humana. En las décadas de 1970 y 1980 se presenciaron el surgimiento de redes neuronales convolucionales, sistemas expertos y el proyecto japonés de la Quinta Generación de Computadoras. El renacimiento de las redes neuronales y la retro propagación en 1986 allanaron el camino para futuros desarrollos, incluido el surgimiento de vehículos autónomos en 1986. A su vez algunos acontecimientos tales como la victoria de IBM

Deep Blue sobre Garry Kasparov en 1997 y los avances en procesamiento de lenguaje natural con IBM Watson en 2011 marcaron hitos clave en la evolución de la inteligencia artificial. En la última década, AlphaGo de DeepMind (2016) y Libratus de la Universidad Carnegie Mellon (2017) demostraron la excelencia de la IA en juegos estratégicos. Además de nuevos proyectos como el Darpa Urban Challenge (2007) y desarrollos continuos en NLP, han mantenido el impulso de la IA en la sociedad, prometiendo un futuro emocionante en el que la inteligencia artificial sigue desafiando y superando las capacidades humanas en diversas áreas (Kaynak, 2021).

2.5.1 Tipos de Inteligencia Artificial

Inteligencia Artificial Débil

Hace referencia a los sistemas de IA que imitan capacidades cognitivas humanas para abordar tareas específicas; sin embargo, carecen de un mecanismo de adaptación frente a situaciones nuevas o desconocidas (Gil, 2023). Dichos sistemas se emplean en aplicaciones que operan mediante técnicas de aprendizaje automático o profundo y procesamiento de lenguaje natural (NLP) con el fin de identificar patrones a partir de un ingente volumen de datos. Entre estas aplicaciones se encuentran aquellas que ejecutan procesos iterativos tales como el reconocimiento de imágenes, chatbots o vehículos autónomos. Entre otras áreas de uso se sitúan los sistemas de seguridad, redes de comunicación y robótica (Chung et al., 2022).

Inteligencia Artificial General

Son los sistemas de cómputo que poseen capacidad de razonamiento, aprendizaje y resolución de un gran espectro de tareas, además de que pueden realizar ajustes frente a nuevos escenarios o entornos (Gil, 2023).

Superinteligencia

También conocida como Super IA, hace referencia a la inteligencia artificial que se posiciona más allá del punto de singularidad (Gil, 2023). Es aquella inteligencia que supera a la humana en diversos dominios cognitivos generales y considerablemente más eficiente debido a una mayor capacidad de almacenamiento, procesamiento de información y habilidades analíticas (Chung et al., 2022).

2.6 Plataformas para Simulación de Redes IoT

Las herramientas de simulación brindan la capacidad de determinar aspectos relevantes tales como el rendimiento, comportamiento y precisión, bajo ese contexto se revisan las siguientes plataformas de simulación que suplan los requerimientos necesarios del proyecto.

FIT IoT-LAB

Es un conjunto de herramientas que forma parte del proyecto FIT (Future Internet of Things), que brinda a los usuarios a nivel académico e industrial una infraestructura con miles de nodos inalámbricos para evaluar y experimentar tecnologías inalámbricas en torno al IoT, así como servicios integrados avanzados (Adjih et al., 2015). FIT IoT-LAB proporciona un control completo de los nodos y acceso directo a las puertas de enlace de manera remota, lo cual facilita la supervisión de las métricas de red y energía. La versatilidad de FIT IoT-LAB permite a los desarrolladores crear diferentes pruebas a partir de distintos nodos, entornos, arquitecturas y aplicaciones (Zivkovic et al., 2020).

Cooja Simulator

Es un simulador de código abierto diseñado para replicar sistemas orientados a IoT. Utiliza como base Java y Java Native Interface (JNI), para facilitar la interacción entre el framework y el sistema operativo emulado. Mediante el emulador de nivel de instrucciones basado en Java MSPSim, se tiene la capacidad de emular completamente

sensores de la serie MSP430 (Eriksson et al., 2009), permitiendo así, la simulación de diferentes stack de redes. Cooja ofrece la oportunidad de personalizar la aplicación mediante la inclusión de módulos adicionales. Aprovechando la flexibilidad que proporciona la estructura del simulador, este puede operar de diversas maneras, al modificar: el entorno de radio, el equipo del nodo, los elementos de entrada y salida. El nodo simulado se caracteriza por sus unidades de memoria de datos, tipo de nodo y entorno de hardware, además, de que puede ejecutar directamente la versión compilada de programas desarrollados en Contiki OS u otros emuladores (Islak et al., 2022).

Qualnet

Es una herramienta que permite el diseño de escenarios virtuales de todo tipo de redes. Qualnet se integra de los siguientes componentes:

- **Diseñador de escenarios:** Permite crear escenarios virtuales para diversos tipos de redes.
- **Animador:** Proporciona visualizaciones y análisis de los escenarios.
- **Diseñador de protocolos:** Facilita la implementación de protocolos personalizados.
- **Analizador:** Ofrece análisis estadístico.
- **Rastreador de paquetes:** Genera archivos de seguimiento relacionados con las transferencias de paquetes.

Los modelos de QualNet están formados por nodos y enlaces que los conectan. Los nodos representan puntos finales de la red, mientras que los enlaces de conexión pueden ser configurados como redes inalámbricas o cableadas. QualNet brinda solidez en la simulación de todo tipo de redes, ya sean cableadas, inalámbricas o híbridas, especialmente situaciones en las que es esencial modelar con precisión redes de alta fidelidad para su posterior análisis. La interfaz gráfica de usuario facilita a los diseñadores

de redes la construcción de sus proyectos en entornos tanto 2D como 3D (Kassab & Darabkh, 2020).

CupCarbon

Es un simulador de eventos discretos que permite el diseño de redes de sensores inalámbricos (WSN) y la configuración de componentes del sensor como batería, alcance de radio enlace. CupCarbon integra una interfaz geográfica compatible con OpenStreetMap por la cual es posible localizar los sensores de la red (Mehdi et al., 2014). El simulador cuenta con 3 componentes:

- Entorno de Simulación Multiagente
- Entorno de Simulación Móvil
- Simulador de redes de sensores inalámbrico (WSN).

La extensión de SenScript brinda la capacidad para configurar cada nodo sensor de manera individual, lo cual posibilita la división o unión de redes (Patel & Gupta, 2018). Además, incorpora tecnologías inalámbricas como ZigBee, LoRa y Wi-Fi (Chaudhari & Zennaro, 2020).

OMNET++

Es un simulador de eventos discretos empleado para modelar redes cableadas, inalámbricas, en chips, de sensores entre otras opciones. OMNET++ se compone de módulos que pueden ser simples o compuestos. Los módulos actúan como interfaces en el intercambio de mensajes y se rigen bajo una jerarquía de conexión. Una red es un módulo compuesto que carece de puertas al mundo exterior, por el contrario, los módulos simples implementan tanto el comportamiento como las funciones de inicialización y finalización (Nardini et al., 2020).

Los modelos de OMNET++ se divide en 3 partes:

- **Comportamiento:** Implementa los algoritmos y la lógica de los módulos de simulación.
- **Descripción:** Define las puertas, conexiones y parámetros del modelo por medio del lenguaje de descripción de red (NED).
- **Valores de los parámetros:** Define los valores específicos responsables de la inicialización del modelo a través de archivos de inicialización (INI).

OMNET++ simplifica el flujo del proceso de simulación al automatizar pasos que tienden a ser lentos y propensos a errores. El entorno de desarrollo facilita la depuración al permitir a los usuarios:

- Examinar módulos.
- Habilitar o deshabilitar la generación de texto durante la ejecución.
- Observar el flujo de mensajes mediante una animación.
- Representar eventos en un diagrama de tiempo.

NS-3

Network Simulator 3 (NS-3) es un entorno de simulación de redes de código abierto escrito en C++ que opera basándose en eventos discretos, cada evento está asociado a un tiempo de ejecución y se efectúan en orden temporal. Su estructura de eventos discretos permite la recreación rápida y escalable de escenarios en tiempo real. NS-3 brinda la capacidad de configurar y analizar redes cableadas o inalámbricas, además de que permite evaluar protocolos y el rendimiento de aplicaciones.

NS-3 ofrece módulos esenciales, como núcleo, Internet, Ethernet, enrutamiento, aplicaciones y monitoreo de estadísticas de flujo. Sus características únicas, como la compatibilidad con registro, depuración, rastreo y computación de estadísticas de flujo confieren un valor para aquellos que realizan simulaciones. Además, NS-3 es compatible con NetAnimator (NetAnim) para visualizar simulaciones de manera efectiva.

Mininet

Mininet es un emulador de red que puede crear una red de conmutadores, controladores, hosts y enlaces virtuales. Los hosts Mininet utilizan software de red Linux estándar, y sus conmutadores utilizan OpenFlow para redes definidas por software y enrutamiento personalizado altamente flexible.

Mininet permite la investigación, el desarrollo, el aprendizaje, la creación de prototipos, las pruebas, la depuración y cualquier otra cosa que pueda recuperar. MiniNet es una plataforma que integra una variedad de dispositivos, como hosts finales, conmutadores, enrutadores y enlaces, y funciona en un solo núcleo de Linux. Su técnica de virtualización ligera permite que un solo sistema opere como una red completa con el mismo núcleo, sistema y código de usuario. A través de la conexión de red al host mediante SSH, cada host en Mininet actúa como una máquina real, lo que permite la interacción y la ejecución de programas específicos.

Estos programas pueden usar interfaces Ethernet virtuales para enviar paquetes de datos que simulan la velocidad y el retraso de enlace reales. Los paquetes son procesados con colas de datos específicas como si estuvieran pasando por un conmutador, enrutador o middlebox Ethernet auténtico. El rendimiento medido dentro de la simulación de Mininet debe ser comparable al de dos máquinas nativas que ejecutan dos programas para establecer una comunicación en la red, similar a un cliente y un servidor de una herramienta de pruebas de red como iperf (Lantz et al., 2017).

La Tabla 1 muestra una comparativa de las diversas opciones de simuladores de red disponibles con base en la revisión de las características expresada en el Anexo 1.

Tabla 1

Comparativa de Simuladores de Red

Simulador	Campo de Aplicación	Lenguaje	Tipo	Capa de Arquitectura IoT	Escalabilidad	Soporte de Estándares	Practicidad General	Simulación de Ataques
Cooja	Red	C, C++	Eventos Discretos	Percepción y Red	Pequeña Escala	Todos los protocolos IoT	Alta	Mediante Extensiones Personalizadas
QualNet	Red	C, C++	Eventos Discretos	Percepción y Red	Gran Escala	Zigbee/802.15.4	Media	Si
CupCarbon	Red	SenScript	Eventos Discretos y Basado en Agentes	Percepción y Red	Gran Escala	LoRaWAN/802.15.4	Alta	No
OMNeT++	Red	C++, Python	Eventos Discretos	Percepción y Red	Gran Escala	Extensiones Manuales	Media	Mediante Extensiones Personalizadas
NS-3	Red	C++, Python	Eventos Discretos	Percepción y Red	Gran Escala	LoRaWAN, 802.15.4, 6LoWPAN	Alta	No
Mininet-WiFi	Red	Python	Eventos Discretos	Percepción y Red	Pequeña y Mediana Escala	6LoWPAN, HTTP/HTTPS, CoAP, MQTT, DDS	Media	Si
FIT IoT-LAB	Red	C, C++, Python, Java, JavaScript, Node.js	Basado en Eventos	Percepción, Red y Aplicación	Pequeña, Mediana y Gran Escala	MQTT, CoAP, LoRaWAN, ZigBee, 6LoWPAN, BLE, WiFi, HTTP, HTTPS	Media	Si

2.7 Procesamiento de Lenguaje Natural (NLP)

Es la rama de la inteligencia artificial que incorpora la lingüística en los procesos de interacción entre el computador y el leguaje humano, con el objetivo de proveer a las máquinas, mecanismos de comprensión e interpretación. Su foco de investigación se centra en cómo las máquinas adquieren conocimiento, al emplear modelos lingüísticos basados en el lenguaje humano, específicamente el lenguaje natural. Su propósito fundamental es llevar a cabo diversas tareas relacionadas con la búsqueda de información, tales como la clasificación de textos, la recuperación de información y la extracción de datos (Gualberto, 2021).

En base al objetivo principal de NLP que consiste en el procesamiento de información se distingue generalmente las siguientes fases:

Palabras Vacías

Las palabras vacías (stopwords) guardan relación con los términos que carecen de contenido léxico o aportan escaso valor al significado de una oración. Dentro del conjunto de palabras vacías de manera general se incluyen a las preposiciones y artículos.

Tokenización

La tokenización se lleva a cabo en el texto restante a nivel de términos por medio del uso de espacios, tabulaciones y líneas nuevas como delimitadores, dividiendo cada texto en términos (tokens). La etapa de tokenización es crucial en el proceso de creación de vocabulario para el corpus, así como para otras acciones de procesamiento de lenguaje natural como la eliminación de stopwords.

Etiquetado de Partes del Discurso (POS)

La tarea de asignación de Parte del Discurso (POS) consiste en asignar una categoría gramatical a cada token, indicando si se trata de un adjetivo (ADJ), un adverbio (ADV), un sustantivo (SUSTANTIVO), un pronombre (PRON), o un verbo (VERBO).

Lematización

La lematización convierte una palabra a su forma base común. Para simplificar las formas derivadas relacionadas de una palabra, este proceso suele implicar el uso de un vocabulario y un análisis morfológico. La palabra separa el morfema raíz (raíz) de sus morfemas accesorios (afijos), devolviéndola a su forma de lema, es decir, obteniendo la raíz en su forma de diccionario.

CAPÍTULO III. Metodología y Diseño del Sistema

En esta sección, se describe el proceso de diseño del sistema de detección de ataques a través de la metodología en cascada, el cual emplea un esquema secuencial enfocado en procesos lineales, aplicando tecnologías emergentes como el procesamiento de lenguaje natural (NLP), el cual analiza, identifica y clasifica el contenido de la información con base en el contexto lingüístico, posteriormente se aborda un análisis de los requerimientos para el funcionamiento del sistema.

3.1 Metodología

El establecimiento de fases y procedimientos brinda un enfoque claro sobre cómo abordar el problema, lo que a su vez facilita el desarrollo apropiado del trabajo de titulación. Bajo este contexto se realiza una investigación preliminar en fuentes bibliográficas que guarden relación con el tema, de esta manera se identifica los aspectos más relevantes de la arquitectura IoT, las clases de amenazas o ataques que tienen como objetivo el robo de información y las técnicas de detección que emplean mecanismos de procesamiento de lenguaje natural (NLP).

Los criterios mencionados permiten establecer una solución que integre el procesamiento de lenguaje natural para determinar ataques. El desarrollo del diseño y esquema de simulación derivan del proceso de identificación de los elementos fundamentales realizado en el apartado de investigación del capítulo anterior y la búsqueda de trabajos relacionados que se realizara en la sección de análisis.

3.1.1 Modelo en Cascada

También conocido como el modelo de ciclo de vida secuencial lineal, este enfoque consta de cinco fases, donde cada una debe completarse antes de avanzar a la siguiente. Este modelo es ampliamente utilizado en el desarrollo de software para garantizar el éxito

del proyecto (Senarath, 2021). Dado el propósito del trabajo, que consiste en desarrollar un sistema de detección de phishing mediante inteligencia artificial, y considerando el enfoque del modelo en cascada, se selecciona esta metodología para la creación del sistema. A continuación, se expone las fases que cubre el modelo en cascada:

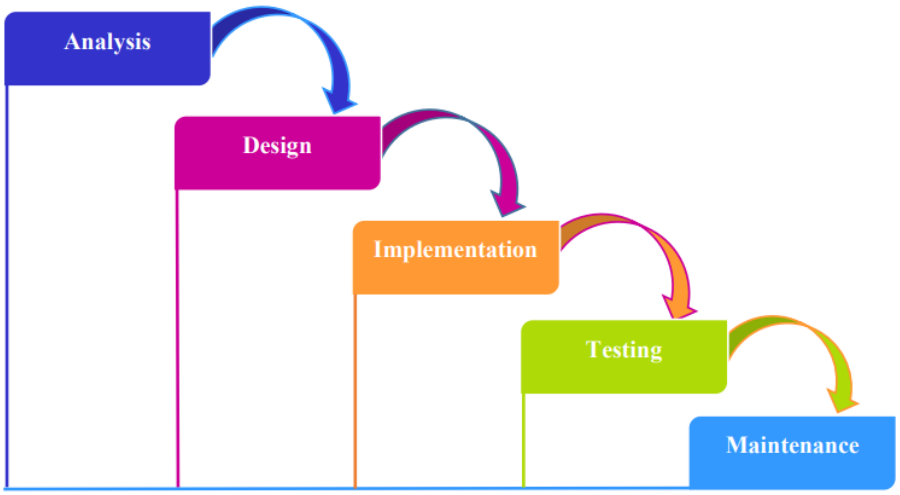
- **Análisis:** Es la fase inicial en la que se definen los requisitos del software (SRS). Los expertos en sistemas y negocios trabajan en conjunto para identificar tanto los requerimientos funcionales como los no funcionales del software. Los requisitos funcionales están relacionados con las interacciones del usuario con el software; por el contrario, los no funcionales engloban parámetros de fiabilidad, escalabilidad y rendimiento. Esta fase establece las bases para el desarrollo del proyecto, definiendo de manera clara lo que debe hacer el software y en qué condiciones.
- **Diseño:** En esta fase, el equipo técnico traduce los requisitos en un plan detallado para la construcción del software. Este plan incluye los componentes clave del software, tales como algoritmos, arquitectura del software, base de datos, diagramas lógicos e interfaz de usuario.
- **Implementación:** Es la fase en la cual los programadores escriben el código fuente, desarrollan la base de datos, elementos visuales, así como el resto de los componentes necesarios, en esta fase se transforma los diseños, los requerimientos y los planes en un producto tangible y funcional.
- **Pruebas:** También denominado verificación y validación, en este punto el software se somete a un escrutinio riguroso para asegurar que cumpla con los requisitos y funciones especificados. Durante este proceso, se identifican y corrigen los fallos del sistema, garantizando así la calidad y fiabilidad del producto final.

- **Mantenimiento:** Constituye la fase final del modelo de desarrollo de software, durante la cual se implementan modificaciones para optimizar el rendimiento, subsanar errores desapercibidos y de ser necesario adaptar el software a nuevos requisitos. Esta fase asegura que el software se mantenga útil y eficiente a lo largo de todo su ciclo de vida.

La Figura 4 ilustra las fases del modelo en cascada, en la cual se aprecia de manera clara el proceso lineal característico del modelo.

Figura 4

Modelo en Cascada



Fuente: Tomado de (Senarath, 2021)

3.2 Análisis

El análisis se enfoca en determinar los requerimientos del sistema, abarcando tanto hardware como el software y herramientas necesarias para el diseño. Entre los componentes requeridos, se precisa de un software que permita simular entornos de red, herramientas que faciliten el almacenamiento y visualización de datos, así como de un software que permita monitorear la red. Adicionalmente, se lleva a cabo un análisis de la situación actual en torno a los ataques phishing y una revisión de trabajos que permita

identificar mecanismos de detección relacionados al uso de inteligencia artificial en ataques phishing.

3.2.1 Situación Actual

El phishing se ha convertido en una de las amenazas cibernéticas de mayor predominio actualmente. Este tipo de ataque se basa en engañar a la víctima para que revele información sensible, como contraseñas, información de cuentas bancarias o credenciales de acceso. Aunque tradicionalmente se asocia con correos electrónicos engañosos, el phishing engloba una amplia gama de técnicas de engaño.

En los últimos años, se ha observado un incremento significativo de los ataques de phishing. Compañías como (SlashNext, 2023) mencionan un aumento de correos maliciosos de phishing, impulsados en gran medida por medio de herramientas de Inteligencia Artificial Generativa (GenAI) como ChatGPT. De hecho, el informe de riesgos emergentes de (Gartner, 2024) sitúa el uso de IA en las tres primeras categorías de amenazas.

Las soluciones contra los ataques phishing no solo se centran en la capacitación del usuario, sino que también incorporan métodos de detección que emplean técnicas de inteligencia artificial. Según (Alanezi, 2021), los métodos de detección se clasifican en las siguientes categorías:

- Métodos tradicionales
    - Legal, Educativo y de Concientización.
    - Métodos de listas negras y listas blancas
    - Métodos de similitudes visuales
    - Métodos de motores de búsqueda

- Métodos no tradicionales
    - Métodos basados en contenido
    - Métodos basados en heurística
    - Métodos de aprendizaje automático
    - Métodos de aprendizaje profundo
    - Métodos basados en reglas difusas
    - Métodos de aprendizaje híbrido
    - Métodos de minería de datos

En respuesta a la creciente amenaza, tanto las empresas como los usuarios han comenzado a adoptar medidas más proactivas para protegerse. Sin embargo, los métodos de detección convencionales podrían resultar insuficientes ante la constante adaptación y evolución de las técnicas de phishing.

3.2.2 Revisión de Trabajos Relacionados

La revisión de trabajos relacionados en el contexto de este trabajo de grado tiene como premisa identificar los requerimientos tecnológicos aplicables a los procesos de diseño e implementación de un sistema de detección de ataques phishing mediante inteligencia artificial. El proceso de revisión se fundamenta en dos criterios principales: la metodología de detección y el tipo de ataque phishing. Estos dos parámetros servirán como guía para la adecuada selección de algunos requisitos que precisa el sistema.

La información relevante de cada trabajo se ha extraído por medio de fichas bibliográficas, cuyo contenido detallado se presenta en el Anexo 2.

La Tabla 2 muestra una síntesis de los datos clave de cada uno de los trabajos revisados, mientras que descripción más exhaustiva de cada investigación pueden ser hallados en el Anexo 2 de este trabajo.

Tabla 2

Trabajos relacionados

Ficha bibliográfica 1	
Autores	Denish Omondi Otieno, Akbar Siami Namin & Keith S. Jones
Editorial	Institute of Electrical and Electronics Engineers (IEEE)
Título	The Application of the BERT Transformer Model for Phishing Email Classification
Año	2023
Tipo	Phishing basado en correo electrónico
Ficha bibliográfica 2	
Autores	Waafi Abdullah Bin Adam, Ivriah Yue Xuan Tan, Chung Shing Lai, Nurtiara Tania Rahim, Benjamin Yu Bin Tham & Huaqun Guo
Editorial	Institute of Electrical and Electronics Engineers (IEEE)
Título	VishingDefender: An Advanced Vishing Defence System Against Vishing Attacks
Año	2023
Tipo	Phishing telefónico
Ficha bibliográfica 3	
Autores	Bianca Montes Jones & Marwan Omar
Editorial	Institute of Electrical and Electronics Engineers (IEEE)
Título	Detection of Twitter Spam with Language Models: A Case Study on How to Use BERT to Protect Children from Spam on Twitter
Año	2023
Tipo	Phishing en redes sociales
Ficha bibliográfica 4	

Autores	Rebet Jones, Marwan Omar, Derek Mohammed, Calvin Nobels & Maurice Dawson
Editorial	Institute of Electrical and Electronics Engineers (IEEE)
Título	IoT Malware Detection with GPT Models
Año	2023
Tipo	Phishing basado en malware

Ficha bibliográfica 5

Autor	Ming-Yang Su & Kuan-Lin Su
Editorial	Sensors
Título	BERT-Based Approaches to Identifying Malicious URLs
Año	2023
Tipo	Phishing basado en url

3.2.3 Análisis de la Revisión de Trabajos Relacionados

- Trabajo 1: El marco técnico del trabajo se fundamenta en el uso del mecanismo de análisis de sentimientos y modelado de temas para identificar patrones distintivos entre correos electrónicos de phishing y no phishing. La experimentación se lleva a cabo en el entorno de Google Colab con el fin de aprovechar las capacidades de computación en la nube para el tema de procesamiento de datos y entrenamiento de modelos. Para el proceso de clasificación se emplea el modelo BERT (Bidirectional Encoder Representations from Transformers) y el conjunto de datos empleado para el entrenamiento proviene de dos fuentes principales: corpus de phishing de Nazario y el dataset de Enron.

- Trabajo 2: En este trabajo se emplea la metodología de conversión de voz a texto, seguido de un análisis comparativo entre algoritmos predictivos y bibliotecas modernas centradas en el procesamiento de lenguaje natural (NLP). Los algoritmos predictivos incluyen Coseno Similarity, Trie y KMP, mientras que librerías de procesamiento de lenguaje natural utilizadas son Gensim, SpaCy, NLTK y SciKit-Learn. Esta comparación permite determinar la eficacia en la identificación de patrones lingüísticos asociados con intentos de ataques vishing. Para el entrenamiento y validación, se emplea el conjunto de datos 'Text-classification-of-voice-phising-transcipts', obtenido del repositorio de GitHub. La solución desarrollada se materializa a manera de aplicación móvil para android y un bot de Telegram, las cuales generan alertas en tiempo real sobre el posible fraude.
- Trabajo 3: En este trabajo se desarrolla un sistema de detección de spam en tweets, utilizando una metodología que consta de 3 fases. Estas fases corresponden al procesamiento de datos (limpieza, normalización, eliminación de caracteres), extracción de características por medio del algoritmo TF-IDF (Term Frequency-Inverse Document Frequency) y el modelado mediante ajuste fino de BERT para detectar spam en tweets relevantes para niños. El conjunto de datos recolectados empleado para el ajuste fino se clasifica de manera preliminar en las categorías de "spam" y "no spam" a través del algoritmo de Naive Bayes. Esta clasificación proporciona la base para el entrenamiento del modelo BERT.
- Trabajo 4: El mecanismo de detección empleada en el trabajo se desarrolla en torno a cuatro etapas, inicialmente se extraen secuencias de código de máquina para el análisis subsiguiente que consiste en el proceso tokenización mediante la plataforma de Hugging Face Transformers. Posteriormente, se convierten los

tokens en vectores por medio de la capa de embedding del modelo GPT (Generative Pre-trained Transformer) para de esta manera hallar la semántica de las secuencias de código. Finalmente se clasifican las secuencias de código y se determina la presencia de malware.

- Trabajo 5: La metodología empleada para la detección de URLs maliciosas consta de tres procesos claves que aprovechan en gran medida la potencia de procesamiento de lenguaje natural que brinda el modelo BERT. La metodología parte con el preprocesamiento de datos, mediante el cual se identifican características por medio del algoritmo de bosque aleatorio. El siguiente proceso, correspondiente a la tokenización, emplea el modelo 'bert-base-cased' para capturar la semántica de las URLs sin comprometer su integridad estructural. El último proceso comprende el entrenamiento por medio del ajuste fino del modelo BERT.

3.3 Requerimientos

La presente sección muestra los factores requeridos para el cumplimiento de los objetivos, así como las medidas de implementación del sistema para de esta manera satisfacer el alcance establecido.

Por lo que se presenta un análisis de los requerimientos aplicables al proyecto, tomando en cuenta la investigación bibliográfica y documental que mencionan las herramientas de simulación de redes IoT.

La ejecución y cumplimiento del apartado toma como referencia de desarrollo la norma (ISO/IEC/IEEE 29148, 2018) y la Arquitectura de requisitos del Sistema (SyRA) considerando la aplicación de adecuaciones según se requiera.

A continuación, se muestra las consideraciones de requerimientos que define el estándar ISO/IEC/IEEE 29148:2018:

- **StRS:** Especificación de Requerimientos de las Partes Interesadas (Stakeholders).
- **SyRS:** Especificación de Requisitos del Sistema.
- **SRS:** Especificación de Requerimientos de Software.
- **SyRA:** Especificación de Requerimientos de Arquitectura.

3.3.1 Especificación de Requisitos de las Partes Interesadas (StRS)

Los Stakeholders determinan las necesidades y requisitos de los usuarios, clientes, patrocinadores, es decir las partes interesadas con el fin de asegurar la cohesión del sistema con los requisitos empresariales y los objetivos estratégicos.

En consideración a los principales aspectos del sistema, se especifican a las partes interesadas de la solución propuesta en la Tabla 3.

Tabla 3

Requerimientos de Stakeholders

Nro.	Stakeholders	Detalles
StRS1	Ing. Fabián Cuzme	Tutor de Tesis
StRS2	Ing. Henry Farinango	Asesor de Tesis
StRS3	Edison Picuasi	Estudiante

3.3.2 Especificación de Requisitos del Sistema (SyRS)

El estándar ISO/IEC/IEEE 29148:2018 destaca la importancia de los requerimientos dentro del esquema de diseño del sistema, denotando el rol que desempeña cada elemento, así como su respectiva función en el proceso de detección por lo que las etapas de diseño, implementación y testeo giran en torno a las SyRS.

Considerando los mencionados parámetros, se establece la Tabla 4 acerca de los requerimientos del sistema:

Tabla 4

Definición de los Requisitos del Sistema

Nro.	Requerimiento	Prioridad		
		Alta	**Media**	**Baja**
SyRS1	Capacidad de virtualización e interacción entre nodos.	X		
SyRS2	Capacidad de virtualización de la comunicación inalámbrica.	X		
SyRS3	Capacidad para monitorear el comportamiento de la red IoT.	X		
SyRS4	Capacidad para generar alertas en caso de detección de ataques.	X		
SyRS5	Compatibilidad con técnicas de NLP para incrementar el grado de detección de ataques	X		
SyRS6	Capacidad de detección, notificación y respuesta en lapsos cortos.		X	
SyRS7	Capacidad de configuración de variables en función del ataque a implementar.	X		
SyRS8	Capacidad para implementar protocolos de comunicación.		X	
SyRS9	Compatibilidad con hardware y software de terceros.		X	

SyRS10	Capacidad para ejecutar pruebas de funcionamiento y mantenimiento.	X		

3.3.3 Especificación de Requisitos de Arquitectura (SyRA)

En este apartado se muestra la estructura del hardware y software que constituye el sistema, los cuales permiten satisfacer los requerimientos del proyecto.

La Tabla 5 muestra la ponderación de los requisitos que servirán como base para el proceso de desarrollo del sistema.

Tabla 5

Definición de los Requisitos de la Arquitectura del Sistema

Nro.	**Requisitos**	**Prioridad**		
		Alta	**Media**	**Baja**
SyRA1	El sistema debe permitir la conexión inalámbrica entre los nodos y el Gateway.	X		
SyRA2	Tiempo de respuesta corta frente a ataques.	X		
SyRA3	Los elementos de la red deben generar y recopilar información de manera constante.		X	
SyRA4	El modelo de NLP debe clasificar el ataque acorde a los parámetros configurados.	X		
SyRA5	Las pruebas deben avalar la detección correcta del ataque en función del tráfico de red.	X		
SyRA6	La red debe ser escalable con relación al número de nodos desplegados en la topología.		X	

SyRA7	La red simulada debe suplir los aspectos básicos de un nodo (procesamiento, conectividad).	X

3.3.4 Especificación de Requisitos del Software (SRS)

El parámetro SRS brinda una descripción del software a desarrollar haciendo énfasis en los requisitos funcionales y no funcionales. Estos requisitos contemplan aspectos relacionados con el software tales como: finalidad, características, rendimiento y comportamiento. Además, guía el proceso de desarrollo y garantiza el cumplimiento de las fases del proyecto.

Los requerimientos de ponderación para el software se establecen en Tabla 6, dichos criterios permitirán establecer la relación con las herramientas a usar en los procesos de simulación e incorporación al sistema.

Tabla 6

Definición de Requisitos de Software

Nro.	Requisitos	Prioridad			Dependencia
		Alta	Media	Baja	
SRS1	El tipo de licencia debe ser de código abierto.	X			
SRS2	El software debe ejecutarse de manera nativa (no requiere una máquina virtual) en el SO anfitrión.		X		SyRS9
SRS3	El lenguaje de programación debe relacionarse a Ciencias de Datos	X			SyRS5

SRS4	El software de ser compatible con módulos y librerías afines a NLP.	X		SyRS5, SyRA4
SRS5	Capacidad para crear y gestionar topologías de red.	X		SyRS1, SyRS2, SyRS8, SyRA1, SyRA7
SRS6	Capacidad para simular redes cableadas e inalámbricas.	X		SyRS2, SyRS8, SyRA1, SyRA7
SRS7	El software debe ser capaz de usar los recursos (interfaces, programas, controladores) del equipo anfitrión.		X	SyRS9

3.3.5 Especificación de Requisitos del Modelo NLP (MRS)

En el sistema de detección de phishing, el modelo NLP tiene su propio grado de relevancia, ya que influye significativamente en la eficiencia y precisión del propio sistema. La elección del modelo determina la capacidad de comprensión del lenguaje y volumen de información que puede manejar. La existencia un número destacable de modelos, cuya idoneidad depende de factores como el área de aplicación, fuente de datos y grado de exactitud influenciaran directamente en el rendimiento del sistema. Por consiguiente, se precisa realizar un análisis para seleccionar el modelo que mejor se adapte a la solución propuesta.

La Tabla 7 detalla el criterio para establecer los requisitos de selección del modelo de lenguaje natural NLP.

Tabla 7

Definición de los Requisitos del Modelo de Procesamiento de Lenguaje Natural (NLP).

Nro.	Requisitos	Prioridad			Dependencias
		Alta	Media	Baja	
MRS1	Capacidad para identificar y clasificar entidades	X			SyRA4
MRS2	Compatibilidad con el lenguaje de programación nativa del software	X			SRS2, SRS3
MRS3	Capacidad para entender el contexto de las palabras	X			
MRS4	Índices de precisión muy altos		X		
MRS5	Tipo de Licencia Open Source	X			SRS1
MRS6	Capacidad para adaptar modelos preentrenados	X			SyRS5, SyRS7, SRS3
MRS7	Capacidad para operar en dispositivos con recursos limitados.	X			

3.4 Diseño

3.4.1 Selección del Software de Simulación

Este apartado se contextualiza en torno a la herramienta de simulación cuya selección radica en las principales premisas que son la facultad para diseñar redes IoT y

la compatibilidad para integrar técnicas de detección de ataques basados en procesamiento de lenguaje natural NLP.

Por consiguiente, la selección del software de simulación se convierte en el foco principal para una adecuada implementación de la solución propuesta y en sí mismo afectará al éxito del proyecto.

De esta manera se realiza un análisis característico de las principales herramientas de simulación en el eje del Internet de las Cosas (IoT) que dará como resultado la selección del software que mejor se adapte a los requerimientos tanto de red como arquitectura.

La Tabla 8 muestra los detalles de ponderación para la selección del software tomando en cuenta los requerimientos expuestos en la Tabla 6 correspondiente a los requisitos del software y tomando en cuenta las características de los simuladores expuestos en el Anexo 1.

Tabla 8

Selección del software de simulación

Software de Simulación	**Requisitos**							**Total**
	SRS1	**SRS2**	**SRS3**	**SRS4**	**SRS5**	**SRS6**	**SRS7**	
Cooja	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**5**
QualNet	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**5**
CupCarbon	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**5**
OMNeT++	**1**	**1**	**—**	**—**	**1**	**1**	**1**	**5**
NS-3	**1**	**1**	**—**	**—**	**1**	**1**	**1**	**5**
Mininet-WiFi	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**7**

FIT IoT-LAB	**1**	**—**	**—**	**1**	**0**	**0**	**3**

1 = Cumple, 0 = No Cumple, — = Cumplimiento Parcial

De acuerdo con los resultados obtenidos en la Tabla 8, se selecciona Mininet-WiFi como software de simulación para el esquema de detección del ataque, cumpliendo los requerimientos impuestos de entre las diversas opciones de software.

En base a los parámetros establecidos se instituye como software de simulación principal a Mininet-WiFi, al ofrecer los requerimientos necesarios de implementación además de que incluye componentes de comunicación específicos para el apartado de los sensores y la compatibilidad inherente con el lenguaje de programación en el que se desarrollara el modelo NLP.

3.4.1.1 Requerimientos de hardware para la simulación

Posterior al análisis del software se realiza una revisión del apartado de hardware del equipo anfitrión, que a priori debe cumplir los requerimientos mínimos que especifica el software de Mininet-WiFi, para que el sistema a desarrollar funcione de manera adecuada. Como consideración previa se debe tomar en cuenta que el equipo debe contar una interfaz de red compatible con “mac80211_hwsim” el cual solo se encuentra en Linux al ser un módulo propio de su kernel. Bajo las consideraciones mencionadas se selecciona como sistema operativo anfitrión la distribución de Ubuntu 20.04, de esta manera se suple los requerimientos necesarios para la instalación del software Mininet-WiFi.

La Tabla 9 expone las especificaciones del equipo en el cual se llevará a cabo la simulación del escenario de red y los procesos de detección de phishing, además de esto se muestran las especificaciones mínimas que requiere el software de simulación. De esta

manera se puede avalar que el equipo suple las especificaciones mínimas que requiere el software de simulación.

Tabla 9

Verificación de especificaciones mínimas

Especificaciones del Equipo	
Procesador	AMD Ryzen 5 5600H with Radeon Graphics @ 3.30 GHz
Memoria RAM	SK Hynix 16.0 GB
Almacenamiento	Disco de estado sólido Western Digital 250 GB
Tarjeta Gráfica	Nvidia GeForce RTX 3060 Laptop GPU
Tarjeta de Red Inalámbrica	Intel(R) Wi-Fi 6 AX200 160 MHz
Sistema Operativo	Ubuntu 20.04 de 64 bits
Requerimientos de Mininet-WiFi	
Procesador	Procesador de doble núcleo o superior
Memoria RAM	Mínimo 2 GB, recomendable 4 GB o más
Espacio en Disco	10 GB de espacio libre en disco
Tarjeta de Red	Tarjeta de red compatible con el sistema operativo.
Sistema Operativo	Ubuntu 16.04 LTS o posterior

3.4.2 Selección del Gestor de Base de Datos.

La presencia o la gestión de varios dispositivos puede precisar también de una disponibilidad de información robusta, por lo cual el almacenaje de datos también desempeña un rol importante en cualquier ámbito de control. Una red IoT no se desvincula de ese parámetro por lo que la selección idónea del gestor permitirá un desempeño fluido del sistema, además pondrá a disposición todo el conjunto de datos

alojado para su uso en los procesos de generación de gráficas históricas y análisis de eventos.

La Tabla 10 muestra el gestor que más se adecua tomando en cuenta los requerimientos del sistema, así como de arquitectura presentada en las Tabla 4 y Tabla 5 respectivamente.

Tabla 10

Selección del gestor de base de datos

Gestor de Base de Datos	Requerimientos			Total
	SyRS3	SyRS9	SyRA3	
MongoDB	0	1	1	2
InfluxDB	1	1	1	3
PostgreSQL	0	1	1	2
1 = Cumple, 0 = No Cumple				

La ponderación del gestor de base de datos arroja como la opción óptima a InfluxDB, la selección radica en torno a la característica de manejo de datos relacionados con IoT, en este caso InfluxDB en el contexto de IoT tiene la capacidad de manejar la información en función de marcas temporales que sumado al servicio Network Time Protocol (NTP) generan unas bases de datos muy compactas. Además de que cuenta con una versión que opera en la nube, la cual permite extender el alcance de la disponibilidad de información.

3.4.3 Selección del Software de Visualización de Datos

En una red IoT, caracterizada por la presencia de múltiples dispositivos que generan información de manera continua, la visualización de datos adquiere una

importancia crucial. Este tipo de herramienta facilita interpretación patrones complejos, la identificación de tendencias y detección de anomalías que pueden pasar desapercibidas en un conjunto de datos en bruto. Un software de visualización de datos permite a obtener una visión más holística de la red, proporcionando una clara representación del flujo de información y el estado de los dispositivos. Consecuentemente, esta capacidad de visualización no solo mejora la comprensión del sistema, sino que además favorece y aligera la toma de decisiones.

La Tabla 11 muestra los criterios de selección del software de visualización de datos haciendo énfasis en los requerimientos del sistema y arquitectura presentes en la Tabla 4 y Tabla 5.

Tabla 11

Selección del software de visualización de datos

Software de Visualización de Datos	Requerimientos				Total
	SyRS3	**SyRS4**	**SyRS9**	**SyRA3**	
Grafana	1	1	1	1	4
Kibana	1	X	X	1	2
Zabbix	X	1	1	1	3

1 = Cumple, 0 = No Cumple, X = Cumple con limitaciones

La ponderación realizada señala como opción más adecuada al software de Grafana, la cual es muy compatible con el gestor de base de datos de InfluxDB, además de que cuenta con una versión en la nube mediante el cual es posible monitorizar el correcto funcionamiento del sistema.

3.4.4 Selección del Sistema de Detección de Intrusos

La incorporación de un Sistema de detección de Intrusos (IDS) en el ámbito de la seguridad tiene un rol crucial, especialmente en entornos donde los dispositivos poseen una limitada capacidad computacional. La integración de este tipo de soluciones suma capas de seguridad adicionales a estos ecosistemas vulnerables, mejorando no solo el comportamiento de la red dada a la carencia de mecanismo de seguridad robustos integrado en los propios dispositivos, sino que también fortalecen la defensa contra los ataques comúnmente dirigidos a redes IoT.

En base a lo expuesto se realiza una selección de un IDS que se adecue a la solución propuesta y estén alineados a los requerimientos tanto de arquitectura, software y sistema.

La Tabla 12 muestra una ponderación para la selección del Sistema de Detección de Intrusos (IDS) en base a los requisitos que se exponen en la Tabla 4 y Tabla 5 correspondiente a los apartados del sistema, arquitectura y software.

Tabla 12

Selección del Sistema de Detección de Intrusos (IDS)

Sistema de Detección de Intrusos (IDS)	**Requisitos**					**Total**
	SyRS3	**SyRS5**	**SyRA2**	**SRS1**	**SRS4**	
Suricata	1	1	1	1	1	5
Snort	1	0	1	1	0	3
OSSEC	0	0	0	1	0	1
1 = Cumple, 0 = No Cumple						

3.4.5 Selección del Modelo NLP

Para fundamentar la selección del modelo, se suma a los requisitos una revisión de trabajos en el ámbito de la ciberseguridad. Estos trabajos emplean modelos basados en NLP para llevar a cabo tareas de detección. De esta manera se podrá establecer una base coherente para la selección del modelo que va de acorde con la propuesta del sistema.

La Tabla 13 muestra una síntesis de los trabajos revisados, cuyos detalles más extensos se encuentran en el Anexo 3. Esta recopilación abarca varias investigaciones en los cuales se prueban mecanismos de análisis basados en inteligencia artificial específicamente aquellas que emplean técnicas de procesamiento de lenguaje natural.

Tabla 13

Resumen de Trabajos

	Trabajo 1
Título	CyBERT: Cybersecurity Claim Classification by Fine-Tuning the BERT Language Model
Autores	Kimia Ameri, Michael Hempel, Hamid Sharif, Juan Lopez Jr. & Kalyan Perumalla
Objetivo	Clasificar reclamos de eventos de ciberseguridad mediante el modelo BERT
	Trabajo 2
Título	SecureBERT: A Domain-Specific Language Model for Cybersecurity
Autores	Ehsan Aghaei, Xi Niu, Waseem Shadid & Ehab Al-Shaer
Objetivo	Detectar texto y frases que guardan relación con tareas de manipulación en el contexto de la ciberseguridad
	Trabajo 3

Título	Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IioT Devices
Autores	Mohamed Amine Ferrag, Mthandazo Ndhlovu, Norbert Tihanyi, Lucas C. Cordeiro, Merouane Debbah, Thierry Lestable & Narinderjit Singh Thandi
Objetivo	Identificar amenazas de seguridad en redes IoT mediante el modelo BERT

Trabajo 4

Título	MalBERTv2: Code Aware BERT-Based Model for Malware Identification
Autores	Abir Rahali & Moulay A. Akhloufi
Objetivo	Detectar malware en el código fuente de aplicaciones mediante el uso del modelo de lenguaje de gran tamaño BERT.

Trabajo 5

Título	Identification of Cybersecurity Specific Content Using the Doc2Vec Language Model
Autores	Otgonpurev Mendsaikhan, Hirokazu Hasegawa, Yukiko Yamaguchi & Hajime Shimada
Objetivo	Extraer información de las amenazas cibernéticas mediante filtros basados en Doc2Vec

Trabajo 6

Título	Beyond Word-Based Model Embeddings: Contextualized Representations for Enhanced Social Media Spam Detection
Autores	Sawsan Alshattnawi, Amani Shatnawi, Anas M.R. AlSobeh & Aws A. Magableh
Objetivo	Detectar spam en plataformas de redes sociales mediante diversos modelos de NLP

Trabajo 7

Título	Detecting Phishing Sites Using ChatGPT
Autores	Takashi Koide, Naoki Fukushi, Hiroki Nakano & Daiki Chiba
Objetivo	Analizar y determinar mediante GPT si el contenido de un sitio web es fraudulento

Trabajo 8	
Título	KARTAL: Web Application Vulnerability Hunting Using Large Language Models
Autores	Sinan Sakaoglu
Objetivo	Detectar vulnerabilidades en aplicaciones web mediante el uso de GPT

Trabajo 9	
Título	BERT-Based Models for Phishing Detection
Autores	Milita Songailaitė, Eglė Kankevičiūtė, Bohdan Zhyhun & Justina Mandravickaitė
Objetivo	Detectar correos electrónicos phishing por medio de la afinación de modelos basados en BERT y la metodología de transferencia de aprendizaje

Una vez planteado y expuesto los requisitos, se selecciona el modelo NLP, el proceso en cuestión se realiza mediante la ponderación.

La Tabla 14 muestra las consideraciones de selección del modelo basándose en un proceso de puntuación que toma en cuenta los requisitos expuestos en la Tabla 7.

Tabla 14

Selección del Modelo NLP

Modelos	**Requisitos**							**Total**
NLP	**MRS1**	**MRS2**	**MRS3**	**MRS4**	**MRS5**	**MRS6**	**MRS7**	

Word2Vec	0	1	0	0	1	0	1	2
Doc2Vec	0	1	0	0	1	0	1	2
BERT	1	1	1	1	1	1	0	6
DistilBERT	1	1	1	1	1	1	1	7
ELMo	0	1	0	1	1	0	0	3
GPT	1	1	0	1	0	1	0	4

1 = Cumple, 0 = No Cumple

Una vez estudiado los trabajos que emplean modelos de NLP en el proceso de detección de amenazas, vulnerabilidades y componentes que hacen referencia a phishing, se selecciona DistillBERT como modelo de NLP aplicables en la red IoT, debido a que posee un contexto de evaluación bidireccional, que permite una mayor compresión del lenguaje y a su vez repercute de manera favorable en la precisión del modelo, además de que al ser una versión más ligera del modelo BERT este puede implementarse en dispositivos de menor capacidad computacional sin sacrificar la precisión.

3.4.6 Selección del Ataque

Este apartado establece la relación entre el sistema de detección y los ataques de phishing. De esta manera se presentan algunos de los ataques que, según su naturaleza, metodología y objetivo, dan cabida a su uso en el proceso de diseño del sistema.

Partiendo del compendio de ataques expuestos en el capítulo II, y tomando como referencia trabajos previos en torno a la detección de phishing mediante técnicas novedosas de inteligencia artificial, se revisa las variantes de phishing más significativas. Este análisis permite establecer el tipo específico de ataque a simular en nuestra red IoT.

Considerando los estudios referenciados en el Anexo 3, se profundiza en la esencia del ataque, explorando su mecanismo de funcionamiento y las medidas de seguridad tradicionales que se pueden aplicar. Esta comprensión del ataque facilita el desarrollo de un sistema de detección más robusto y eficaz.

El enfoque empleado no solo facilita una caracterización más exacta del ataque, sino que también establece las bases para crear contramedidas innovadoras que aprovechen las capacidades de la inteligencia artificial.

La Tabla 15 describe las características generales de algunos tipos de ataque phishing que se consideran a implementar.

Tabla 15

Mecanismos de funcionamiento y seguridad de los ataques phishing

Ataque Phishing	**Tipo de Ataque**	**Mecanismo de Funcionamiento**	**Medidas de Seguridad**
Phishing mediante email	Engañoso	Envía correos electrónicos fraudulentos que pretenden ser de entes confiables con el propósito de robar información confidencial.	• Emplear filtros antispam y anti-phishing. • Verificar la dirección del remitente y la autenticidad del correo. • Evitar el acceso en enlaces de email sospechosos.
Phishing Telefónico	Engañoso	Emplea llamadas telefónicas o SMS en donde adopta la identidad de fuentes	• No proporcionar informacional personal a través de llamadas.

		confiables para notificar problemas de seguridad y de esta manera persuadir a la víctima para que revele información personal.	• Verificar la identidad del interlocutor y la entidad a la que representa.
Phishing en redes sociales	Engañoso	Emplea plataformas de redes sociales para ejecutar acciones maliciosas que conlleva en ataques como el secuestro de cuentas, suplantación de identidad, estafas y distribución de malware.	• Ajustar la configuración de privacidad. • Evitar el acceso en enlaces desconocidos. • Desconfiar de mensajes que provengan de contactos desconocidos y sospechosos.
Phishing basado en Malware	Subterfugio Técnico	Ejecuta software malicioso previamente plantado en el computador de la víctima con el objetivo de llevar a cabo acciones fraudulentas o robo de información.	• Emplear software antivirus y antimalware. • Realizar escaneos del sistema para detectar y eliminar malware.
Ataques de ofuscación URL	Subterfugio Técnico	Persuade al usuario para que acceda a enlaces que redirigen a la víctima hacia sitios y servidores maliciosos.	• Verificar la autenticidad de los URLs antes de acceder. • Configurar el navegador para mostrar alerta sobre URL sospechosas.

Estudios recientes de ciberseguridad han revelado patrones relevantes con relación a la naturaleza y frecuencia de los ataques cibernéticos. Un reporte presentado por Egress, un proveedor de seguridad de correo electrónico en la nube que emplea aprendizaje automático y redes neuronales para evaluar el riesgo humano muestra datos del presente año sobre los incidentes de ciberseguridad. El informe señala que el phishing se mantiene como la amenaza predominante para las organizaciones (Egress, 2024). Según el reporte, el 94% de las empresas encuestadas que experimentaron incidentes fueron víctimas de phishing. Entre las tácticas más comunes se sitúan:

- Phishing basado en URLs maliciosa.
- Distribución de malware o ransomware a través de archivos adjuntos.
- Ataques de phishing enviado desde cuentas comprometidas.

Es destacable mencionar que el 79% de los ataques centrados en la adquisición de credenciales de cuentas se iniciaron mediante técnicas de phishing basadas en correo electrónico, a su vez las tácticas mencionadas anteriormente denotan el uso de este mecanismo. De igual forma se menciona que el 64% de los encuestados fueron perjudicas a nivel financiero y en torno a la reputación de la empresa (Egress, 2024).

Proofpoint, una empresa de ciberseguridad especializada en la protección contra amenazas, la seguridad de correo electrónico, protección de datos y el cumplimiento normativo, presenta en su reporte de 2024 una visión porcentual más detallada sobre la prevalencia de diversos tipos de ataques cibernéticos en comparación con los años 2023 y 2022 (Proofpoint, 2024):

- Phishing masivo: 76%
- Spear phishing: 74%

- Business Email Compromise (BEC): 73%
- Ransomware: 77%
- Smishing: 75%
- Vishing: 67%
- Ataques mediante USB Drop: 60%
- Ataques a través de redes sociales: 72%
- Riesgos en la cadena de suministro: 69%
- Pérdida de datos por atacantes externos: 66%
- Pérdida de datos por insiders: 64%
- TOAD (Callback Phishing): 67%

SlashNext, empresa especializada en la protección de organizaciones contra amenazas avanzadas y ataques de ingeniería social mediante inteligencia artificial, presenta en su informe de 2023 datos acerca de los canales más empleados para el despliegue de ataques phishing (SlashNext, 2023):

- Correo electrónico: 80%
- Mensajes de texto: 28%
- Llamadas telefónicas: 23%
- Redes sociales: 14%
- Plataformas de mensajería instantánea (WhatsApp, Telegram, Signal): 14%
- Herramientas de colaboración (Slack, Microsoft Teams, Zoom): 10%

En base a la información menciona por (Egress, 2024; Proofpoint, 2024; SlashNext, 2023), se realiza la selección del ataque, tomando como criterio de ponderación el tipo de ataque que presenta mayor uso por parte del atacante.

La Tabla 16 muestra la ponderación de los ataques de phishing mencionados en los reportes expresados a manera de ranking.

Tabla 16

Selección del ataque phishing en base al ranking.

Tipo de Ataque Phishing	Reporte 1	Reporte 2	Reporte 3	Ranking promedio
Phishing basado en email	5	5	5	5
Phishing Telefónico	-	3	4	2.33
Phishing en redes sociales	-	2	3	1.66
Phishing basado en malware	3	4	-	2.33
Phishing de ofuscación URL	4	-	-	1.33
5 = nivel más alto de ranking, 1 = nivel más bajo del ranking				

Nota: La tabla de selección se elabora a partir de los datos provistos por 3 empresas: Egress (Reporte 1), Proofpoint (Reporte 2) y SlashNext (Reporte 3).

Dado la gran variedad ataques de phishing, se ha decidido por un ataque que técnicamente sea sencilla de incorporar a la red IoT generada mediante Mininet, los ataques en cuestión corresponden al ataque phishing basado en email y al ataque phishing basado en URLs que generalmente se anexan en emails. La decantación por estos ataques se fundamenta en los reportes mencionados, así como la ponderación realizada en base al ranking presentado en la Tabla 16, de esta manera se podrá poner aprueba nuevos mecanismos para contrarrestar ataques Phishing como es el caso de uso de modelos NLP.

3.4.7 Descripción del Escenario

Una vez definido los requerimientos y seleccionadas las diversas herramientas necesarias, se procede al planteamiento del escenario, el cual tiene un enfoque orientado con una arquitectura del Internet de las Cosas.

En el marco de este proyecto, se ha diseñado un escenario que comprende un conjunto de dispositivos integrados en una red IoT de tipo Smart-Home. Estos

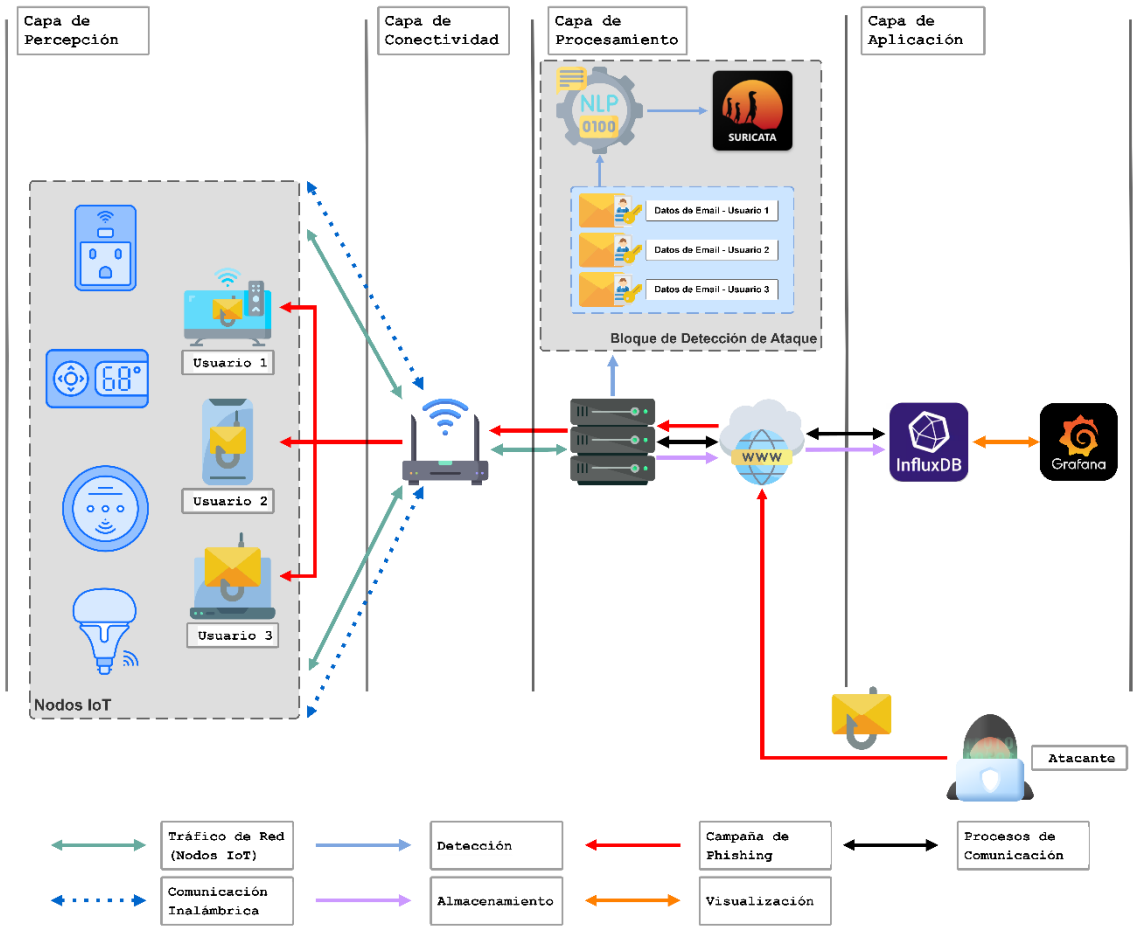
dispositivos servirán como objetivos potenciales para simular ataques de phishing, con el propósito de verificar la eficacia y la funcionalidad del sistema propuesto. El desarrollo de este escenario se estructura en dos fases: la primera centrada en la simulación de la red IoT por medio de un software especializado, y la segunda orientada al sistema de detección de phishing.

La configuración del escenario se ha diseñado tomando en cuenta el tipo de ataque phishing seleccionado previamente, así como los mecanismos y herramientas fundamentales que deben estar presentes en una red IoT.

La Figura 5 muestra la distribución de los elementos en relación con el escenario propuesto con un énfasis en una arquitectura IoT de cuatro capas y acoplada a un esquema de dirección de un ataque phishing.

Figura 5

Arquitectura IoT acoplada a un caso de ataque phishing



3.4.8 Diseño de la Red

El apartado de red del escenario propuesto se caracteriza por una serie de nodos simulados, diseñados para ejecutar procesos de comunicación y transferencia de datos. El proceso de comunicación inicia con el establecimiento del enlace WiFi entre los nodos con el punto de acceso, tras lo cual se asignan los roles de cliente y servidor a los nodos dentro de la topología de red.

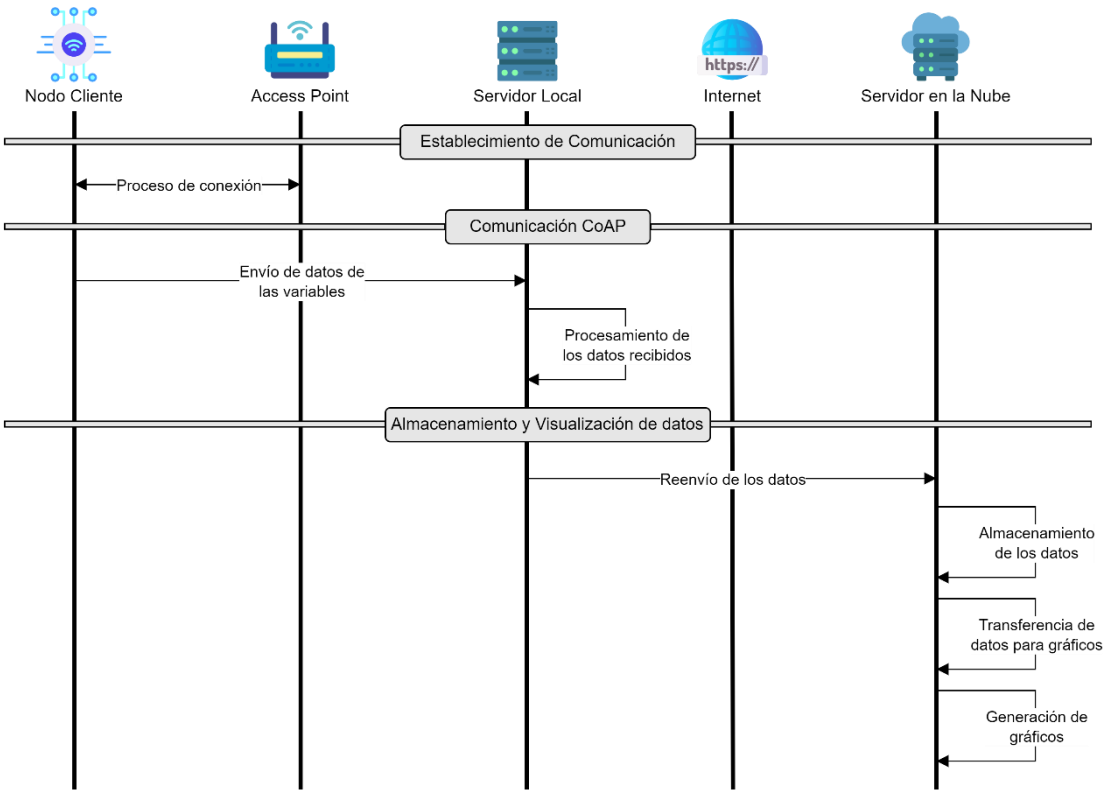
El nodo servidor, definido como servidor local, actúa como un concentrador central que gestiona el envío y recepción de datos mediante el protocolo CoAP. Una vez recibida la información, el servidor local procesa y almacena los datos en InfluxDB

Cloud, una plataforma de base de datos alojada en la nube que permite manejar datos de series temporales. Finalmente, los datos se transfieren al servicio de Grafana Cloud, la cual permite generar gráficas de monitoreo para el análisis y visualización de información.

La Figura 6 muestra la secuencia general de las interacciones realizadas por los elementos que conforman la red IoT. En el siguiente apartado, se profundizarán con mayor detalle los procesos descritos, estableciendo una relación con una arquitectura IoT de cuatro capas.

Figura 6

Diagrama de secuencia del proceso de transferencia de datos entre el servidor local y la nube



3.4.8.1 Generación de Datos

En el escenario planteado, los nodos presentes en la capa de percepción (según se indica en la Figura 5) generan los datos en función de los eventos que se suscitan en los dispositivos. Estos dispositivos se comunican por medio de WiFi, y para el presente estudio se ha planteado la simulación de múltiples dispositivos característicos de una red IoT del tipo Smart Home.

Los nodos generarán las variables relacionadas con los factores ambientales y el comportamiento del dispositivo en sí. La simulación se llevará a cabo por medio del software de simulación de red Mininet-WiFi.

La Tabla 17 presenta un listado de dispositivos típicos en una red Smart Home, así como información relevante acerca de las variables a registrar por cada nodo.

Tabla 17

Distribución de nodos presente en la capa de percepción

Sección	Nodo/Dispositivo	Variables de registro
Sala	Bombilla Inteligente	Brillo, Estado, Color
	Termostato	Temperatura, Estado
Cocina	Foco Inteligente	Brillo, Estado, Color
Dormitorio	Foco Inteligente	Brillo, Estado, Color
Entrada Principal	Enchufe Inteligente	Estado (Encendido)
Garaje	Enchufe Inteligente	Estado (Encendido)
	Sensor	Temperatura

Adicionalmente, en la Tabla 18 se especifica el direccionamiento IP de cada uno de los dispositivos, la cual refleja su distribución en la topología lógica que se representara en la simulación de red propuesta.

Tabla 18

Distribución lógica de los nodos en la capa de percepción

Sección	Nodo	Dirección IP
Sala	Bombilla Inteligente	10.0.0.11/24
	Termostato	10.0.0.12/24
Cocina	Foco Inteligente	10.0.0.13/24
Dormitorio	Foco Inteligente	10.0.0.14/24
Entrada Principal	Enchufe Inteligente	10.0.0.15/24
Garaje	Enchufe Inteligente	10.0.0.16/24
	Sensor	10.0.0.17/24

3.4.8.2 Registro de datos

El siguiente apartado se enfoca en la transmisión de los datos generados por los nodos utilizando comunicación WiFi y el protocolo CoAP. A través del software de simulación de red Mininet-WiFi, se emula el plano de conectividad de la arquitectura IoT mediante la creación de dispositivos virtuales, como switchs y puntos de acceso, que interconectar los distintos elementos de la red.

Los procesos de gestión y control de la red se realizan por medio del protocolo OpenFlow, que opera de manera inherente en el software de Mininet-WiFi, esto permite separar los planos de control y de datos, lo cual facilita la gestión de la red de manera significativa.

La Tabla 19 presenta los detalles de los elementos principales que intervienen en la transferencia de información y que hacen alusión a la capa de conectividad de la arquitectura IoT.

Tabla 19

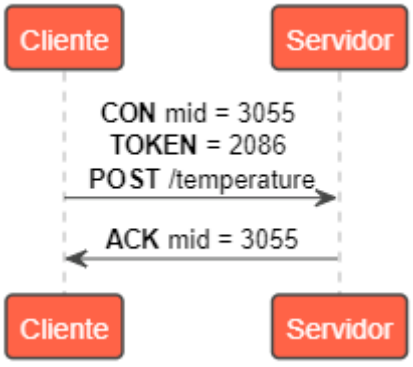
Distribución lógica de los nodos de la capa de conectiva y procesamiento

Nodo	Dirección IP
Servidor Local	10.0.0.1/24
Punto de Acceso	10.0.0.1/24

Para el apartado de la comunicación mediante el protocolo CoAP, se establece el contexto de cliente en cada uno de los nodos de la capa de percepción para el envío de datos al servidor local (ver Figura 7). Los procesos de transferencia de datos por medio de CoAP se implementará mediante la librería aiocoap del lenguaje de programación Python, que a su vez es compatible con la simulación generada por Mininet-WiFi. El servidor local actúa como un concentrador que recopila la información y la encamina hacia los planos superiores de la arquitectura IoT.

Figura 7

Proceso de comunicación del protocolo CoAP



3.4.8.3 Procesamiento de datos

Esta sección establece los mecanismos de preprocesamiento que ejecutan los procesos de decodificación y el análisis del payload (carga útil) asociados al protocolo CoAP de cada uno de los nodos de la red, facilitando el envío de información a la nube para la persistencia de los datos. Una vez determinado el contenido del payload, se envía las etiquetas y campos de información relevantes hacia el plano de aplicación de la arquitectura IoT, que en este caso agrupa los sistemas de almacenamiento de datos de InfluxDB Cloud y Grafana Cloud. Los procesos de envío hacia InfluxDB Cloud se realizan por medio de su API, mientras que para Grafana Cloud los datos se ponen a disposición a través de peticiones realizadas al servicio de almacenamiento.

3.4.9 Diseño del Sistema de Detección

La siguiente fase de diseño explora los procesos involucrados en la solución a implementar, de manera general se expone en la Figura 8 las interacciones ejecutadas por los distintos elementos durante el método de detección de correo phishing, el cual parte posterior al envío del correo desde la fuente externa hacia el objetivo dentro de la red.

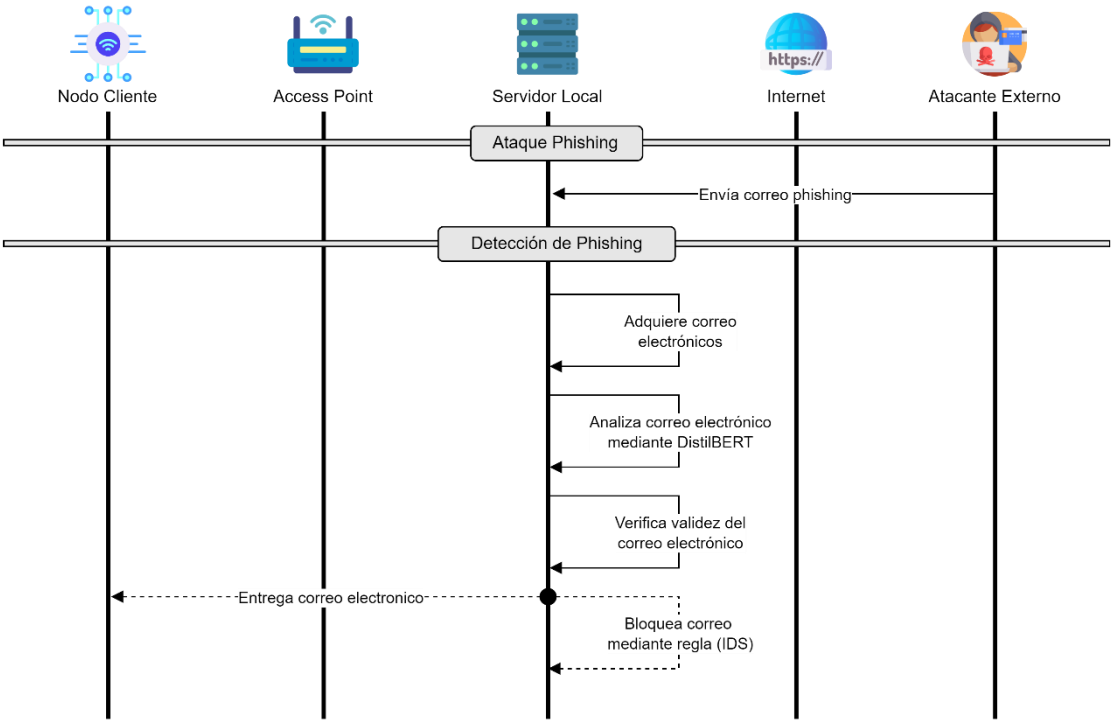
El sistema, alojado en el servido local, actúa como un proxy para interceptar los correos de los clientes de la red. Una vez receptado el correo, se analiza su contenido mediante técnicas de preprocesamiento y el modelo DistilBERT, de esta manera se logra determinar patrones inusuales en los distintos campos de un correo. El análisis se centra en información clave tales como URLs, el remitente, y el texto del email.

Posteriormente, se aplican procedimientos de clasificación para detectar si corresponde a un ataque phishing. Si se presenta un caso de phishing, se aplican

mecanismos para evitar el acceso al correo; caso contrario, se procede a la entrega del correo al cliente como uno legítimo.

Figura 8

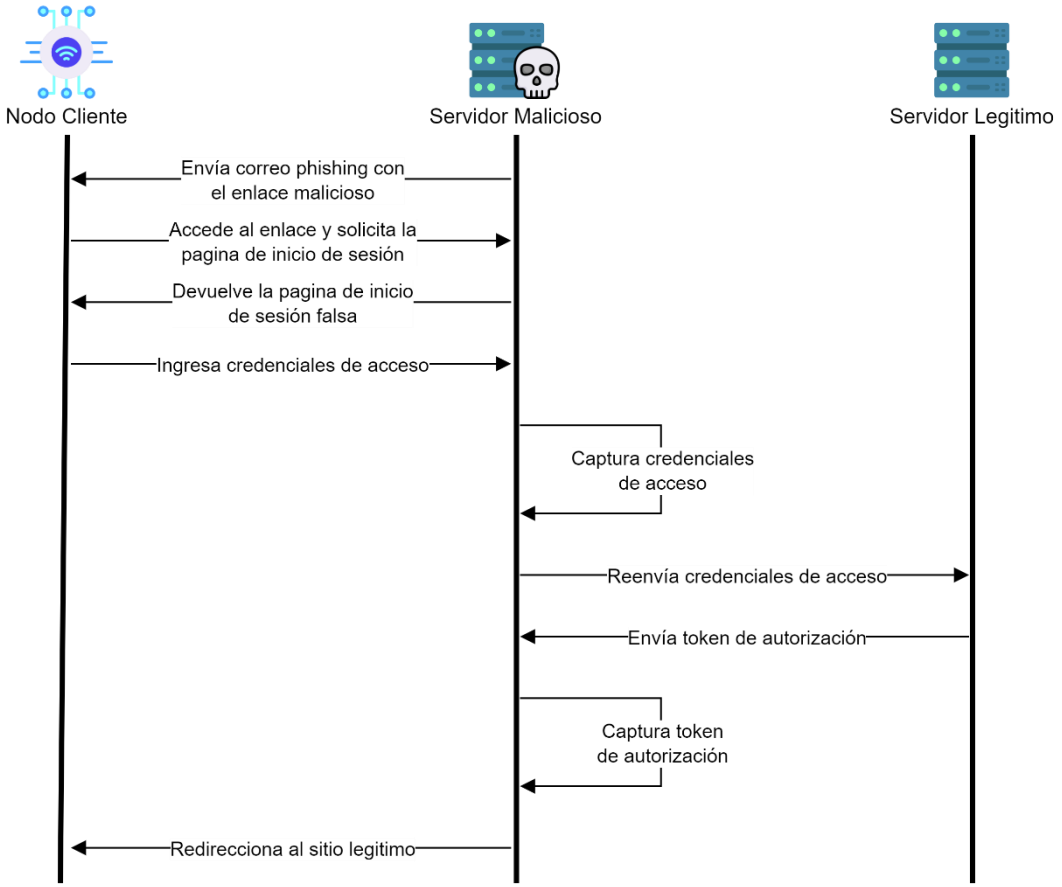
Diagrama de secuencia del proceso de detección de phishing



El proceso de adquisición de credenciales producto de un ataque phishing efectivo involucra varias interacciones, las cuales se ilustran en la Figura 9. El servidor malicioso, bajo el control del atacante, integra herramientas como Gophish y Evilginx, las cuales operan en base a la estructura de comunicación establecida por los phishlets. Estos componentes tienen como función la redirección de la víctima hacia al sitio web malicioso y consecuentemente la adquisición de las credenciales de acceso, así como los datos relacionados con las autorizaciones de sesión.

Figura 9

Diagrama de secuencia del proceso de adquisición de credenciales

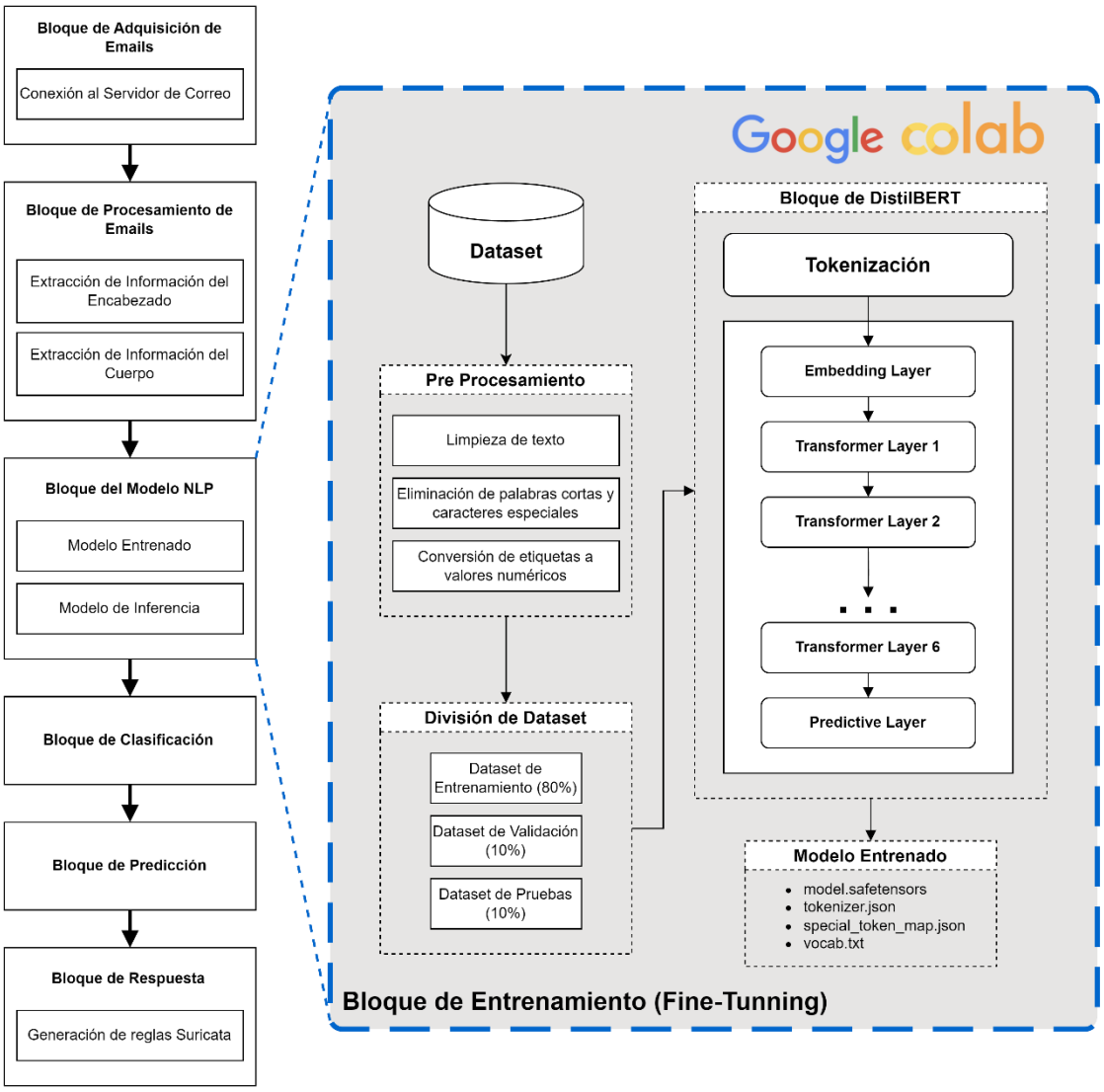


3.4.9.1 Método de detección

Para ampliar el modo de funcionamiento del sistema de detección mediante NLP presente la Figura 5, se expone mediante la Figura 10 los procesos internos que se ejecutan para determinar si el contenido de los correos electrónicos presenta detalles contextuales que hacen referencia a ataques phishing, para de esta manera efectuar las medidas pertinentes para contrarrestar el ataque.

Figura 10

Diagrama de bloques del método de detección de ataques basado en NLP



El diagrama está conformado por los siguientes bloques: bloque de adquisición de emails, bloque de procesamiento de emails, bloque del modelo NLP, bloque de clasificación, bloque de predicción, bloque de respuesta. A continuación, se define las acciones de cada uno.

- Bloque de adquisición de emails: En este apartado se obtiene los correos electrónicos de los nodos clientes haciendo uso de librerías de Python que permiten interactuar y ejecutar acciones por medio de lMAP o la API de Outlook.

- Bloque de procesamiento de emails: Define los procesos para extraer información específica de los distintos campos que conforman un email, los tipos de información a extraer comprenden direcciones URL, dominios web y texto plano del mensaje.
- Bloque del modelo NLP: El bloque en cuestión contiene los detalles relacionados al modelado mediante el proceso de ajuste fino (fine tunning) el cual se realiza en el servicio de Google Colab, por medio de un conjunto de datos se realiza la adecuación del modelo LLM para la clasificación de información a través de las etiquetas de “Safe Email” y “Phishing Email”.
- Bloque de clasificación: El modelo preentrenado se incorpora a los procesos presentes en el código del servidor local
- Bloque de predicción: Ejecuta los procesos de cálculo de porcentajes para determinar si el correo analizado es un correo phishing:
- Bloque de respuesta: Establece las reglas de bloqueo basándose en los parámetros analizados (URL, dominios web)

3.4.9.2 Proceso de Entrenamiento

En el siguiente apartado se cubre los procedimientos en torno a la preparación del modelo para la detección de phishing, en base a las secuencias especificadas en el bloque de entrenamiento de la Figura 10. El bloque en cuestión muestra los principales mecanismos del proceso de entrenamiento que corresponden al preprocesamiento de datos, segmentación del conjunto de datos, afinamiento del modelo DistilBERT con un enfoque hacia la detección de correos phishing.

Preprocesamiento de datos: Este mecanismo se encarga de eliminar caracteres innecesarios sin alterar la idea principal, entre dichos caracteres se encuentran

emoticones, tabulaciones y saltos de línea. Además de esto también se ejecutan correcciones sintácticas y de ser necesario se realiza la traducción de términos.

Segmentación del conjunto de datos: Corresponde a la distribución de los correos presentes en el conjunto de datos en partes porcentuales que se harán uso en el proceso de ajuste del modelo, así como para los casos de inferencia una vez obtenido el modelo resultante.

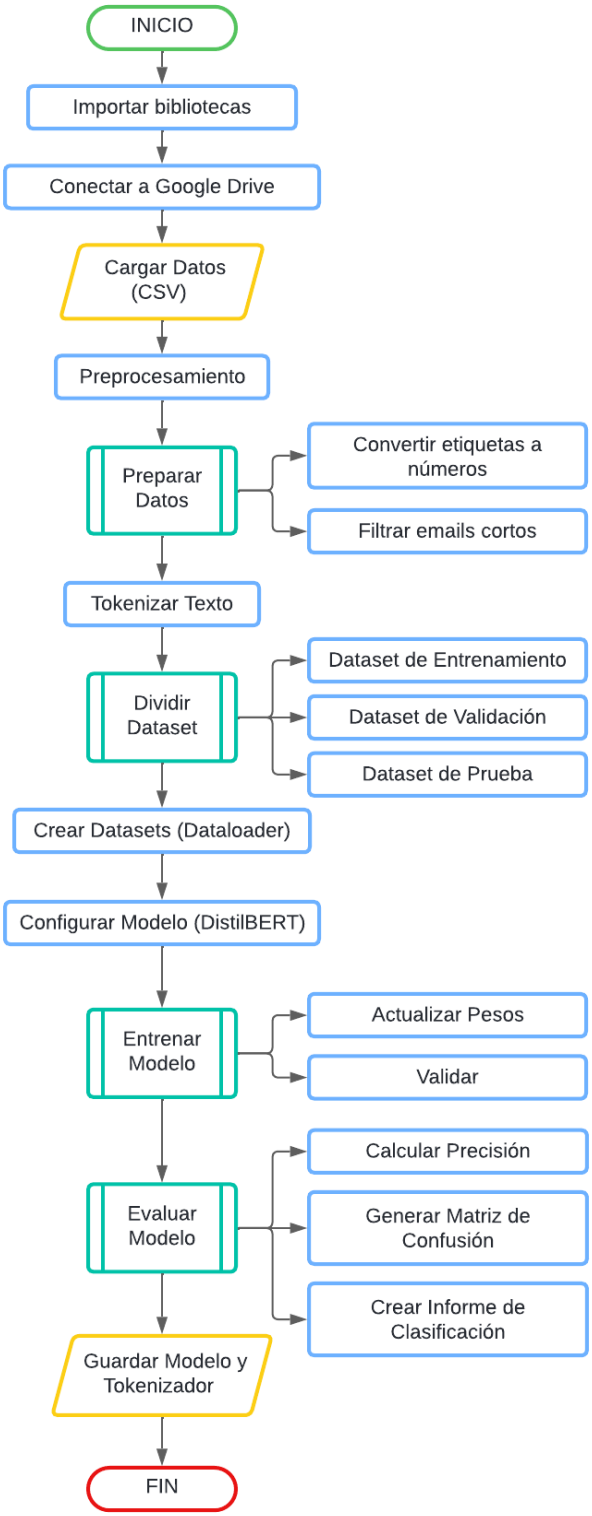
Afinamiento del modelo NLP: En este apartado se llevan a cabos las tareas de lematización, el etiquetado de partes del discurso (POS), la tokenización y la eliminación palabras vacías (Stopwords). Con el objetivo reducir la dimensionalidad del texto por medio de la eliminación de componentes innecesarios sin comprometer la idea principal de cada texto y así finalmente obtener las características del texto.

El proceso de entrenamiento se desarrollará en Google Colab, en el entorno de entrenamiento provisto por la plataforma se llevará a cabo el ajuste del modelo base de DistilBERT. El procedimiento se enfocará directamente en la identificación y clasificación de oraciones y frases cuya estructura gramatical corresponda con las características típicas de un correo de phishing. Como resultado de este proceso, se genera un nuevo modelo especializado en realizar inferencias de clasificación de los correos electrónicos en dos categorías específicas, “Safe Email” y “Phishing Email”.

La Figura 11 muestra los procesos y acciones a ejecutar en el proceso de afinamiento del modelo (Fine-Tunning) mediante DistilBERT.

Figura 11

Diagrama de flujo del proceso de entrenamiento (Fine-tuning)



3.4.9.3 Proceso de Integración del Sistema de Detección

Una vez obtenido el modelo, se integra el proceso de inferencia al servidor local que forma parte del escenario IoT, en el proceso global referente al sistema de detección de phishing se anexan las librerías y se establecen los bloques de las funciones para la inferencia además de los archivos de configuración para la operación conjunta con el modelo y el tokenizador, así como con el vocabulario para el análisis sintáctico. Conforme a las funciones integradas se presenta el conjunto de librerías empleadas, la información de la funcionalidad que desempeña cada una de ellas se detalla en la Tabla 20.

Posteriormente, se realiza el proceso de análisis del contenido del correo, el cual parte de la adquisición de los correos asociados, del contenido presente en cada uno se extrae información relevante de las secciones del encabezado como del cuerpo, se aplica el análisis de phishing al texto y a las direcciones url mediante un tratamiento previo de dicha información de esta manera se establece un valor porcentual de la probabilidad de phishing, con base en el resultado de este último proceso y de acuerdo a la etiqueta del resultado “Safe Email” y “Phishing Email” se generan las acciones a ejecutar.

Tabla 20

Detalles de las librerías empleadas

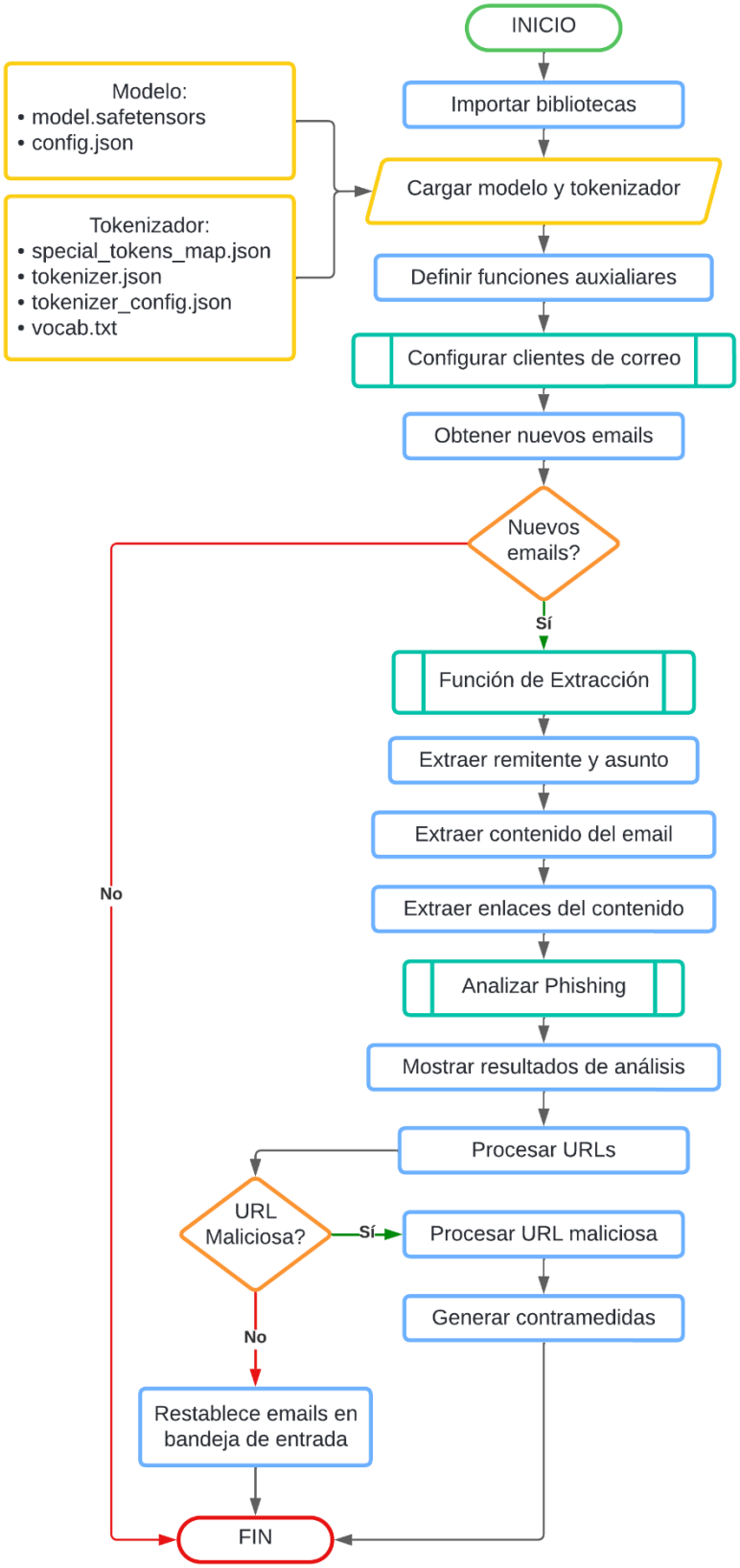
Librerías	
Nombre	**Funcionalidad**
Os	Permite la interacción y el uso de funcionalidades que dependen del sistema operativo
msal	Brinda la capacidad de manipular los encabezados del mensaje (remitente, destinatario, asunto) y el contenido de los correos electrónicos.

Transformers	Es una librería de procesamiento de lenguaje natural (NLP) que permite la interaccione con los modelos preentrenados, incluye funciones para realizar inferencias y ejecutar tareas específicas como clasificación, traducción y análisis de sentimientos.
Re	Permite realizar operaciones de búsqueda y manipulación de texto en base a patrones.
Tldextract	Librería para realizar tareas de extracción de dominios y subdominios presentes en una URL
bs4	Es una biblioteca que permite el análisis de los archivos HTML y XML, proporciona funciones para navegar, buscar y modificar datos por medio de un árbol de análisis.
Chardet	Detecta la codificación de caracteres de texto en archivos o cadenas por medio del análisis de la distribución estadística de los valores de bytes.
Collections	Proporciona un conjunto de tipos de contenedores especializados, utilizado para el almacenaje, acceso e iteración de objetos de manera más eficiente.
Subprocess	Permite la ejecución de programas externos o comandos desde Python, facilita tareas de automatización e integración de otros programas.

Un espectro más detallado de los procesos y acciones a ejecutar para el método de detección son presentados en Figura 12.

Figura 12

Diagrama de flujo del método de detección



3.4.10 Plan de Ataque

Una vez planteado el escenario de la red Smart-Home, se procede con la planificación del ataque phishing a través del correo electrónico con el propósito de adquirir información confidencial. Para este propósito, se hará uso del marco de referencia MITRE ATT&CK, que describe diferentes tipos de ataque desde el punto de vista del actor de la amenaza.

3.4.10.1 Definición del plan de ataque

El plan de ataque se definirá a partr del framework MITRE ATT&CK, el cual agrupa una extensa base de conocimientos sobre las técnicas y tácticas que son empleadas por los ciberdelincuentes. Adicionalmente, el marco ofrece mecanismos para la prevención y mitigación de amenazas.

A base de la taxonomía MITRE ATT&CK, se realiza una revisión de las técnicas y tácticas en torno al ataque phishing. El objetivo de esta acción es simular con una la mayor precisión posible, siguiendo las diversas fases que expone el marco de la MITRE ATT&CK. Las principales fases contempladas son: Reconocimiento, Desarrollo de recursos, Acceso inicial, Ejecución, Persistencia, Escalamiento de privilegios, Evasión de defensa, Acceso a credenciales, Descubrimiento, Movimiento lateral, Comando y control, Exfiltración, Impacto.

Para el caso del plan de prueba a desarrollar, se definirán y ejecutarán técnicas y sub técnicas específicas, con especial énfasis en el ciclo de vida de un ataque cibernético que señala el marco MITRE ATT&CK (Ver Tabla 21).

Tabla 21

Lista de técnicas de un ataque phishing

TAC	ID	Nombre	Descripción
Reconocimiento	T1589	Recopilar información de identidad de la víctima	Los oponentes pueden reunir datos sobre la identidad de la víctima que podrían utilizarse en la elección de objetivos. Esta información puede abarcar tanto detalles personales como datos sensibles, como credenciales o configuraciones de autenticación multifactor (MFA).
	T1590	Recopilar información de la red de víctimas	Los oponentes pueden recolectar datos sobre las redes de las víctimas que podrían utilizarse en la elección de objetivos. Esta información puede abarcar detalles administrativos (rangos de direcciones IP, nombres de dominio), así como aspectos relacionados con la topología y las operaciones.

Desarrollo de recursos	T1583	Adquirir infraestructura	Los oponentes pueden adquirir, rentar, alquilar o conseguir infraestructura que puede ser utilizada durante el proceso de segmentación. Hay un amplio catálogo de opciones de infraestructura disponibles para albergar y gestionar las operaciones de los adversarios. Entre las soluciones se encuentran servidores físicos o en la nube, dominios y servicios web proporcionados por terceros.
Acceso Inicial	T1566.002	Phishing: Enlace de spear phishing	Los oponentes pueden enviar correos electrónicos de spear phishing que contengan enlaces maliciosos con el objetivo de acceder a los sistemas de la víctima. Esta es una modalidad de spear phishing que utiliza enlaces para descargar el malware directamente desde el correo, en vez de incluirlo como archivo adjunto. Además, el spear phishing incorpora técnicas de ingeniería social y la suplantación de una entidad confiable.

Ejecución	T1598	Phishing para obtener información	Los oponentes pueden enviar mensajes de phishing con el fin de obtener datos confidenciales que podrían utilizarse en la segmentación. Esta subtécnica busca engañar a las víctimas para que revelen información, como credenciales u otros datos útiles, con el propósito principal de recopilar información más que de ejecutar código malicioso.
	T1589.001	Recopilar información de identidad de la víctima: Credenciales	Los oponentes pueden recolectar credenciales que podrían ser utilizadas en la segmentación. Estas credenciales obtenidas pueden estar directamente vinculadas con la organización objetivo o intentar aprovechar la tendencia de usar las mismas contraseñas en cuentas personales y laborales.

Persistencia y Escalada	T1078	Cuentas válidas	Los oponentes pueden obtener y explotar las credenciales de cuentas existentes como una forma de conseguir acceso inicial, mantener persistencia, escalar privilegios o evadir defensas. Estas credenciales comprometidas pueden ser utilizadas para eludir los controles de acceso a diversos recursos dentro de los sistemas de la red, o para mantener un acceso continuo a sistemas remotos y servicios externos (como VPN, Outlook Web Access, dispositivos de red y escritorio remoto). Además, las credenciales comprometidas pueden proporcionar mayores privilegios en sistemas específicos o acceso a áreas restringidas de la red, complicando así la detección de la presencia del adversario.

Acceso a Credenciales	T1539	Robar cookie de sesión web	Un atacante puede sustraer cookies de sesión de aplicaciones web o servicios y emplearlas para acceder como un usuario autenticado sin requerir credenciales. Las aplicaciones y servicios web frecuentemente usan cookies de sesión como tokens de autenticación una vez que un usuario ha iniciado sesión en un sitio web.

La estrategia de mitigación se define en consonancia con la del sistema de detección phishing propuesto, tomando como referencia los lineamientos que establece el marco de MITRE ATT&CK.

Los detalles específicos de la mitigación se muestran de manera detallada en la Tabla 22.

Tabla 22

Lista de mitigación contemplada para el ataque phishing

TAC	ID	Nombre	Descripción

Mitigación	M1031	Prevención de intrusos en la red	Los sistemas de detección de intrusiones en la red y aquellos diseñados para escanear y eliminar archivos adjuntos o enlaces maliciosos en correos electrónicos pueden utilizarse para impedir actividades no deseadas.

Reconocimiento: El desarrollo de esta fase se llevará a cabo mediante técnicas de Inteligencia de Fuentes Abiertas (OSINT), que permitirán la recolección de datos relevantes sin la intervención directa con los sistemas objetivos. Además, se desarrollarán estrategias de ingeniería social como el análisis de las redes sociales y mapeo organizacional para de esta manera construir un perfil del objetivo que serán la base de las siguientes etapas.

Desarrollo de Recursos: Esta fase implica el despliegue del ecosistema diseñado para simular y evaluar el ataque, en este sentido se emplearán herramientas especializadas para phishing como Evilginx y Gophish, que proporcionan capacidades para la ejecución de técnicas como la suplantación de identidad y captura de credenciales. Sumado a esto se configurará un dominio fraudulento para recrear un escenario más realista desde el punto de vista de la ingeniera social.

Acceso Inicial: Esta fase se centra en el desarrollo de métodos de acceso que combinan la parte técnica y la social, dentro de estos métodos se incluye la construcción de correos electrónicos personalizados y el diseño de páginas de suplantación.

Ejecución: Correspondiente a la materialización práctica de los mecanismos diseñados anteriormente y que operan con las herramientas especializadas en phishing, por medio de los señuelos y plantillas de correo buscan eludir al objetivo para que revele información confidencial.

Persistencia y Escalada: En esta fase se emplea la información del objetivo adquirida tales como usuario y contraseña asociada a los servicios legítimos de correo para de esta manera prolongar la estancia en la red sin que se genera indicios de presencia por parte del atacante.

Acceso a Credenciales: Esta fase se alinea con el resto de las fases, es la parte final en la que el adversario captura el token de identificación asociada al servicio en este caso el correo, por medio de mecanismos asociados a las cookies el atacante puede ser capaz de acceder a la cuenta del objetivo.

CAPÍTULO IV: Implementación y Pruebas

El siguiente capítulo cubre los apartados de configuración de la red y el sistema de detección que buscan mejorar la seguridad en un entorno IoT mediante el uso de inteligencia artificial junto con un sistema de detección de intrusos (IDS), a su vez se ejecutaran pruebas para determinar la factibilidad de la propuesta de seguridad. El apartado de pruebas se categoriza en dos ámbitos funcionales y específicas, en los cuales se evalúan para el primer caso las diversas secciones de la red y el sistema de manera individual, para el segundo caso se pone a prueba de manera conjunta el sistema completo en relación con los casos en donde el sistema se encuentra activo e inactivo.

4.1 Implementación y Configuración de la Red

El apartado de la red se desarrolla a base de una configuración de nodos que desempeñaran los roles de nodos clientes y nodo servidor para el caso del proceso de comunicación y envío de datos mediante CoAP, a su vez también se generan los dispositivos virtuales que actuaran como puntos de acceso o concentrador para el tema de conexión mediante WiFi y la interconexión de la red. Finalmente, se redirige los datos de la red local hacia la nube para garantizar la persistencia y disponibilidad de los datos.

4.1.1 Generación de Datos

El siguiente proceso toma como referencia el listado de dispositivos de la Tabla 17 presente en el apartado de diseño. Con base en dicha información se generan una serie de dispositivos zonas predeterminadas del escenario Smart Home, posteriormente se establecen las configuraciones de red respectiva (ver Figura 13).

Figura 13

Creación de Nodos

```
# SALA
sec1_n1 = net.addStation('sala_n1', mac='00:00:00:00:00:01', ip='10.0.0.11/24', position='170,175,0')
sec1_n2 = net.addStation('sala_n2', mac='00:00:00:00:00:02', ip='10.0.0.12/24', position='150,203,0')
host1 = net.addStation('host1', mac='00:00:00:00:00:10', ip='10.0.0.20/24', position='140,190,0')
# COCINA
sec2_n1 = net.addStation('coci_n1', mac='00:00:00:00:00:03', ip='10.0.0.13/24', position='185,185,0')
# DORMITORIO
sec3_n1 = net.addStation('dorm_n1', mac='00:00:00:00:00:04', ip='10.0.0.14/24', position='163,250,0')
# ENTRADA
sec4_n1 = net.addStation('entr_n1', mac='00:00:00:00:00:05', ip='10.0.0.15/24', position='170,115,0')
# GARAJE
sec5_n1 = net.addStation('gara_n1', mac='00:00:00:00:00:06', ip='10.0.0.16/24', position='128,170,0')
sec5_n2 = net.addStation('gara_n2', mac='00:00:00:00:00:07', ip='10.0.0.17/24', position='128,140,0')
# SERVIDOR
serv = net.addHost('serv', mac='00:00:00:00:00:08', ip='10.0.0.100/24', position='128,225,0')
serv2 = net.addHost('serv2', mac='00:00:00:00:00:09', ip='10.0.0.101/24' , position='128, 230,0')
```

A su vez se crean los scripts en Python para la simulación de las variables a monitorear y que serán enlazadas a su respectivo dispositivo. Para ampliar los detalles de los dispositivos se presenta las variables y estados a representar mediante la simulación en la Tabla 23.

Tabla 23

Datos de monitoreo de los dispositivos

Dispositivo	Variable	Estados
Sensor de temperatura	temperatura (15-25 °C)	-
Termostato	temperatura (21 – 25 °C)	Encendido/Apagado
Enchufe Inteligente	-	Encendido/Apagado
Foco inteligente	color (Hexadecimal) brillo (0-100)	Encendido/Apagado

La Figura 14 muestra la sección de código para la generación aleatoria de temperatura correspondiente al sensor y a su vez la Figura 15 muestra el código de asociación de los scripts de la interfaz con su respectivo dispositivo.

Figura 14

Código de generación aleatoria de datos de temperatura

```
async def send_data():
    protocol = await Context.create_client_context()
    while True:
        temperature = round(random.uniform(15.0, 25.0), 2)  # Simulación de datos de temperatura
```

Figura 15

Código de asociación de scripts de generación de datos

```
# SALA
makeTerm(sec1_n1, title='Sala-Foco',cmd="bash -c 'sleep 5; python3 SmartBulb.py bulb1'")
makeTerm(sec1_n2, title='Sala-Termostato',cmd="bash -c 'sleep 5; python3 Thermostat.py'")
# COCINA
makeTerm(sec2_n1, title='Cocina-Foco',cmd="bash -c 'sleep 5; python3 SmartBulb.py bulb2'")
# DORMITORIO
makeTerm(sec3_n1, title='Dorm-Foco',cmd="bash -c 'sleep 5; python3 SmartBulb.py bulb3'")
# ENTRADA
makeTerm(sec4_n1, title='Entrada-Plug',cmd="bash -c 'sleep 5; python3 SmartPlug.py plug1'")
# GARAJE
makeTerm(sec5_n1, title='Garaje-Plug',cmd="bash -c 'sleep 5; python3 SmartPlug.py plug2'")
makeTerm(sec5_n2, title='Garaje-Sensor',cmd="bash -c 'sleep 5; python3 SensorTemp.py'")
```

4.1.2 Registro de Datos

Continuando con la configuración de la red se generan los dispositivos encargados de los procesos de conexión por medio de WiFi y los procesos de transferencia de datos mediante CoAP.

El apartado de configuración para la comunicación mediante WiFi inicia a partir de la designación de identificativos, direcciones físicas (dirección MAC) y direcciones lógicas (dirección IP) se establecen las configuraciones iniciales de los nodos que actúan como estaciones en la red. Entre los aspectos de la red inalámbrica se define el Service Set Identifier (SSID) que permite identificar la Red de Área Local Inalámbrica (WLAN) para diferenciarlas de otras, el estándar de operación “g” y el canal de operación que funciona de acorde a las condiciones de la red inalámbrica, con el fin de simular una red más realista. El comportamiento de la red en Mininet-WiFi para el estándar de comunicación 802.11 se establece a través del servicio en segundo plano WMEDIUMD

mediante el cual se gestiona las características de atenuación, interferencia y la tasa de error.

La Figura 16 muestra las principales características en relación con la tecnología WiFi, además de la configuración del host que ejecutara el servidor de CoAP.

Figura 16

Creación de los nodos de la capa conectividad y procesamiento

```
# ACCESS POINT
ap1 = net.addAccessPoint('ap1', ssid='SmartHome-1', mode='g', channel='1', position='160,200,0', range='100')

# SERVIDOR
serv = net.addHost('serv', mac='00:00:00:00:00:08', ip='10.0.0.100/24', position='128,225,0')
```

Continuando con el proceso de configuración, se establecen los criterios del modelo propagación correspondiente a la de distancia logarítmica el cual señala el decremento de la intensidad que se obtendrá a medida que la estación se aleja del transmisor. Además, se establece el valor de la pérdida de 4 que de acuerdo con el modelo hace referencia a un entorno urbano denso o interiores con un gran número de obstrucciones. Una vez definidos las características del medio inalámbrico, se generan los enlaces de conexión entre los diversos elementos de la red (ver Figura 17), la cual corresponde a una topología del tipo estrella.

Figura 17

Configuración y Conexión de Enlaces de Comunicación

```
info("*** Configurando propagación del modelo de WiFi\n")
net.setPropagationModel(model="logDistance", exp=4)

info("*** Configurando nodos wifi\n")
net.configureWifiNodes()

info("*** Creando enlaces\n")
net.addLink(ap1, serv)
net.addLink(ap1, serv2)

info("*** Configurando conexiones inalámbricas\n")
net.addLink(ap1, sec1_n1)
net.addLink(ap1, sec1_n2)
net.addLink(ap1, host1)
net.addLink(ap1, sec2_n1)
net.addLink(ap1, sec3_n1)
net.addLink(ap1, sec4_n1)
net.addLink(ap1, sec5_n1)
net.addLink(ap1, sec5_n2)
```

Acto seguido se establece el dispositivo que ejecutara los procesos de Traducción de Direcciones de Red (NAT) y a la vez que actuara como puerta de enlace para la red simulada y otras redes externas. Una vez establecida la red, se configura el reenvío de paquetes entre interfaces en el punto de acceso (ap1). Posteriormente, se asignan las rutas por defecto y la dirección del Sistema de Nombres de Dominio (DNS) en todos los nodos de la red, los detalles del código para los procesos descritos anteriormente son representados en la Figura 18.

Figura 18

Inicialización de la Topología de Red

```
# Controlador
c0 = net.addController('c0')

info("*** Inicializar Red\n")
net.build()
net.addNAT(name="nat0", linkTo='ap1', ip='10.0.0.1').configDefault()

info("*** Iniciando controladores\n")
for controller in net.controllers:
    controller.start()

info("*** Iniciando puntos de acceso\n")
ap1.start([c0])

# Habilitar forwarding en el host
net.get('ap1').cmd('sysctl -w net.ipv4.ip_forward=1')
for sta in [serv, sec1_n1, host1, sec2_n1, sec3_n1, sec4_n1, sec5_n1, sec5_n2]:
    # Configurar gateway y DNS
    sta.cmd('route add default gw 10.0.0.1')
    sta.cmd('echo "nameserver   8.8.8.8" > /etc/resolv.conf')
```

En el siguiente apartado se inicializa el servicio referente al protocolo CoAP para la recepción de los datos provenientes desde la capa de percepción (ver Figura 19). Cabe destacar que una vez inicializada la red Mininet-WiFi genera un conjunto de subredes e interfaces de manera interna por lo que el método de análisis del tráfico se centrara en la interfaz ‘hwsim0’ la cual reproduce la comunicación entre las interfaces simuladas dentro de la topología de red.

Figura 19

Inicialización del Servidor Local

```
# Inicializar Servicios y Nodos
info("*** Inicializar Servicios\n")
# Servidor CoAP
makeTerm(serv, title='Nodo Servidor CoAP',cmd="bash -c 'sleep 5; python3 CoAP_Server.py'")
```

La configuración del protocolo de comunicación Constrained Application Protocol “CoAP” en nuestra red simulada se definen en función de la RFC 7252 y se rige bajo un modelo de cliente-servidor.

Para el apartado de envío de datos se establece el proceso asíncrono en el que se incluye la creación del contexto para el cliente CoAP, el cual se ejecutará siempre y cuando el valor boleano sea verdadero. Dentro del proceso interno se establece la variable a simular, para el caso de descripción se detalla la información con relación al sensor de temperatura el cual genera valores aleatorios entre un rango de 20.0 a 25.0. Posteriormente, se aplican procesos de conversión de datos en donde se transforma el valor de la temperatura a una cadena de texto, que a su vez será codificada en formato “utf-8”. Acto seguido se crea la petición por medio de la función “MESSAGE” en la cual se establecen varios parámetros como el método “POST”, el identificador del servidor “coap://10.0.0.10/temperatura” y el payload (carga útil). Finalmente, se envía la petición y se espera la respuesta del servidor.

La Figura 20 muestra los detalles asociados a la creación del contexto cliente del sensor de temperatura.

Figura 20

Código de envío de datos

```
async def send_data():
    protocol = await Context.create_client_context()
    while True:
        temperature = round(random.uniform(15.0, 25.0), 2)  # Simulación de datos de temperatura
        payload = str(temperature).encode('utf8')
        request = Message(code=POST, uri='coap://10.0.0.100/temperature', payload=payload)
        response = await protocol.request(request).response
        print(f"Envío de datos temperatura: {temperature}")
        print(f"Respuesta: {response.code}\n{response.payload.decode('utf8')}")
        await asyncio.sleep(15)  # Enviar datos cada 15 segundos
```

4.1.3 Procesamiento de Datos

El apartado del procesamiento inicia a partir de la creación y ejecución del contexto de servidor CoAP. Sin embargo, la recepción de datos se realiza por medio de una clase en la que se definen una serie de contenedores que alojan los recursos del protocolo CoAP en base a cada nodo y de acorde a la variable a receptar, de manera puntualizada para el caso del sensor de temperatura se crea el recurso la clase TemperatureSensorResource, acto seguido se define una tarea asíncrona para renderizar la publicación a partir de una petición, de esta manera se decodifica el payload de la solicitud en formato “utf8” para su posterior impresión.

La Figura 21 muestra el fragmento de código que hace referencia al proceso de creación del recurso para el contexto del servidor.

Figura 21

Creación del recurso del sensor de temperatura

```
class TemperatureSensorResource(resource.Resource):
    async def render_post(self, request):
        payload = request.payload.decode('utf8')
        print(f"Datos de Sensor de Temperatura Recibidos: {payload}")
        server.temperature_sensor = float(payload)
```

Una vez definido lo anterior se anexan los recursos de cada uno de los dispositivos para su respectivo uso, a su vez que se asignan etiquetas para la identificación en el apartado de persistencia de datos que engloba los procesos de almacenamiento y visualización de información (ver Figura 22).

Figura 22

Código de Recepción de datos

```
async def main(self):
    root = resource.Site()
    root.add_resource(['bulb'], self.BulbResource())
    root.add_resource(['thermostat'], self.ThermostatResource())
    root.add_resource(['temperature'], self.TemperatureSensorResource())
    root.add_resource(['plug'], self.PlugResource())
    await Context.create_server_context(root)
    await asyncio.get_running_loop().create_future()
```

Finalmente, se envía la información recolectada hacia el servicio de almacenamiento de InfluxDB Cloud por medio de la API de gestión que provee el servicio (ver Figura 23).

Figura 23

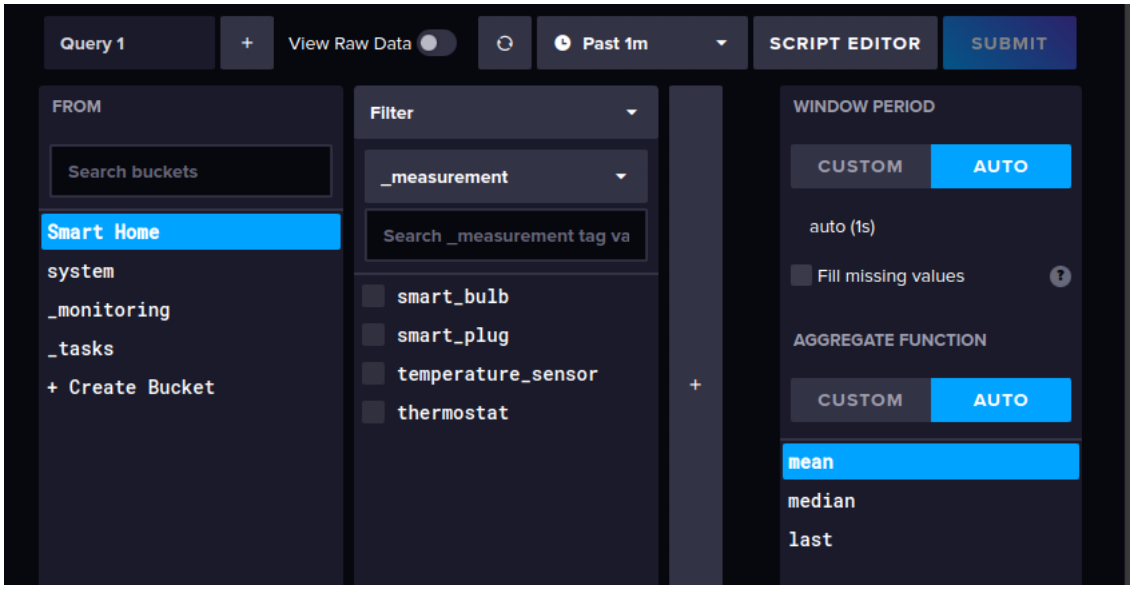
Código de almacenamiento de satos en InfluxDB Cloud

```
# Envio de Datos a InfluxDB
server.influxdb_manager.write_to_influxdb(
    measurement="temperature_sensor",
    tags={},
    fields={
        "temperature": server.temperature_sensor
    }
)
```

En el apartado de los servicios que hacen alusión a la capa de aplicación, se configuran los detalles del almacenamiento de InfluxDB Cloud en función de los parámetros asociados al bucket “Smart Home” establecido en el script de la simulación red. Una vez realizado la configuración se inicia el proceso de almacenamiento en el servicio de InfluxDB Cloud (ver Figura 24).

Figura 24

Configuración del bucket "Smart Home"



El proceso de visualización de datos en software de Grafana precisa la configuración de una query en InfluxDB Cloud, la cual se toma como base de consulta para importar los datos y realizar la representación de datos a manera de líneas de tiempo en que se suscitan los diversos cambios en este caso que el sensor de temperatura registra.

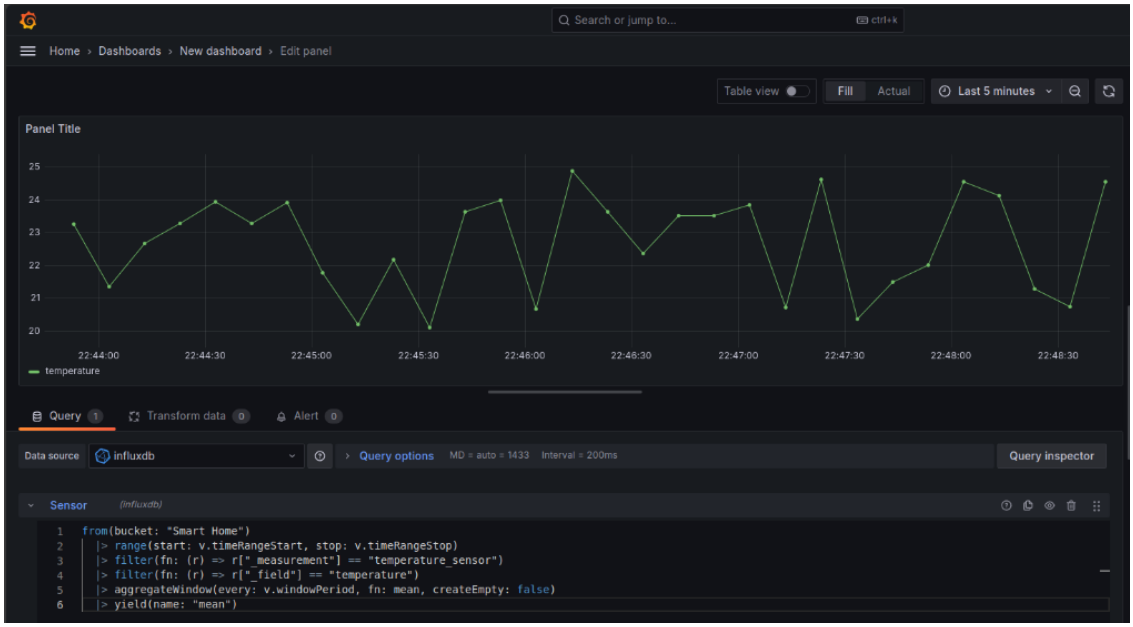
La Figura 25 y Figura 26 muestra el formato de la query de InfluxDB Cloud asociado al sensor de temperatura y la representación de los datos en Grafana Cloud en base a la query de dicho sensor respectivamente.

Figura 25

Configuración de la solicitud de información en InfluxDB Cloud

Figura 26

Generación de gráficas del sensor de temperatura



El proceso de implementación de la red, así como la instalación de los servicios y librerías necesarias para su correcto funcionamiento, se describen de manera más detallada en el Anexo 4. En dicho anexo, se presenta los procedimientos de configuración, el software empleado y las dependencias que fueron integradas en el código.

4.1.4 Configuración de recursos de la red IoT

La presente sección se desarrolla basándose en que el sistema de detección de phishing precisa de un par de herramientas para su funcionamiento, bajo esa premisa se incorpora dichas herramientas en computador anfitrión de manera que una vez se despliegue la red simulada los recursos sean accesibles y ejecutables desde cualquier nodo de la red IoT.

4.1.4.1 Instalación del Sistema de Detección de Intrusos (IDS) – Suricata

Como primer punto se añade el repositorio de paquetes de software para Suricata en su versión estable (ver Figura 27), posteriormente se actualiza la base de datos de los

paquetes en el sistema y se instala el servicio de Suricata a través de la siguiente combinación de comandos (ver Figura 28)

Figura 27

Adición del repositorio de archivos de Suricata

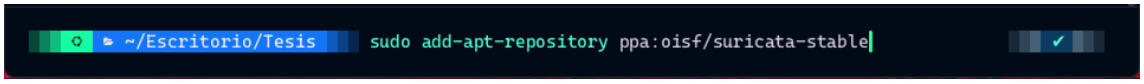


Figura 28

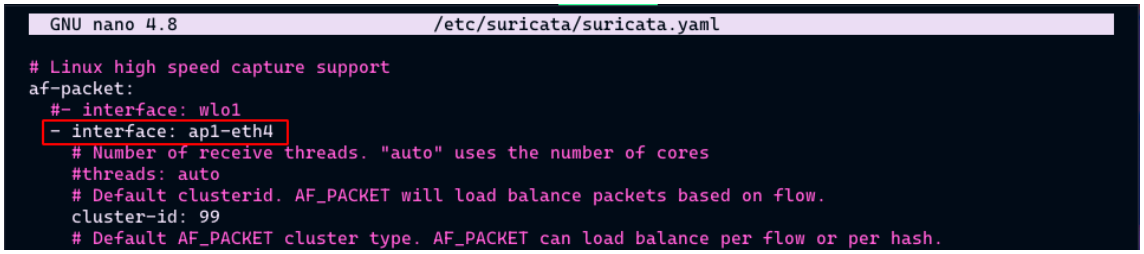
Instalación de Suricata

En la siguiente parte se configuran los parámetros de análisis de las interfaces, en este caso la interfaz a especificar corresponde a la del nodo generado en Mininet-WiFi que actúa como punto de acceso.

La Figura 29 resalta la interfaz “ap1-eth4” del nodo que actúa como punto de acceso y es a la vez punto concentrador del tráfico generado en la red IoT.

Figura 29

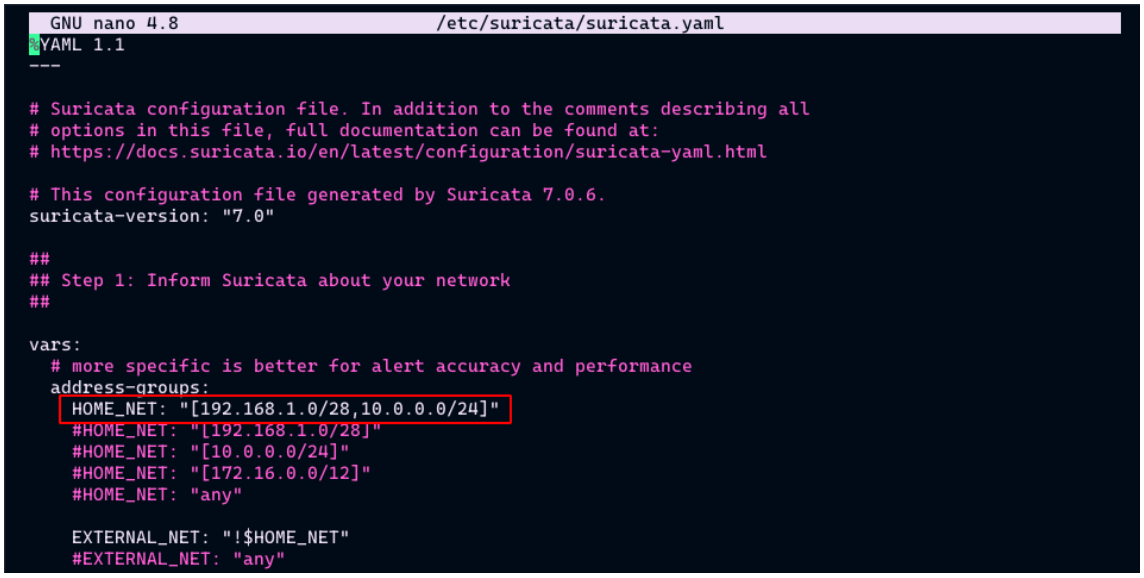
Configuración de la interfaz de análisis



De la misma manera también se configura los segmentos de red a analizar, en la sección de ***address-groups*** se adjunta la red externa que corresponde con la red en que se encuentra el computador anfitrión y la red IoT generada por Mininet-WiFi (ver Figura 30).

Figura 30

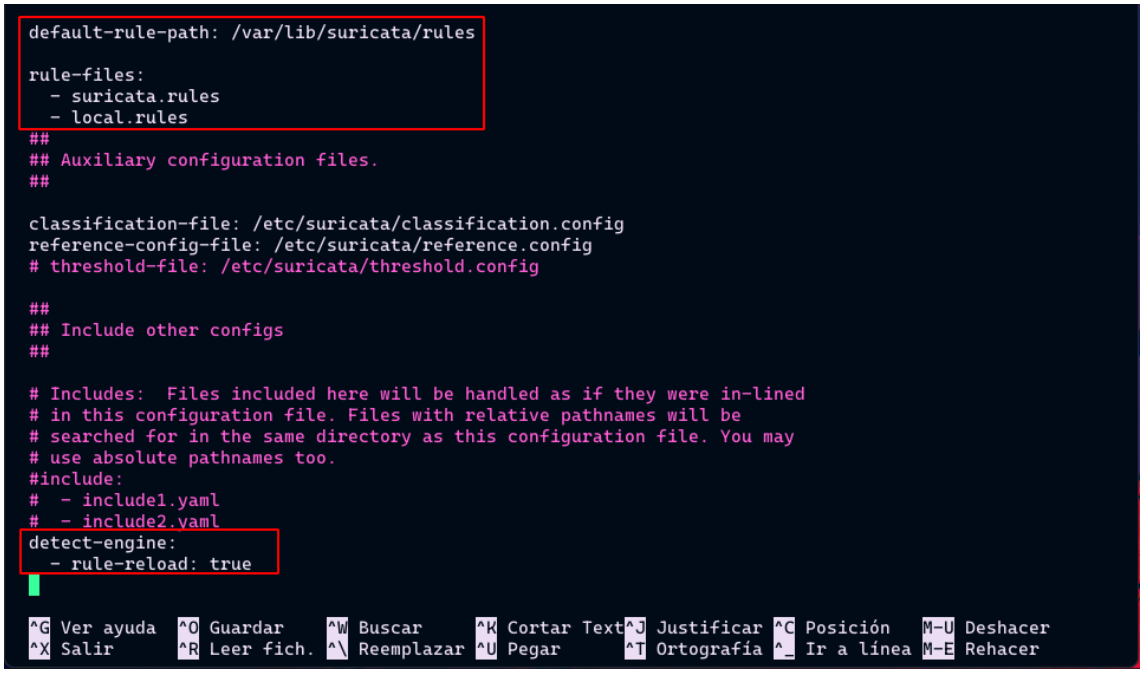
Configuración de grupos de red



Como parte final del proceso de configuración de Suricata se definen el directorio de reglas que el sistema emplea en los procesos de análisis de la red, así como las reglas que entran en acción, junto a esto también habilita la actualización de reglas (ver Figura 31).

Figura 31

Configuración de reglas

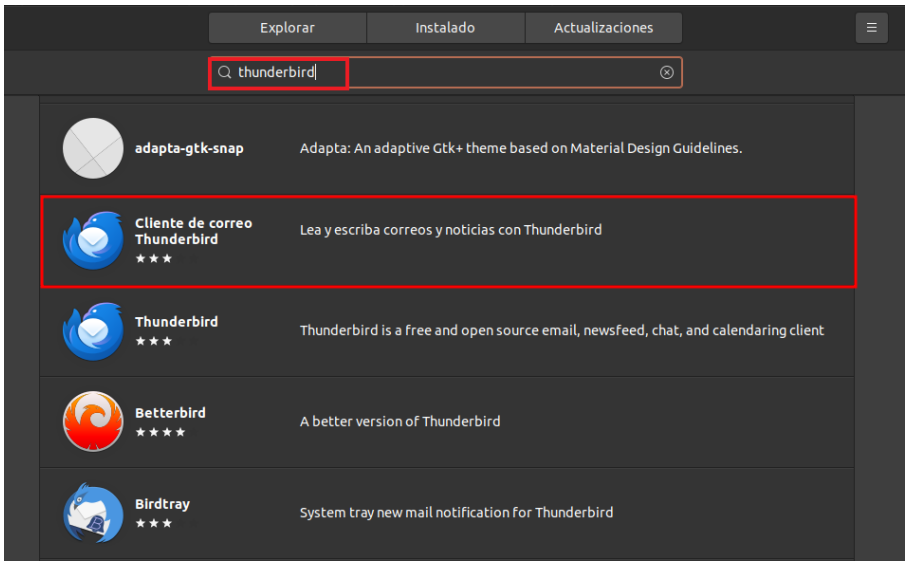


4.1.4.2 Instalación del cliente de correo – Thunderbird

En este apartado se configura el cliente de correo Thunderbird como un recurso disponible para la red IoT simulada en Mininet-WiFi. El proceso inicia accediendo a la tienda de aplicaciones de Ubuntu, por medio del buscador se identifica el software deseado “Thunderbird” (ver Figura 32).

Figura 32

Búsqueda del cliente de correo Thunderbird

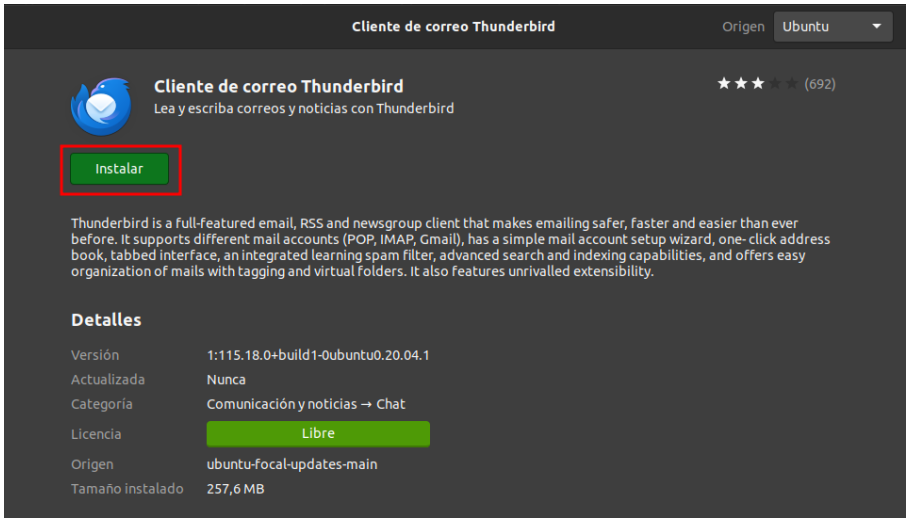


Una vez seleccionada la opción deseada se instala el software haciendo clic en el botón ***instalar*** que se muestra en la ventana de información del producto.

La Figura 33 muestra la información referente a Thunderbird, dentro de los detalles se especifica los parámetros como la versión, licencia, origen y tamaño del paquete de instalación.

Figura 33

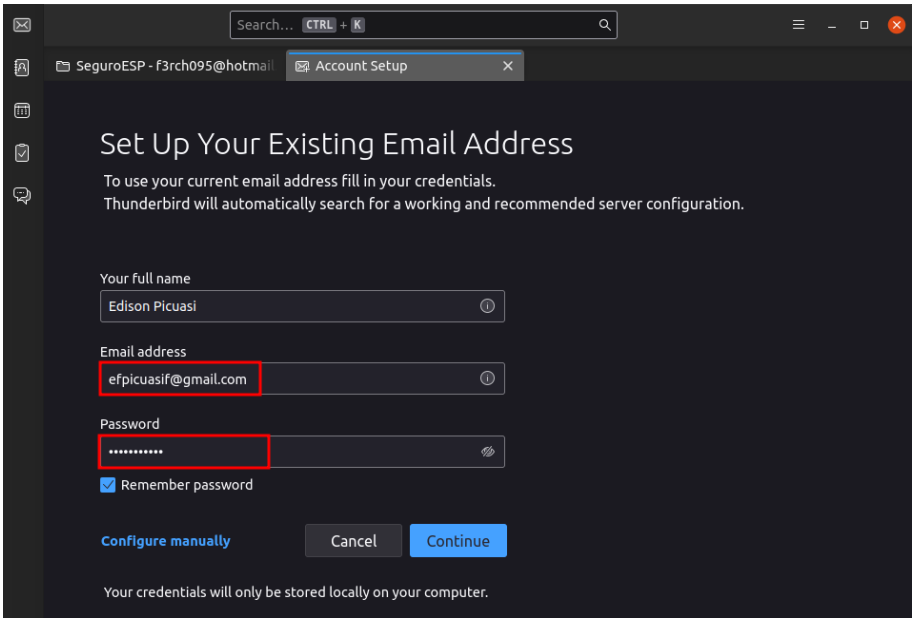
Instalación del cliente de correo



Una vez finalizado la instalación, se continúa con la configuración del correo electrónico, en la ventana desplegada, se ingresa la información de nuestra cuenta de correo (ver Figura 34). Mediante el botón de continuar situado en la parte inferior se comprueba si el correo es correcto.

Figura 34

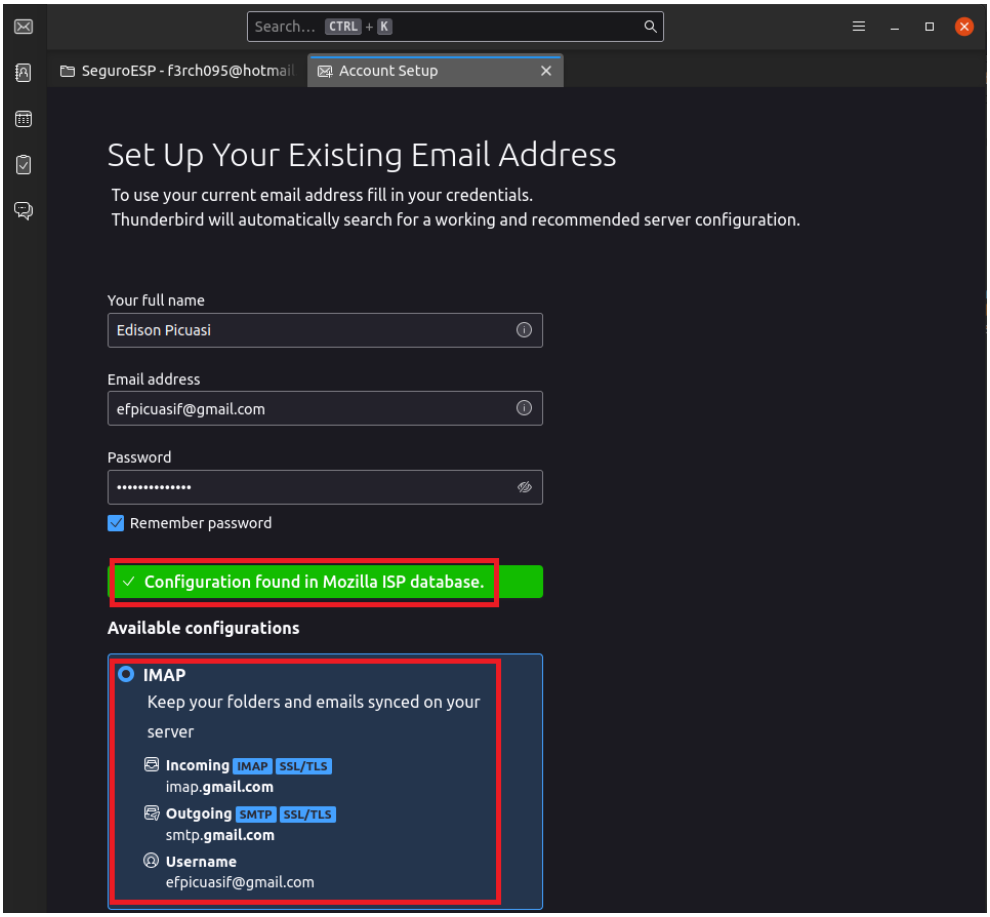
Configuración de la cuenta de correo



Una vez se concrete la verificación, se dará muestra acerca de la disponibilidad de protocolos con los que será configurada el correo en Thunderbird (ver Figura 35), en este caso se selecciona la opción IMAP junto con el resto funciones asociadas.

Figura 35

Verificación de existencia de la cuenta de correo

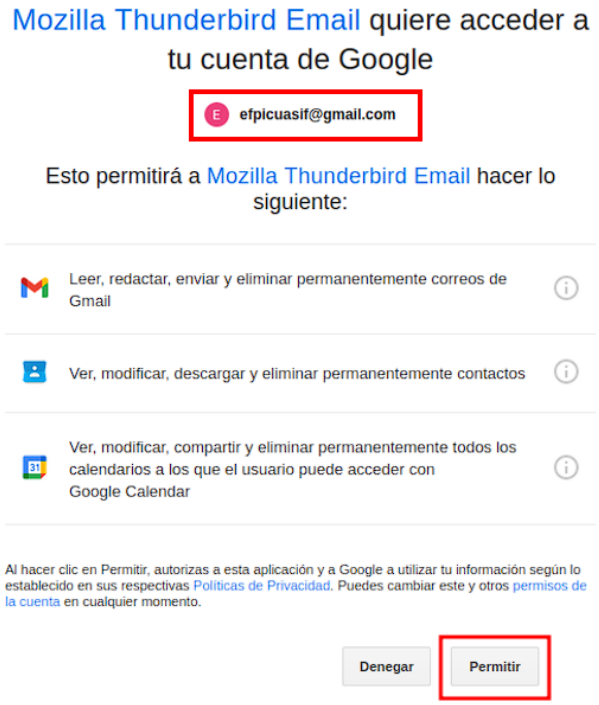


En el siguiente paso se concede los permisos de ejecución necesarios para que Thunderbird pueda gestionar sin problemas la nueva cuenta correo.

La Figura 36 muestra la petición del cliente de correo para realizar acciones de lectura, escritura, envió y eliminación en torno a las herramientas de correo, contactos y calendario.

Figura 36

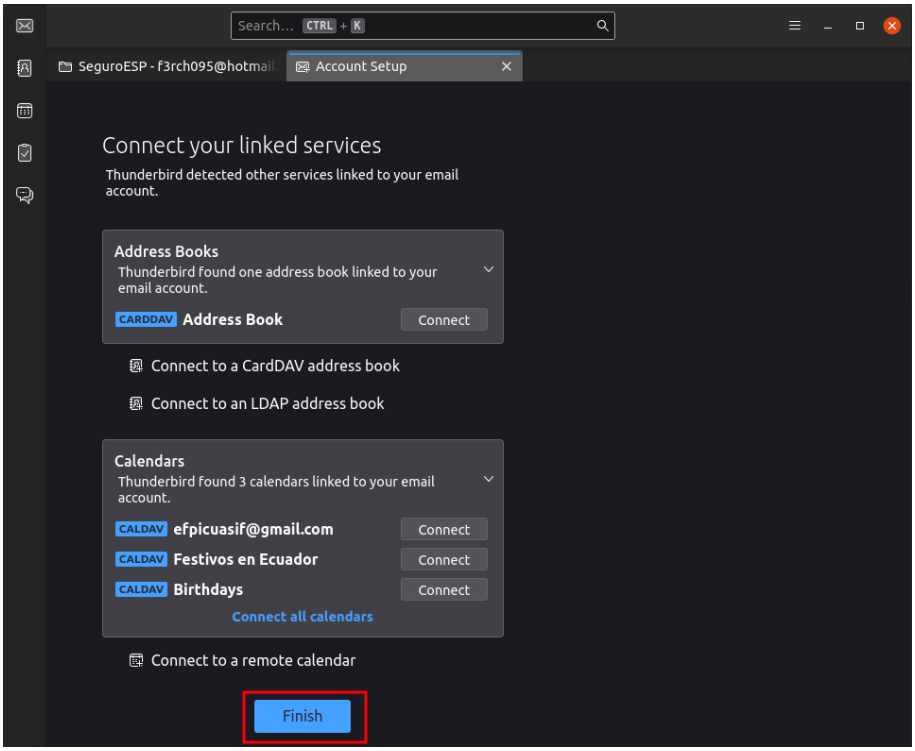
Habilitar permisos de uso



Finalmente se culmina el proceso de instalación y asociación de la cuenta de correo por medio del botón ***finish*** presente en la parte final (ver Figura 37).

Figura 37

Finalizar configuración de la cuenta de correo



4.2 Implementación y Configuración del Sistema de detección de Phishing

Este apartado describe el desarrollo del modelo de detección de phishing basado en el análisis del contenido del correo electrónico, en el que se hace uso de técnicas de NLP, así como el mecanismo de Fine-Tunnig y en conjunto con modelos preentrenados, dicho procedimiento cubre varios procesos enfocados en la adaptación del dataset, el manejo de datos y finalmente el entrenamiento.

4.2.1 Proceso de Entrenamiento

Este proceso será desarrollado en la plataforma de Google Colab, debido a que brinda la infraestructura necesaria para el desarrollo de proyectos en campos como la inteligencia artificial.

En términos generales en este apartado se realizará la adaptación del modelo preentrenado de distilbert-base-multilingual-cased de manera que este sea capaz de ejecutar tareas de clasificación de correos en función de dos categorías (seguro y phishing), además de que pueda operar en base a dos idiomas (inglés y español).

La metodología del entrenamiento se estructura en 4 procesos principales: preparación del conjunto de datos, adquisición de datos, desarrollo del modelo, evaluación y pruebas.

4.2.1.1 Preparación del Conjunto de Datos

El siguiente proceso se centra en expandir la funcionalidad del modelo, con el fin de permitir la detección de correo phishing tanto en inglés como en español. Para llevar a cabo este propósito, es necesario adecuar el dataset con un número consistente de correos en español, lo cual permitirá reducir posibles sesgos lingüísticos en el modelo, la metodología sigue un esquema de traducción similar al empleado por (Rastenis et al., 2021) en el que se hace uso del servicio de traducción de Google para adaptar parte del conjunto de datos empleado en el estudio.

El dataset contemplado para este trabajo se encuentra disponible en la plataforma de Kaggle, el cual consiste en alrededor de 18 mil correos electrónicos, clasificados en dos categorías: ‘Safe Email’ (correos seguros) y “Phishing Email” (correos fraudulentos).

La traducción se realizará por medio de un script desarrollado específicamente para este fin, el cual integra diversas funciones vinculadas a los procesos de codificación, representación de datos, combinación de información y la interpretación del contenido del correo a español.

El proceso inicia con la distribución de los correos en porciones equitativas, para garantizar un balance adecuado de las muestras en español e inglés. Posteriormente, se

procede con la importación del archivo CSV a Google Colab por medio de la librería de Pandas, a la vez que se configuran los parámetros necesarios para una correcta visualización de los datos (Ver Figura 38).

Figura 38

Importación del conjunto de datos

```
# Enlazar directorio de almacenamiento en la nube
drive.mount('/content/drive')

# Cargar el dataset original
email_data = pd.read_csv('/content/drive/MyDrive/CodePhishing_v3/Phishing_Email.csv', encoding='utf-8')
pd.set_option('display.max_colwidth', None)

# Representar los primeros datos del dataset
email_data.head()
```

En los detalles de visualización del conjunto de datos, se muestra los campos más relevantes, correspondientes a las columnas de “Email Text” y “Email Type”. La columna de “Email Text” contiene el texto del mensaje, mientras que la columna de “Email Type” contiene las etiquetas que define si el correo es del tipo seguro o phishing (Ver Figura 39). Además, se muestra que el dataset cuenta con un total de 18650 correos (Ver Figura 40).

Figura 39

Representación de elementos del conjunto de datos

	Unnamed: 0	Email Text	Email Type
0	0	re : 6 . 1100 , disc : uniformitarianism , re : 1086 ; sex / lang dick hudson 's observations on us use of 's on ' but not 'd aughter ' as a vocative are very thought-provoking , but i am not sure that it is fair to attribute this to " sons " being " treated like senior relatives " . for one thing , we do n't normally use ' brother ' in this way any more than we do 'd aughter ' , and it is hard to imagine a natural class comprising senior relatives and 's on ' but excluding ' brother ' . for another , there seem to me to be differences here . if i am not imagining a distinction that is not there , it seems to me that the senior relative terms are used in a wider variety of contexts , e . g . , calling out from a distance to get someone 's attention , and hence at the beginning of an utterance , whereas 's on ' seems more natural in utterances like ' yes , son ' , ' hand me that , son ' than in ones like ' son ! ' or ' son , help me ! ' (although perhaps these latter ones are not completely impossible) . alexis mr	Safe Email
1	1	the other side of * galicismos * * galicismo * is a spanish term which names the improper introduction of french words which are spanish sounding and thus very deceptive to the ear . * galicismo * is often considered to be a * barbarismo * . what would be the term which designates the opposite phenomenon , that is unlawful words of spanish origin which may have crept into french ? can someone provide examples ? thank you joseph m kozono < kozonoj @ gunet . georgetown . edu >	Safe Email

Figura 40

Representación numérica en base al tipo de correo

```
email_type = email_data['Email Type'].value_counts()
print(email_type)
```

```
Email Type
Safe Email        11322
Phishing Email     7328
Name: count, dtype: int64
```

Una vez analizado como se conforma el conjunto de datos, se procede al desarrollo del código necesario para su traducción y la manipulación. Con el objetivo de evitar interrupciones debido a posibles fallas durante la ejecución, se ha diseñado que el proceso de traducción opere en base a puntos de guardado. De esta manera, a pesar de

que se produzcan fallos, la traducción se reanudara a partir del número de correos en el que se produjo el error.

A partir de la premisa expuesta, se configuran las variables de operación, así como de los puntos de guardado. A través de un mecanismo condicional, se constata la existencia correos en caché, lo que permite conservar el flujo de trabajo hasta culminar en su totalidad la traducción de todos los correos.

La Figura 41 ilustra el código correspondiente a la declaración de variables y la funcionalidad de verificación de correos en caché.

Figura 41

Configuración de variables y condicionales de almacenamiento

```
# Declaracion de Variables
SLEEP_TIME = 1
CHECKPOINT_PATH = '/content/drive/MyDrive/CodePhishing_v3/Traduccion/translation_checkpoint.pkl'
BATCH_SIZE = 10

print("Iniciando proceso de traducción")

# Verificación de traducciones almacendas en cache
if os.path.exists(CHECKPOINT_PATH):
    with open(CHECKPOINT_PATH, 'rb') as f:
        translated_cache = pickle.load(f)
    print(f"Checkpoint cargado. Traducciones en caché: {len(translated_cache)}")
else:
    translated_cache = {}
    print("Iniciando nuevo caché de traducciones")
```

Posteriormente, se duplica y segmenta los conjuntos de datos en una proporción del 50%, tomando como referencia el tipo de correo. Los subconjuntos resultantes se asocian a variables identificables, las cuales se empleará en el resto de las funciones del script (ver Figura 42).

Figura 42

División del conjunto de datos

```
# Duplicar los correos electrónicos según su tipo
safe_emails = email_data[email_data['Email Type'] == 'Safe Email'].copy()
phishing_emails = email_data[email_data['Email Type'] == 'Phishing Email'].copy()

# Dividir el 50% del dataset en base al tipo de correo electrónico
safe_sample = safe_emails.sample(frac=0.5, random_state=42)
phishing_sample = phishing_emails.sample(frac=0.5, random_state=42)
```

Para el proceso de traducción, se establece una función que evalúa cada elemento de manera iterativa. En cada iteración, se inicializa el método de la traducción con un número determinado de intentos, y los resultados se almacenan en caché.

La Figura 43 ilustra el código de la función que implementa el ciclo iterativo de la traducción, así como del apartado de almacenamiento en caché.

Figura 43

Segmento de código de la función de traducción

```
def traducir_texto(texto, idx, intentos=3):
    # Caso especial de excepción
    if idx == 12500:
        return texto

    # Verificación de traducciones en caché
    if texto in translated_cache:
        print("Usando traducción existente del caché")
        return translated_cache[texto]

    for intento in range(intentos):
        try:
            print(f"Intentando nueva traducción (intento {intento + 1})")
            time.sleep(SLEEP_TIME)
            translator = Translator()
            traducido = translator.translate(texto, dest='es').text
            # Guardar en caché las traducción exitosas
            translated_cache[texto] = traducido
            print("Nueva traducción completada y guardada en caché")
            return traducido
        except Exception as e:
            print(f"Error en intento {intento + 1}: {str(e)}")
            if intento == intentos - 1:
                return texto
            time.sleep(3)
    return texto
```

En la siguiente parte se segmenta el proceso de traducción para los casos en los que el correo pertenece al tipo seguro o phishing. Mediante ciclos for, se distribuye las tareas de traducción y a través del método loc se accede al contenido del correo por medio de la etiqueta “Email Text”. Posteriormente, se ejecuta la función de traducción, definida como “traducir_texto”, y se reemplaza el texto original con la versión traducida en la celda correspondiente. Además, se agrega una nueva columna en la cual se asigna una etiqueta con el idioma.

Finalmente, los correos traducidos se almacenan una vez alcanzados el número de elementos definido por el tamaño del lote. El procedimiento se aplica de la misma manera tanto para los correos seguros como para los phishing (ver Figura 44 y Figura 45).

Figura 44

Segmento de código del proceso de traducción para el caso de correo seguro

```
for idx in tqdm(safe_sample.index, desc="Traduciendo Safe Emails"):
    texto = safe_sample.loc[idx, 'Email Text']
    print(f"\nProcesando Safe Email {idx}")
    traducido = traducir_texto(texto, idx)
    safe_sample.at[idx, 'Email Text'] = traducido
    safe_sample.at[idx, 'Language'] = 'es'

    emails_procesados += 1
    if emails_procesados % BATCH_SIZE == 0:
        guardar_progreso()
        print(f"Progreso guardado: {emails_procesados} emails procesados")
```

Figura 45

Segmento de código del proceso de traducción para el caso de correo phishing

```
for idx in tqdm(phishing_sample.index, desc="Traduciendo Phishing Emails"):
    texto = phishing_sample.loc[idx, 'Email Text']
    print(f"\nProcesando Phishing Email {idx}")
    traducido = traducir_texto(texto, idx)
    phishing_sample.at[idx, 'Email Text'] = traducido
    phishing_sample.at[idx, 'Language'] = 'es'

    emails_procesados += 1
    if emails_procesados % BATCH_SIZE == 0:
        guardar_progreso()
        print(f"Progreso guardado: {emails_procesados} emails procesados")
```

La Figura 46 ilustra los resultados acumulados del proceso de traducción de los correos seguros como de phishing.

Figura 46

Detalles del progreso de traducción

```
Procesando Phishing Email 6132
Usando traducción existente del caché

Procesando Phishing Email 9884
Intentando nueva traducción (intento 1)
Traduciendo Phishing Emails: 100%|██████████| 3664/3664 [1:15:41<00:00,  1.24s/it]Nueva traducción completada y guardada en caché


Estadísticas finales:
Total de emails procesados: 9325
Total de traducciones en caché: 8909
```

Una vez culminado la traducción de la totalidad de correos, se une en un solo conjunto de datos. A la vez se asigna dicha información a una variable, la cual servirá como parámetro para almacenar el nuevo conjunto de datos (ver Figura 47).

Figura 47

Combinación y almacenaje de correos traducidos

```
# Combinar los datasets traducidos
translated_dataset = pd.concat([safe_sample, phishing_sample])

# Almacenar el dataset combinado
translated_dataset.to_csv('/content/drive/MyDrive/CodePhishing_v3/Traduccion/translated_dataset_final.csv', index=False)
```

A partir del nuevo conjunto de datos, conformado únicamente por correos traducidos, se inicia el proceso de reorganización de la nueva información en los campos correspondientes a la del conjunto de datos original.

Previo al proceso de reorganización, se realiza una verificación de la información que conforma el nuevo conjunto de datos. Para ello, se emplea la librería de pandas, mediante la cual se importa el nuevo conjunto de datos (ver Figura 48).

Figura 48

Importación del dataset

```
email_data_en_es = pd.read_csv('/content/drive/MyDrive/CodePhishing_v3/Traduccion/translated_dataset_final.csv', encoding='utf-8')
pd.set_option('display.max_colwidth', None)

email_data_en_es.head()
```

Asimismo, se representan los primeros elementos de la matriz, con el propósito de constatar la traducción de los correos, que en un principio se encontraba en el idioma inglés (ver Figura 49).

Figura 49

Representación del dataset

	Unnamed: 0	Email Text	Email Type	Language
0	2132	Eirikur dijo:\n> Este incidente es un microcosmos interesante de cómo se desarrollan la sociedad y el derecho.\n> Las mujeres de este pueblo están intentando regular la competencia. lo social \n> los mecanismos de estándares (supongo que la vergüenza ya se ha probado) tienen \n> falló, por lo que lo legal es lo siguiente. El objetivo también podría ser molestar al bañista por motivos no indicados en el\nla historia. El bañista probablemente enojó a los quejosos en otros\ntambién de muchas maneras. La ley y la sociedad son totalmente triviales. Grandes conceptos confusos como\n"putas contra mamás" son vehículos convenientes para disputas insignificantes con la mayor frecuencia posible.\nno.- Lucas	Safe Email	es
1	4702	nominaciones hpl para el 6 de junio de 2000 (ver archivo adjunto : hplo 606 . xls) - hplo 606 . xls	Safe Email	es
2	9938	El martes 13 de agosto de 2002 a las 12:48:18 a.m. +0100, wintermute mencionó:\n> ¿Qué tan inimaginablemente difícil es hacer esto? Hay, hasta donde yo sé,\n> no hay hackers del kernel de Linux, ni distribuciones que se originen en esta feria\n> isla ¿verdad? Bien. Antefacto Linux integrado...lo hizo. Pequeña y agradable distribución que se ejecutaría en una tarjeta flash de 64 MB (con 12 MB libres), con\nun sistema de archivos raíz de sólo lectura. Perfecto para pequeños sistemas integrados. fue un\nVersión muy reducida de RedHat 7.2. Hay que ver si podemos lanzar el sistema de compilación y todo. Ese fue el\npoco interesante. Las distribuciones, en general, son una pérdida de tiempo. No tienes idea\n¿Cuánto mejor podría ser algo como Debian o RedHat que algo\nlo hiciste tú mismo, a menos que hayas elegido un área específica.Kate\n– \nGrupo irlandés de usuarios de Linux: ilug@linux.ie\nhttp://www.linux.ie/mailman/listinfo/ilug para obtener información sobre (des)suscripción.\nMantenedor de la lista: listmaster@linux.ie	Safe Email	es

De la misma manera, por medio del método value_counts, se verifica la cantidad de correos electrónicos que cuentan con la etiqueta del idioma en español (es), así como del número de correos que poseen la etiqueta de ‘Safe Email’ y ‘Phishing Email’ (ver Figura 50).

Figura 50

Representación de información del dataset

```
email_language = email_data_en_es['Language'].value_counts()
print(email_language)
```

```
Language
es    9325
Name: count, dtype: int64
```

```
email_type = email_data_en_es['Email Type'].value_counts()
print(email_type)
```

```
Email Type
Safe Email        5661
Phishing Email    3664
Name: count, dtype: int64
```

Por su parte, la Figura 51 muestran el antes y después del texto presente en el apartado de contenido del correo electrónico. Esto como evidencia del correcto desarrollo de los procesos de traducción.

Figura 51

Detalles de traducción

```
email_data.iloc[2132]['Email Text']
```

```
'Eirikur said:\n> This incident is an interesting microcosm of how society and law develop.\n> The
women of this town are attempting to regulate competition.  The social \n> standards mechanisms (I
expect that shame has already been tried) have \n> failed, so legal is next. The point could also
be to hassle the sunbather for reasons not given in\nthe story.  The sunbather has probably pissed
off the complainers in other\nways, too.Law and society are wholeheartedly trivial.  Big fuzzy con
cepts like\n"sluts vs. moms" are convenient vehicles for petty disputes as often as\nnot.- Luca
s'
```

```
email_data_en_es.iloc[0]['Email Text']
```

```
'Eirikur dijo:\n> Este incidente es un microcosmos interesante de cómo se desarrollan la sociedad
y el derecho.\n> Las mujeres de este pueblo están intentando regular la competencia.  lo social \n
> los mecanismos de estándares (supongo que la vergüenza ya se ha probado) tienen \n> falló, por l
o que lo legal es lo siguiente. El objetivo también podría ser molestar al bañista por motivos no
indicados en el\nla historia.  El bañista probablemente enojó a los quejosos en otros\ntambién de
muchas maneras. La ley y la sociedad son totalmente triviales.  Grandes conceptos confusos como
\n"putas contra mamás" son vehículos convenientes para disputas insignificantes con la mayor frecu
encia posible.\nno.- Lucas'
```

Una vez verificado la manera como está constituido el conjunto de datos se inicia el procedimiento de reorganización, para esto se importan los dos conjuntos de datos, a través del método read_csv de la librería de pandas (ver Figura 52).

Figura 52

Importación de los datasets

```
# Importación de los datasets (Ingles - Español)
original_emails = pd.read_csv('/content/drive/MyDrive/CodePhishing_v3/Phishing_Email.csv', encoding='utf-8')
translated_emails = pd.read_csv('/content/drive/MyDrive/CodePhishing_v3/Traduccion/translated_dataset_final.csv', encoding='utf-8')
```

Con el fin de no alterar la información del conjunto de datos original, se realiza una copia de este y se lo asigna a una nueva variable, con relación al conjunto de datos de los correos traducidos este último se convierte en una estructura de datos del tipo diccionario.

Conforme a los parámetros anteriores se realiza el posicionamiento de cada una las celdas del conjunto de datos el cual por medio de un ciclo for y un condicional anidado se constata la coincidencia de la posición en cada iteración a la vez que se almacena el texto en el nuevo conjunto de datos, finalmente se reordenan los elementos conforme al número es decir su posición numérica y se almacena el dataset resultante en el almacenamiento en la nube de Google Drive (ver Figura 53).

Figura 53

Proceso de combinación y almacenamiento del dataset

```
final_emails = original_emails.copy()
translated_emails_dict = translated_emails.set_index('Numero').to_dict('index')

# Actualización de datos
for numero in translated_emails_dict:
    mask = final_emails['Numero'] == numero
    if mask.any():
        for columna in ['Texto Correo', 'Tipo Correo', 'Idioma']:
            final_emails.loc[mask, columna] = translated_emails_dict[numero][columna]

# Almacenar el dataset combinado y reordenado
final_emails = final_emails.sort_values(by='Numero').reset_index(drop=True)

final_emails.to_csv('/content/drive/MyDrive/CodePhishing_v3/Traduccion/final_emails.csv', index=False)
```

4.2.1.2 Adquisición de Datos

Esta fase corresponde a la obtención del conjunto de datos que se utilizara para entrenar el modelo de detección de phishing. En este caso, se emplea el conjunto de datos modificado en el apartado anterior, al cual se le realizaron las adecuaciones necesarias para incluir un numero robusto de muestras de correo tanto en español como en inglés.

Una vez preparado el conjunto de datos, se importa el contenido a la plataforma de Google Colab y respectivamente a Google Drive (ver Figura 54).

Figura 54

Carga de conjunto de datos

Tras la carga del dataset, se vincula la información al entorno de Google Colab mediante la librería pandas y la configuración de representación, como se ilustra en la Figura 55.

Figura 55

Importación de datos en el entorno de Google Colab

```
FILEPATH = "/content/drive/MyDrive/CodePhishing_v4/Traduccion/final_emails.csv"
drive.mount('/content/drive')

# Cargar los datos
email_data = pd.read_csv(FILEPATH, encoding='utf-8')
pd.set_option('display.max_colwidth', None)
email_data.head()
```

Posteriormente, se lleva a cabo una exploración con más detalles de los datos internos del dataset mediante la función de head, se ilustra los primeros cinco componentes, tal y como se muestra en la Figura 56.

Figura 56

Revisión de contenido del Dataset

	Numero	Texto Correo	Tipo Correo	Idioma
0	0	re : 6 . 1100, disco: uniformismo, re: 1086; Las observaciones de sex/lang Dick Hudson sobre el uso que hacemos de 's on' pero no 'd aughter' como vocativo invitan a la reflexión, pero no estoy seguro de que sea justo atribuir esto a que los "hijos" están siendo "tratados". como parientes mayores ". Por un lado, normalmente no usamos 'hermano' de esta manera más que 'daughter', y es difícil imaginar una clase natural que comprenda a parientes mayores y 's on' pero excluyendo a 'hermano'. por otro lado, me parece que hay diferencias aquí. Si no estoy imaginando una distinción que no existe, me parece que los términos relativos mayores se usan en una variedad más amplia de contextos, e. gramo. , llamar desde lejos para llamar la atención de alguien y, por tanto, al principio de una expresión, mientras que 's on' parece más natural en expresiones como 'sí, hijo', 'pásame eso, hijo' que en expresiones como 'sí, hijo', 'pásame eso, hijo'. ' hijo ! ' o ' hijo , ayúdame ! ' (aunque quizás estos últimos no sean del todo imposibles). alexis señor	Safe Email	es
1	1	the other side of * galicismos * * galicismo * is a spanish term which names the improper introduction of french words which are spanish sounding and thus very deceptive to the ear . * galicismo * is often considered to be a * barbarismo * . what would be the term which designates the opposite phenomenon , that is unlawful words of spanish origin which may have crept into french ? can someone provide examples ? thank you joseph m kozono < kozonoj @ gunet . georgetown . edu >	Safe Email	en

La Figura 57 ilustra los resultados de los procesos de exploración y visualización, de la misma manera en la Figura 58 se muestran las cantidades de correo según el tipo y el idioma respectivamente. Estas etapas son cruciales para obtener una comprensión

inicial de la estructura y características del conjunto de datos, que serán útiles para el análisis en los siguientes pasos.

Figura 57

Representación de emails según su tipo

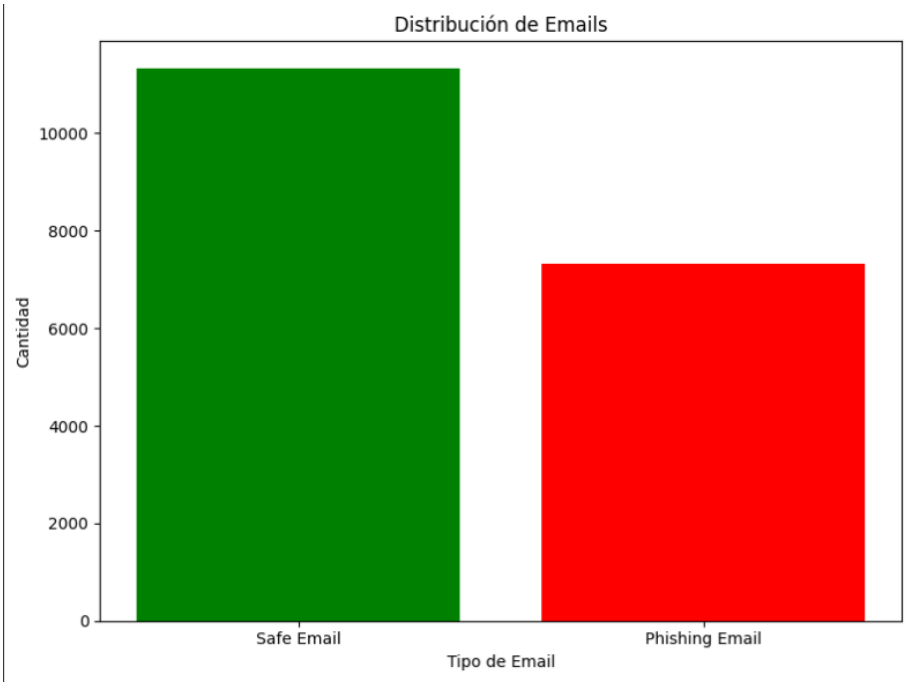
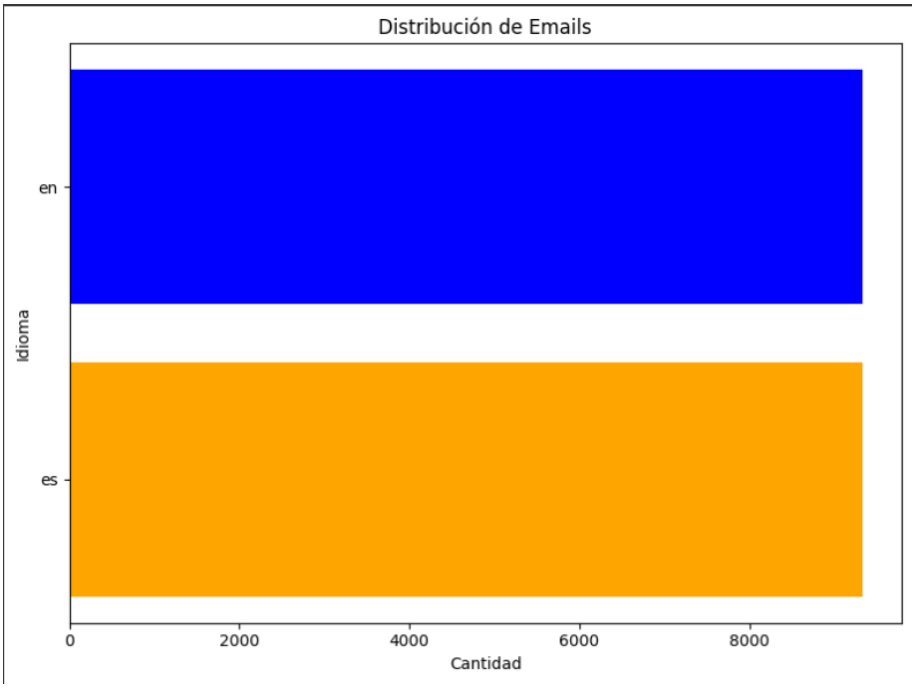


Figura 58

Representación de emails según el idioma



Después de la etapa de obtención de datos se continúa con los procedimientos de preprocesamiento en la cual se realizan acciones de normalización y eliminación de elementos no deseados tales como: caracteres especiales, números y símbolos. También se normalizan la existencia de espacios múltiples (ver Figura 59).

Figura 59

Función de normalización de datos

```
def clean_text(text):
    if pd.isna(text):
        return ""
    # Convertir a string y limpiar caracteres especiales
    text = str(text).encode('ascii', 'ignore').decode('ascii')
    return text.strip()
```

En el siguiente proceso se ejecutan métodos de preprocesamiento que consiste esencialmente en generar uniformidad de la información y mejorar la calidad de la entrada para el entrenamiento del modelo.

La Figura 60 muestra las acciones de ajuste, eliminación de celdas en el conjunto de datos asociadas a la función de preprocesamiento de datos

Figura 60

Función de preprocesamiento de datos

```
def load_and_prepare_data(filepath):
    # Cargar datos con encoding explícito
    data = pd.read_csv(filepath, encoding='utf-8')
    # Renombrar columnas según tu estructura
    data.columns = ['Numero', 'Texto Correo', 'Tipo Correo', 'Idioma']

    # Limpiar texto y asegurarse de que sea string
    data['Texto Correo'] = data['Texto Correo'].apply(clean_text)

    # Filtrar filas vacías
    data = data[data['Texto Correo'] != ""]

    # Convertir etiquetas (ajusta según tus valores específicos)
    data['label'] = data['Tipo Correo'].apply(lambda x: 0 if 'safe' in str(x).lower() else 1)

    return data
```

Posteriormente, se aplica un método de balanceo de datos esto debido a la menor proporción de correos de tipo phishing en el conjunto de datos, como se muestra en la Figura 57. El método combina técnicas de submuestreo y sobremuestreo con el fin de equilibrar las clases del conjunto de datos. Para ello, se reduce los correos seguros a un 80% de su tamaño original por medio de un submuestreo aplicado de manera aleatoria. Paralelamente, se incrementa el tamaño de las muestras de correo de phishing de manera que alcance un numero equivalente al 90% de las muestras de correo seguro Figura 61. Esta aproximación de datos se realiza con el fin de mejorar la generalización del modelo.

Figura 61

Balanceo del conjunto de datos

```
def balance_data(data):
    phishing = data[data['label'] == 1]
    safe = data[data['label'] == 0]

    # Emplear un factor de submuestreo para la clase mayoritaria
    majority_size = int(len(safe) * 0.8)  # Reducir la clase mayoritaria al 80%
    safe_downsampled = resample(safe,
                                replace=False,
                                n_samples=majority_size,
                                random_state=42)

    # Emplear un factor de sobremuestreo más moderado para la clase minoritaria
    minority_size = int(majority_size * 0.9)  # Mantener un minimo de desbalance
    phishing_upsampled = resample(phishing,
                                  replace=True,
                                  n_samples=minority_size,
                                  random_state=42)

    balanced_data = pd.concat([safe_downsampled, phishing_upsampled])
    return balanced_data.sample(frac=1, random_state=42)
```

Se añade una función para el tratamiento de los datos, en la cual se reasignan el tipo de dato con el cual será tratado la columna de ‘Texto Correo’ en la cual se halla el mensaje. Posteriormente se dividen el conjunto de datos en los subconjuntos de validación y pruebas mediante la función de (*stratify*) para mantener la distribución de las

etiquetas, los detalles del código que explican los procesos anteriormente mencionados se ilustran en la Figura 62.

Figura 62

Función de división de conjunto de datos

```
def prepare_datasets(data, tokenizer, max_len):
    # Verificar y limpiar datos
    data['Texto Correo'] = data['Texto Correo'].astype(str)
    data['label'] = data['label'].astype(int)

    # Dividir en train, validation y test
    train_texts, temp_texts, train_labels, temp_labels = train_test_split(
        data['Texto Correo'].tolist(),
        data['label'].tolist(),
        test_size=0.3,
        random_state=42,
        stratify=data['label'].tolist()
    )

    val_texts, test_texts, val_labels, test_labels = train_test_split(
        temp_texts,
        temp_labels,
        test_size=0.5,
        random_state=42,
        stratify=temp_labels
    )
```

En la siguiente parte se trata un aspecto relevante en el contexto del NLP, mediante el tokenizador de DistilBERT se realiza la conversión de los datos de texto en un formato idóneo para el entrenamiento del modelo. La función presente en la Figura 63 establece los procesos para rellenar secuencias cortas de texto, así como cortar secuencias largas de texto, en ambos casos el proceso se encuentra delimitado por el parámetro de *max_length.*

Figura 63

Función de tokenización de datos

```
def tokenize_function(examples):
    """Función de tokenización con manejo adecuado de los tensores"""
    return tokenizer(
        examples['text'],
        padding=True,
        truncation=True,
        max_length=max_len,
        return_tensors=None
    )
```

Finalmente se crean los datasets para el entrenamiento, validación y prueba. En cada caso se crea también un diccionario que incluye el texto y las etiquetas, a los cuales se le aplicaran la función de tokenización. Este proceso se realiza para cada dataset, además con el propósito de hacer más eficientemente el trabajo, los procesos mencionados operan a manera de lotes, a la vez que se eliminan las columnas innecesarias una vez se haya aplicado la tokenización.

La Figura 64 muestra el código relacionado a los procesos de creación de datasets para el entrenamiento, así como los procesos de tokenización de texto.

Figura 64

Creación de datasets y tokenización de texto

```
# Crear y tokenizar dataset de entrenamiento
train_dataset = HFDataset.from_dict({
    'text': train_texts,
    'labels': train_labels
})
train_dataset = train_dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=['text']
)
# Crear y tokenizar dataset de validación
val_dataset = HFDataset.from_dict({
    'text': val_texts,
    'labels': val_labels
})
val_dataset = val_dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=['text']
)
# Crear y tokenizar dataset de prueba
test_dataset = HFDataset.from_dict({
    'text': test_texts,
    'labels': test_labels
})
test_dataset = test_dataset.map(
    tokenize_function,
    batched=True,
    remove_columns=['text']
)
```

4.2.1.3 Desarrollo del Modelo

Para el proceso de entrenamiento se establece una función que emplea un modelo de DistilBERT adaptado a la clasificación de secuencias junto con un optimizador que mejora la estabilidad y rendimiento del entrenamiento. A través de un proceso de entrenamiento de 3 épocas, se itera el modelo para determinar en cada ciclo los valores de perdida y retro propagación. Dichos valores se registran con el fin de dar una perspectiva de cómo evoluciona el aprendizaje del modelo.

La Figura 65, muestra la definición de los parámetros de entrenamiento, el cual señala que el entrenamiento se efectuara con un lote que cuenta con 16 muestras, la tasa de aprendizaje será de un 2e-5, la técnica de optimización para warmup ratio será equivalente a 0.1 y contara con un índice de decaimiento de 0.1. Además de esto se establece un apartado de monitoreo que evalúa el modelo cada 100 pasos y almacena las métricas cada 50 pasos. Con relación al último proceso, se establecen puntos de guardado de manera periódica y al final se obtiene el mejor modelo.

Figura 65

Definición de parámetros y función de entrenamiento

```
def train_model(train_dataset, val_dataset, model, output_dir, batch_size=16, epochs=3):
    training_args = TrainingArguments(
        output_dir=output_dir,
        num_train_epochs=epochs,
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=batch_size,
        warmup_ratio=0.1,
        weight_decay=0.1,
        logging_dir=f'{output_dir}/logs',
        logging_steps=50,
        eval_strategy="steps",
        eval_steps=100,
        save_strategy="steps",
        save_steps=100,
        load_best_model_at_end=True,
        metric_for_best_model='f1',
        save_total_limit=2,
        push_to_hub=False,
        report_to="none",
        learning_rate=2e-5,
        gradient_accumulation_steps=2,
        fp16=True,
        gradient_checkpointing=True
    )
```

A continuación, y como muestra la Figura 66, se define un proceso que detiene la ejecución del entrenamiento, siempre y cuando no se observe una mejora en las métricas de rendimiento, el proceso mencionado precisa de los siguientes parámetros.

- early_stopping_patience: Hace referencia al número de épocas que debe esperar para detener el entrenamiento si no se presentan mejoras del modelo

- early_stopping_threshold: Es el rango mínimo de mejora que debe presentar el modelo con cada iteración.

Figura 66

Proceso de optimización de entrenamiento

```
early_stopping_callback = EarlyStoppingCallback(
    early_stopping_patience=3,
    early_stopping_threshold=0.01
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics,
    callbacks=[early_stopping_callback]
)
```

Una vez inicializado el entrenamiento, y como se presenta en Figura 67 también se visualiza su progreso, en este sentido se puede apreciar como a partir del paso 500 el modelo alcance un balance adecuado, en donde las pérdidas de validación se van estabilizando y las métricas de rendimiento alcanzan un rango optimo en el que no se presentan sobreajustes. En ese sentido se puede decir que en el paso 700 ya se cuenta con un aprendizaje efectivo del modelo y generalización adecuada del mismo, por lo que se interrumpe el entrenamiento tal y como se lo había considerado previo al entrenamiento.

Figura 67

Entrenamiento del modelo

Iniciando entrenamiento...

[700/940 06:54 < 02:22, 1.68 it/s, Epoch 3/5]

Step	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
100	0.322900	0.223671	0.913212	0.913189	0.913205	0.913212
200	0.126700	0.106816	0.961255	0.961266	0.961352	0.961255
300	0.109700	0.107837	0.959706	0.959730	0.962089	0.959706
400	0.058800	0.080089	0.974041	0.974058	0.974835	0.974041
500	0.041000	0.074713	0.976366	0.976380	0.976936	0.976366
600	0.027900	0.080912	0.974041	0.974057	0.974665	0.974041
700	0.028300	0.079285	0.976366	0.976379	0.976793	0.976366

Culminado el proceso de entrenamiento se realiza una evaluación del modelo empleando el conjunto de prueba reservado, de esta manera se obtiene una estimación del rendimiento del modelo en base a las predicciones resultantes de los datos de prueba. Por medio de una matriz de confusión se determinan los valores de aciertos y errores, estos parámetros permiten identificar patrones en la clasificación. Adicionalmente se genera un informe de clasificación en la cual se detallan las métricas de precisión, recall y F1-score, a través de estas métricas se obtiene una comprensión del rendimiento del modelo.

Finalmente se almacena tanto el modelo como el tokenizador esto con la idea de incorporar la funcionalidad de distinción al sistema de detección de ataques phishing basado en correo.

La Figura 68 muestra la fracción del código responsable de los cálculos de las métricas de evaluación para el modelo de clasificación binaria de correos electrónicos, donde los correos seguros son identificados por el valor de 0 a la vez que los correos phishing se identifican con el valor de 1.

Figura 68

Función de evaluación del modelo

```python
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)

    # Métricas generales (weighted average)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='weighted')
    acc = accuracy_score(labels, preds)

    # Métricas por clase (sin promedio)
    precision_per_class, recall_per_class, f1_per_class, support_per_class = \
        precision_recall_fscore_support(labels, preds, average=None)

    return {
        'accuracy': acc,
        'f1': f1,
        'precision': precision,
        'recall': recall,
        # Métricas para correos seguros (clase 0)
        'safe_precision': precision_per_class[0],
        'safe_recall': recall_per_class[0],
        'safe_f1': f1_per_class[0],
        'safe_support': support_per_class[0],
        # Métricas para correos phishing (clase 1)
        'phishing_precision': precision_per_class[1],
        'phishing_recall': recall_per_class[1],
        'phishing_f1': f1_per_class[1],
        'phishing_support': support_per_class[1]
    }
```

En el campo de la Inteligencia Artificial el método habitual para determinar el rendimiento del modelo se lo realiza por medio de la matriz de confusión, esta herramienta permite visualizar de forma más comprensible el número de éxitos y fracasos registrados en la etapa de evaluación. Este matriz se integra de filas y columnas que hacen referencia a valores reales y valores predichos respectivamente.

En este caso, debido al hecho que se opera en función a dos tipos de etiquetas “Safe” y “Phishing” se suscitan cuatro posibles resultados:

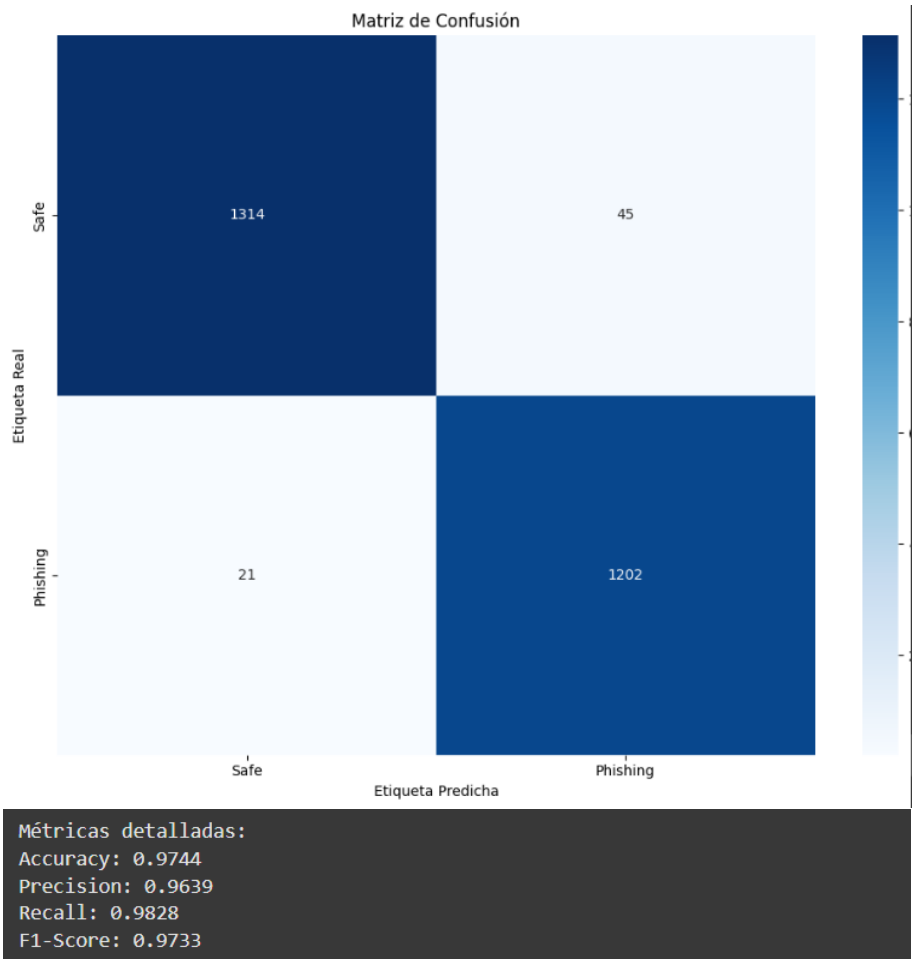
- Verdaderos Negativos (VN): Es el número de casos en donde el correo pertenece al del tipo seguro y el modelo lo clasifico como seguro.
- Verdaderos Positivos (VP): Representa el número de casos en donde el correo pertenece al tipo phishing y el modelo lo catalogo como phishing.
- Falso Negativo (FN): Este caso se produce cuando un correo que pertenece al tipo seguro es clasificado por el modelo como uno del tipo phishing.

- Falso Positivo (FP): Sucede cuando el correo que pertenece al tipo phishing es clasificado por el modelo como uno del tipo seguro.

La Figura 69 muestra el número casos registrados, los cuales fueron obtenidos producto del análisis efectuado al conjunto de datos asignado para las pruebas

Figura 69

Matriz de confusión resultante del entrenamiento



Además del hecho que la matriz de confusión es la representación de los posibles casos a presentarse, también existen métricas más relevantes que definen el rendimiento del modelo y son calculadas a partir los diferentes casos de evaluación (VP, VN, FP y FN).

La Tabla 24 muestra los detalles de las métricas de rendimiento asociadas al conjunto de datos evaluada.

Tabla 24

Métricas de rendimiento

	Predicción	
	Safe	Phishing
Safe	VN: 1314	FP: 45
Phishing	FN: 21	VP: 1202

- **Precisión:** Se utiliza una métrica para determinar el porcentaje de casos positivos correctamente identificados. Esto se calcula dividiendo el número de verdaderos positivos entre el total de resultados positivos, que incluye tanto los verdaderos positivos como los falsos positivos. El modelo entrenado muestra una precisión del 97%.

$$P = \frac{VP}{VP + FP} * 100$$

$$P = \frac{1202}{1202 + 45} * 100 = 96{,}39\%$$

- Recuperación: Es la proporción de verdaderos positivos en relación con la suma de los verdaderos positivos y los falsos negativos.

$$R = \frac{VP}{VP + FN} * 100$$

$$R = \frac{1202}{1202 + 21} * 100 = 98{,}28\%$$

- Puntuación F1: Es el promedio entre precisión y recuperación. El desempeño será mejor mientras más se acerque el valor a 1.0.

$$F1\ score = 2 * \frac{Precisión * Recuperación}{Precisión + Recuperación}$$

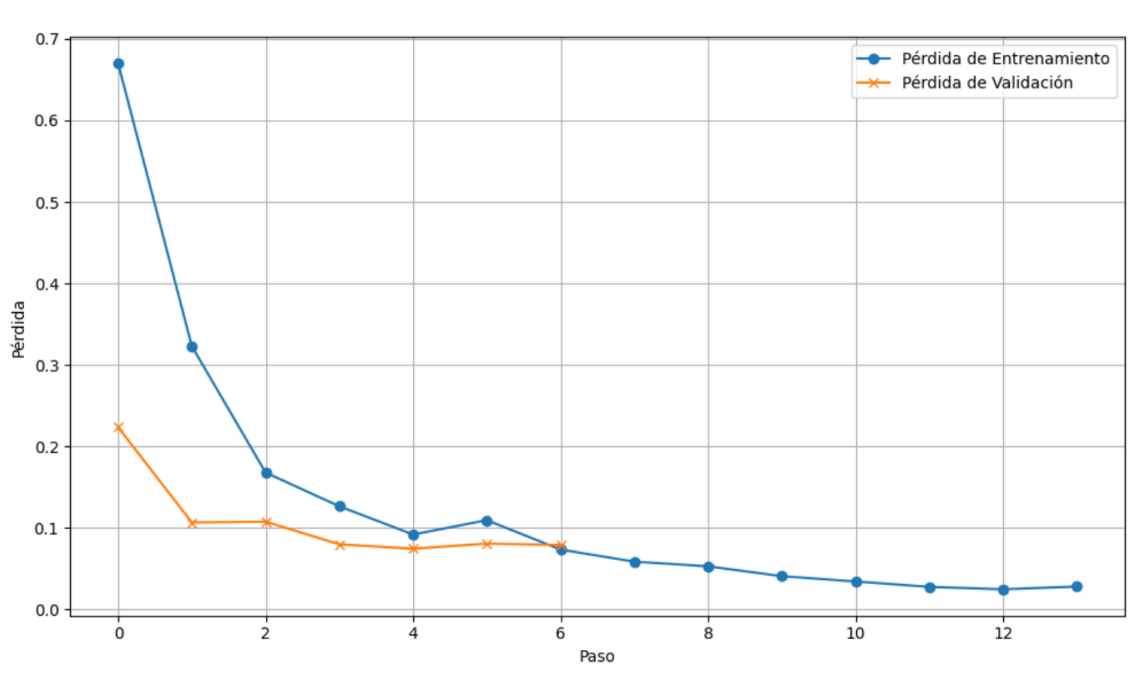
$$F1\ score = 2 * \frac{96{,}39 * 98{,}28}{96{,}39 + 98{,}28} = 97{,}33$$

Posteriormente, se generan las gráficas de las pérdidas correspondientes al entrenamiento y validación con el propósito de examinar el comportamiento del modelo durante su desarrollo. A través de esta gráfica, es posible visualizar cómo, a medida que aumenta el número de pasos, existe una disminución de las perdidas. Es importante destacar que los valores de perdida se registran cada 100 pasos para el caso de la validación y cada 50 pasos para el caso del entrenamiento. Además, mediante la integración de los procesos de optimización y comparación del valor de *f1-score* en cada ciclo, se determina el mejor modelo, a la vez que se reduce el sobreajuste (overfitting) del modelo.

En la Figura 70 se muestra la gráfica los valores de perdida registrados en cada época para los conjuntos de datos del entrenamiento y validación. Se observa que, desde un inicio, ambos parámetros presentan una tendencia decreciente hasta el punto en que se produce la interrupción del entrenamiento. Esto indica, en terminaos generales, que el modelo logro alcanzar una generalización adecuada. Un entrenamiento más prolongado conllevaría al sobreajuste del modelo.

Figura 70

Gráfica de pérdidas



4.2.1.4 Evaluación y Pruebas

El proceso de pruebas emplea dos funciones que operan en base al mapeo etiquetas y el pipeline con el fin de facilitar la inferencia de nuevos textos de correo electrónico. El pipeline permite unificar los procesos de preprocesamiento, tokenizacion y clasificación. A su vez el mapeo de etiquetas permite convertir las etiquetas numéricas a categorías legibles “Safe Email” y “Phishing Email” para mejorar la interpretación al momento de visualizar los resultados.

Los detalles de la interpretación y el fragmento del código asociado a los procesos de inferencia se presentan en la Figura 71.

Figura 71

Función de mapeo de etiquetas y clasificación de emails

```python
def classify_email(email_text, model_path, tokenizer_path):
    """Clasifica un correo electrónico como seguro o phishing."""
    model = DistilBertForSequenceClassification.from_pretrained(model_path)
    tokenizer = DistilBertTokenizer.from_pretrained(tokenizer_path)

    inputs = tokenizer(
        email_text,
        padding=True,
        truncation=True,
        max_length=MAX_LEN,
        return_tensors="pt"
    )

    with torch.no_grad():
        outputs = model(**inputs)
        logits = outputs.logits
        prediction = torch.argmax(logits, dim=1).item()

    if prediction == 0:
        return "Safe Email"
    else:
        return "Phishing Email"
```

La Figura 72 muestra un ejemplo de inferencia en el cual se emplea los datos del tokenizador y el modelo resultante del proceso de entrenamiento.

Figura 72

Pruebas de clasificación

```
email_example = """
    URGENT: Your account has been compromised!

    Dear user, we have detected suspicious activity in your account. Click here immediately to verify your identity: http://suspicious-link.com

    If you don't verify within 24 hours, your account will be permanently closed.

    Security Team
    """
result = classify_email(email_example, f"{OUTPUT_DIR}/final_model", f"{OUTPUT_DIR}/final_model")
print(f"El correo fue clasificado como: {result}")
```

```
El correo fue clasificado como: Phishing Email
```

4.2.2 Código de integración del sistema de detección

Una vez concluido el proceso de preentrenamiento, se procede a la integración del modelo resultante en el escenario de simulación. Esta integración está diseñada como un

proceso capaz de filtrar y gestionar correos potencialmente del tipo phishing, accesible para cualquier dispositivo que sea capaz de ejecutar un cliente de correo. El sistema emplea un proceso de inferencia para realizar predicciones en base a los datos de entrada, que en este caso corresponden a los correos electrónicos.

La Figura 73 ilustra el fragmento de código implementado para realizar las pruebas de funcionamiento del sistema. En este segmento, se importan las librerías y módulos esenciales, de los cuales se destaca principalmente las librerías de "transformers" necesaria para el procesamiento de lenguaje natural (NLP), "email" que permite la manipulación de correos electrónicos, "beatifulsoup" empleada para el proceso de análisis del contenido HTML y "tldextract" para extraer la información de las URLs.

Subsecuentemente, se definen las variables necesarias para la carga y configuración de los parámetros del modelo y del tokenizador, estableciendo así el proceso de inferencia.

Figura 73

Configuración de parámetros del modelo

```
import os
from dotenv import load_dotenv
import httpx
import re
import tldextract
from bs4 import BeautifulSoup
import chardet
from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline
from collections import Counter
import subprocess
from mso_api import get_access_token, MS_GRAPH_BASE_URL
from outlook import search_folder, get_messages, get_sub_folders, delete_message
import time

# Deshabilitar paralelismo de tokenizadores
os.environ["TOKENIZERS_PARALLELISM"] = "false"

# Cargar modelo y tokenizer
MODEL_PATH = "/home/eddpic/Escritorio/Tesis/NLP_Process/ModelData"
tokenizer = AutoTokenizer.from_pretrained(MODEL_PATH)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_PATH)
pipe = pipeline("text-classification", model=model, tokenizer=tokenizer)
```

El proceso de detección opera en base a varias funciones en la que cada uno desempeña una tarea específica.

La función “decode_content” ilustrada en la Figura 74, se encarga de decodificar el contenido del correo electrónico, empleando varias opciones de codificación para interpretar de manera correcta el texto, posteriormente la información decodifica pasa a ser dividida en varios segmentos a los cuales se les aplica procesos para eliminar estilos y fragmentos de código presente en el código HTML, finalmente se obtiene el texto plano del contenido del correo.

Figura 74

Función de decodificación de contenido del correo electrónico

```
def decode_content(content):

    if isinstance(content, bytes):
        detected = chardet.detect(content)
        encoding = detected['encoding']

        try:
            content = content.decode(encoding)
        except:
            for enc in ['utf-8', 'latin-1', 'ascii', 'iso-8859-1']:
                try:
                    content = content.decode(enc)
                    break
                except:
                    pass

    # Divide el contenido en partes
    soup = BeautifulSoup(content, 'html.parser')

    # Remover scripts y estilos
    for script in soup(["script", "style"]):
        script.decompose()

    # Obtener texto plano
    plain_text = soup.get_text(separator='\n')
    plain_text = re.sub(r'\n\s*\n', '\n', plain_text).strip()

    return plain_text
```

De la misma manera, la función “extract_links” presente en la Figura 75 identifica y extrae los hipervínculos presentes en el contenido del correo electrónico, utilizando el atributo ***href*** propias del lenguaje de marcado HTML se extraen los enlaces presentes en las etiquetas ***a*** y **link**.

Figura 75

Función de extracción de enlaces

```
def extract_links(email_content):
    soup = BeautifulSoup(email_content, 'html.parser')

    # Extraer las URLs de los atributos href y src
    urls = set()
    for tag in soup.find_all(['a', 'link'], href=True):
        url = tag.get('href')
        if url and (url.startswith(('http://', 'https://', 'www.'))):
            urls.add(url)

    return list(urls)
```

La función “analyze_phishing” emplea la instancia de clasificación establecida al momento de configurar los parámetros del modelo para evaluar los primeros 512 tokens del correo electrónico. Esta función calcula la probabilidad de phishing en basándose en los tokens y a la etiqueta “LABEL_1”, que corresponde a un caso positivo de phishing. Paralelamente, se evalúa los enlaces dentro del correo electrónico, calculando la proporción de enlaces sospechosos en relación con el dominio del remitente del correo electrónico. Finalmente, se realiza una ponderación de estas métricas, asignando un peso del 65% a la probabilidad de phishing y un 35% a la presencia de enlaces sospechosos (ver Figura 76).

Figura 76

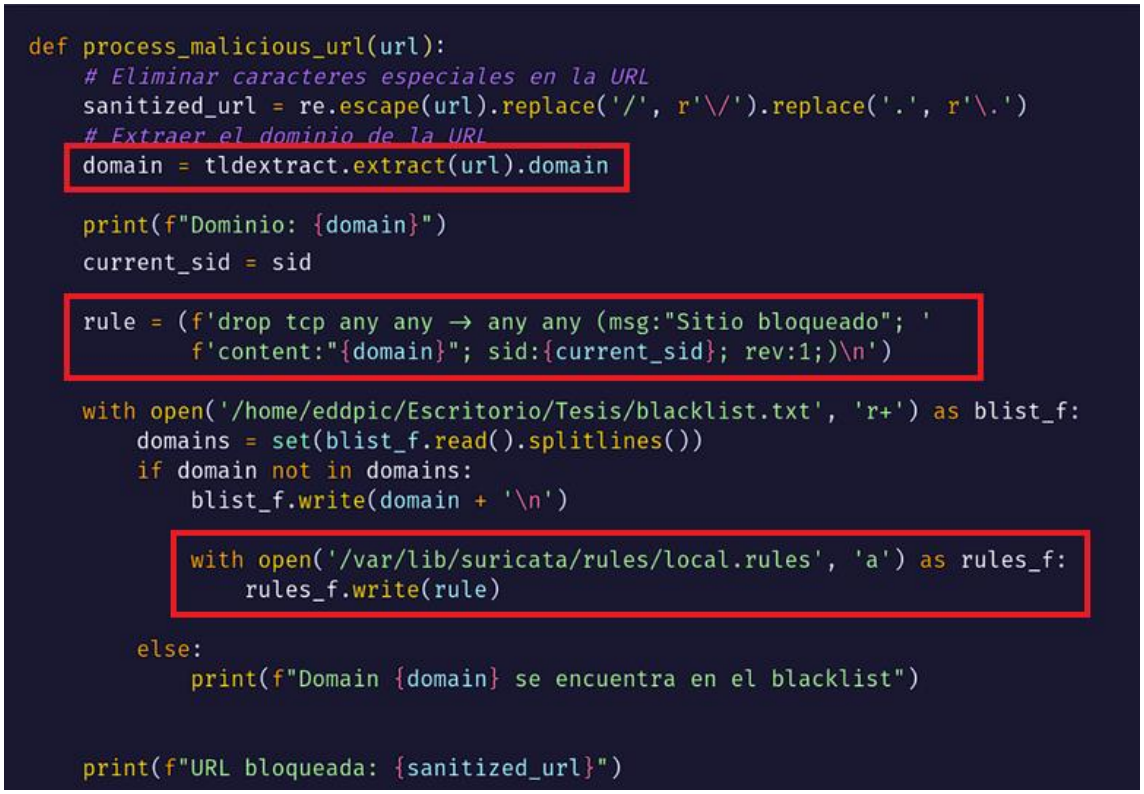
Función de análisis de phishing

```
def analyze_phishing(text, sender_domain, links):
    try:
        result = pipe(text[:512])
        phishing_prob = next((item['score'] for item in result if item['label']
        == 'LABEL_1'), 0)

        suspicious_links = sum(1 for link in links if sender_domain not in link)
        link_ratio = suspicious_links / len(links) if links else 0

        combined_prob = (phishing_prob * 0.65) + (link_ratio * 0.35)

        return combined_prob, phishing_prob, link_ratio
```

La función de “process_malicious_url” presente en la Figura 77 está diseñada para manejar los enlaces identificadas como potencialmente peligrosos. Inicialmente, se limpia la URL eliminando las expresiones regulares que pueden generar inconsistencias en el contexto. Posteriormente, se agrega a un archivo de texto que actúa como lista negra. A continuación, se genera una nueva regla de bloqueo en un formato compatible con el sistema de detección de intrusos Suricata, configurando así el descarte de tráfico http dirigido a la URL maliciosa.

Figura 77

Función de procesamiento de enlaces



```
def process_malicious_url(url):
    # Eliminar caracteres especiales en la URL
    sanitized_url = re.escape(url).replace('/', r'\/').replace('.', r'\.')
    # Extraer el dominio de la URL
    domain = tldextract.extract(url).domain

    print(f"Dominio: {domain}")
    current_sid = sid

    rule = (f'drop tcp any any → any any (msg:"Sitio bloqueado"; '
            f'content:"{domain}"; sid:{current_sid}; rev:1;)\n')

    with open('/home/eddpic/Escritorio/Tesis/blacklist.txt', 'r+') as blist_f:
        domains = set(blist_f.read().splitlines())
        if domain not in domains:
            blist_f.write(domain + '\n')

            with open('/var/lib/suricata/rules/local.rules', 'a') as rules_f:
                rules_f.write(rule)

        else:
            print(f"Domain {domain} se encuentra en el blacklist")


    print(f"URL bloqueada: {sanitized_url}")
```

Esta regla se incorpora al archivo de reglas vinculado a la base de Suricata. La Tabla 25 describe el formato de la regla que se incorporara tanto a la lista negra como a la base de reglas utilizada por Suricata.

Tabla 25

Formato de reglas de Suricata

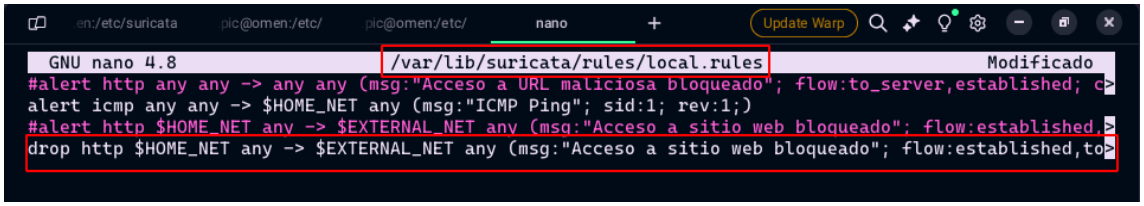
Componente	Valor
Acción	Drop
Protocolo	http
Origen	any any
Destino	any any
Mensaje	"Sitio bloqueado"

Contenido	Variable de Dominio
Identificador Único (SID)	Variable de identificador (sid)
Número de Revisión	1

La Figura 78 muestra la regla generada y almacenada en el archivo de reglas de Suricata.

Figura 78

Base de reglas de Suricata



En la Figura 79 se exhibe la función encargada de generar el correo de alerta, mediante el cual se notifica el número de casos phishing detectados durante el análisis. En términos generales, la función establece los parámetros correspondientes al asunto, el cuerpo del mensaje, los destinatarios y el nivel de importancia. La estructura de la función esta basada en el formato JSON.

Figura 79

Función de correo de alerta

```
def draft_message_body(subject, counter_phishing):
    message = {
        'subject': subject,
        'body': {
            'contentType': 'HTML',
            'content': 'Se han detectado '+str(counter_phishing)+' correos phishing'
        },
        'toRecipients': [
            {
                'emailAddress': {
                    'address': 'f3r      @hotmail.com'
                }
            }
        ],
        'ccRecipients': [
            {
                'emailAddress': {
                    'address': 'f3r      @hotmail.com'
                }
            }
        ],
        'importance': 'high'
    }
    return message
```

La función “main” actúa como componente que integra el resto de las funciones anteriormente descritas. El proceso inicia con la verificación de las credenciales del correo electrónico a través de la API de Microsoft, lo cual habilitara la ejecución de acciones de lectura y escritura de correos electrónicos (ver Figura 80).

Figura 80

Proceso de acceso a cuenta

```
# Credenciales de la API de Microsoft
APPLICATION_ID = os.getenv('APPLICATION_ID')
CLIENT_SECRET = os.getenv('CLIENT_SECRET')
SCOPES = ['User.Read', 'Mail.ReadWrite']

# Contadores de URLs y correos phishing
url_counter = Counter()
counter_phishing = 0

try:
    # Adquirir token de acceso
    access_token = get_access_token(
        application_id=APPLICATION_ID,
        client_secret=CLIENT_SECRET,
        scopes=SCOPES
    )
    headers = {
        'Authorization': 'Bearer ' + access_token
    }
```

En la Figura 81 se ejecuta el siguiente paso en lo que respecta al proceso de conexión a la cuenta de correo, a partir de los métodos resaltados se realiza la identificación del subdirectorio al cual se aplicará el análisis de phishing mediante el modelo.

Figura 81

Búsqueda de subdirectorios en la cuenta de correo

```
    # Buscar la carpeta de la Bandeja de Entrada
    folder_name = 'Bandeja de entrada'
    target_folder = search_folder(headers, folder_name)
    folder_id = target_folder['id']

    # Adquirir subcarpetas
    sub_folders = get_sub_folders(headers, folder_id)
```

Una vez obtenido el directorio de análisis se inicializa el método de adquisición de correos y por medio de la iteración de cada correo, se extrae la información del

remitente del correo, decodifica el contenido del mensaje, extrae enlaces y analiza su contenido (ver Figura 82).

Figura 82

Proceso iterativo de análisis de phishing

```
for sub_folder in sub_folders:
    if sub_folder['displayName'].lower() == 'TestEmailModel'.lower():
        sub_folder_id = sub_folder['id']
        messages = get_messages(headers, sub_folder_id)
        for message in messages:
            time.sleep(1)
            # Reinicia el contador de URLs
            url_counter = Counter()
            # Extraer información del mensaje
            sender = message.get('sender', {}).get('emailAddress', {}).get('address', '')
            subject = message.get('subject', '')
            body_content = message.get('body', {}).get('content', '')
            # Decodificar el contenido del mensaje
            decoded_body = decode_content(body_content)
            #print(f"Contenido del Correo:\n{decoded_body}")
            print()
            # Extraer enlaces
            links = extract_links(body_content)
            print(f"Links: {links}")
            url_counter.update(links)
            # Extraer el dominio del remitente
            sender_domain = tldextract.extract(sender.split('@')[-1]).domain
            print(f"Sender domain: {sender_domain}")
            # Analiza phishing
            combined_prob, model_prob, link_ratio = analyze_phishing(
                decoded_body,
                sender_domain,
                links)
```

Finalmente, y una vez que culmine el análisis de phishing se procede a ejecutar las contramedidas en las que se incluye la generación de las reglas de Suricata y la eliminación de los correos categorizados como phishing. Además, se genera una alerta a manera de un borrador de correo electrónico en donde se menciona el número de casos de phishing suscitados (ver Figura 83).

Figura 83

Segmento de código asociado a la generación de alertas y contramedidas

```
    if model_prob > 0.50:
        counter_phishing += 1
        print("Correo Phishing")
        print()
        print("Blocked URLs:")
        for url in links:
            process_malicious_url(url)
            print(url)
        delete_message(headers, message['id'])
    else:
        print("Correo seguro")
        counter_safe+=1
    print()
    print('=' * 80)
    print()
print(f"Total de correos phishing detectados: {counter_phishing}")
print(f"Total de correos seguros: {counter_safe}")
print(f"Total de correos: {item}")
subject = "Alerta de Phishing"
message = draft_message_body(subject, counter_phishing)
email_data = {"message": message}
endpoint = f'{MS_GRAPH_BASE_URL}/me/sendMail'
response = httpx.post(endpoint, headers=headers, json=email_data)
if response.status_code != 202:
    print(f"Error al enviar notificación {subject}: {response.text}")
print(response.status_code)
```

4.3 Implementación del plan de Ataque

4.3.1 Reconocimiento

El siguiente apartado se centra en la recopilación de datos acerca del blanco del ataque, por medio de técnicas de Inteligencia de Fuentes Abiertas (OSINT). Para ello, se empleó la herramienta de Hunter.io, la cual permite investigar direcciones de correos electrónicos asociados a entidades u organizaciones y por consiguiente obtener información relevante el de la víctima. Para este caso se hace un análisis del dominio utn.edu.ec, con el objetivo de identificar el patrón de los correos electrónicos de los individuos que forman parte de dicha organización,

Con base en esta información, se plantean estrategias de Ingeniería Social para tratar de obtener nueva información por medio de ataques dirigidos hacia los puntos más vulnerables dentro de la organización.

La Figura 84 muestra el proceso de recopilación de información asociada a la Universidad Técnica del Norte, en el que se identifica el patrón de correos electrónicos. Asimismo, la Figura 85 muestra un número considerable de resultados vinculados con la organización educativa.

Figura 84

Detalles del descubrimiento y recolección de correos

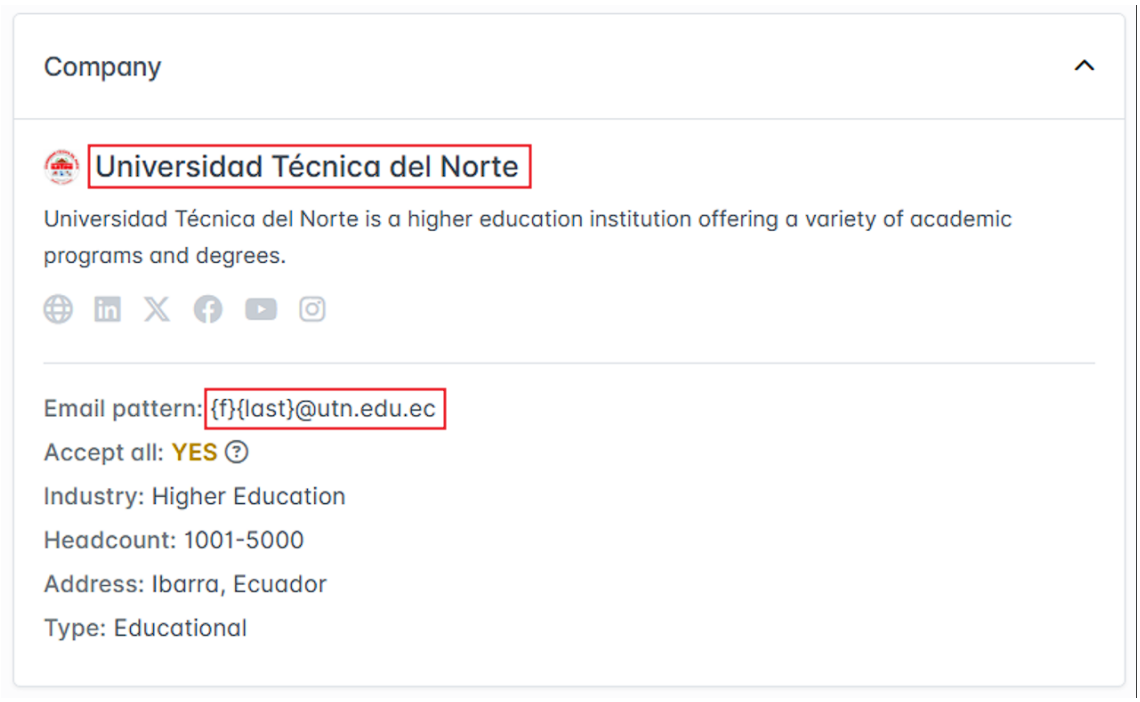
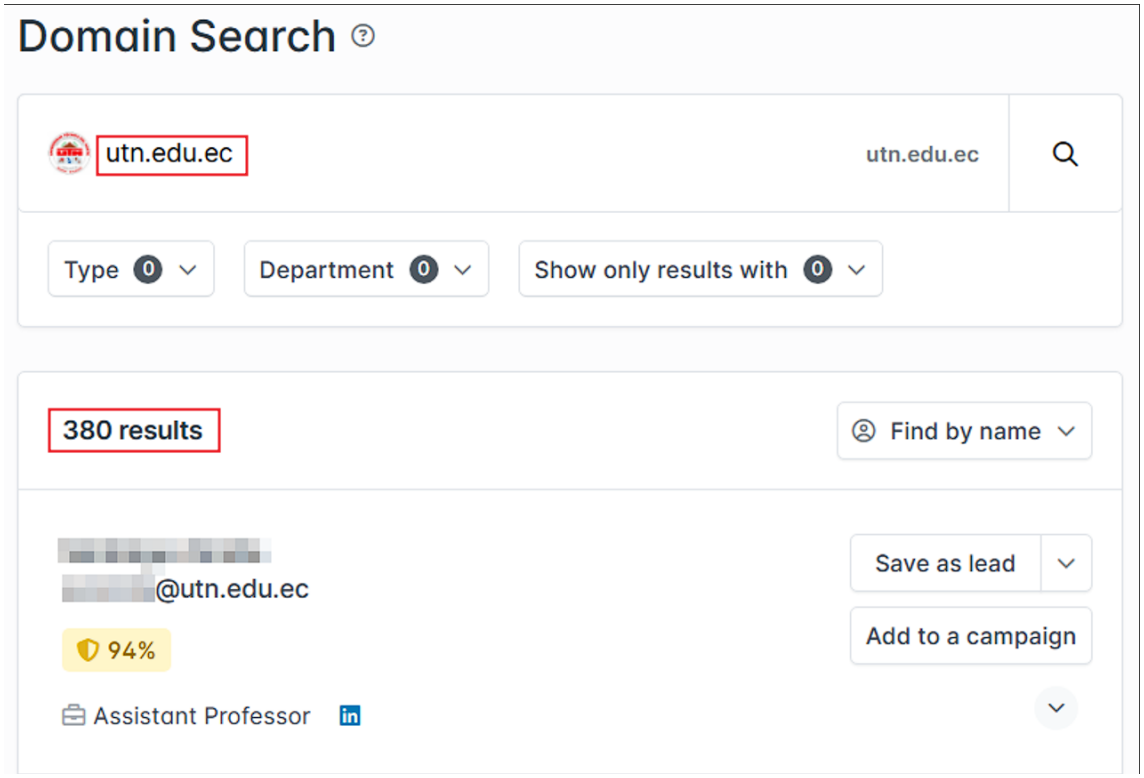


Figura 85

Detalles de la búsqueda de dominio



4.3.2 Desarrollo de Recursos

En el siguiente proceso se preparan las herramientas y servicios necesarios para ejecutar el ataque. Para ello, se inicia con el despliegue del entorno destinado a la elaboración de señuelos y a la captura de credenciales de inicio de sesión, utilizando Evilginx. Paralelamente se configura la herramienta de Gophish para la creación y gestión de las campañas de phishing. Como complemento a la simulación del ataque, se configura un dominio fraudulento, que será empleado tanto por Evilginx como por Gophish

Los aspectos mencionados anteriormente se ejecutarán de manera secuencial, partiendo desde la creación de una máquina virtual en la nube, seguida de la instalación y configuración de las herramientas. Además, se elabora un segmento centrado en la creación del dominio fraudulento.

4.3.2.1 Instalación de Evilginx

Para el despliegue de Evilginx, se optó por utilizar la plataforma de servicios en la nube Azure, propietaria de Microsoft. Una vez dentro del entorno de Azure, se procede con la creación de una nueva máquina virtual, en la cual se instalará el servidor correspondiente.

Como parte del proceso, se generó un nuevo proyecto en el cual se establecen el modelo de suscripción y el grupo de recursos a cuál se asociará el proyecto. Además, se especificaron las características de la nueva instancia. En términos generales, se configuraron los detalles prioritarios, tales como: la imagen que hace referencia a la plantilla de configuración del sistema operativo, la región que determina la ubicación del centro de datos en donde se alojaran los recursos, y las especificaciones de hardware (ver Figura 86 y Figura 87).

Figura 86

Configuración de la instancia de Evilginx - Parte 1

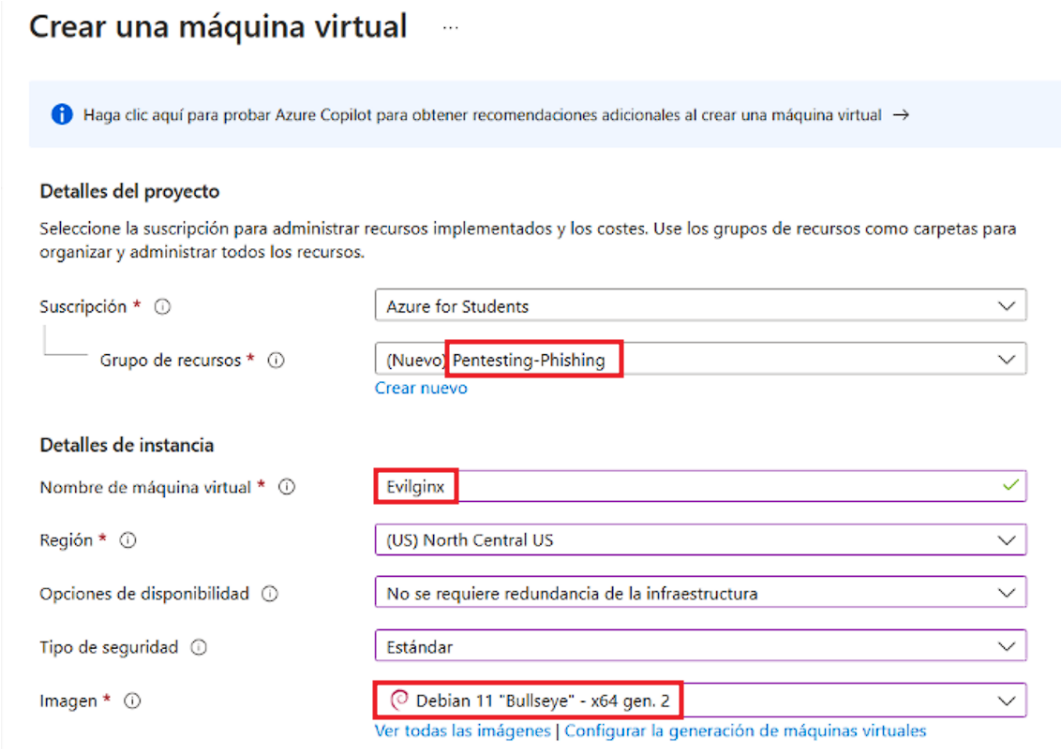
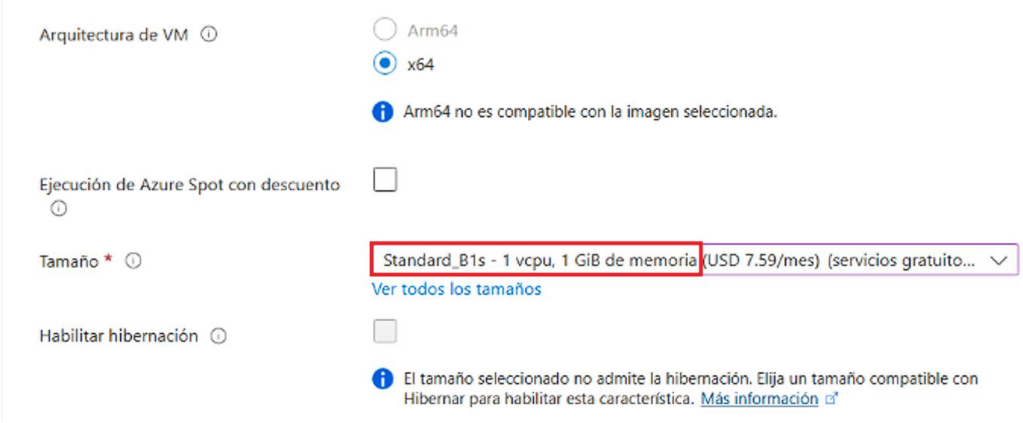


Figura 87

Configuración de la instancia de Evilginx - Parte 2

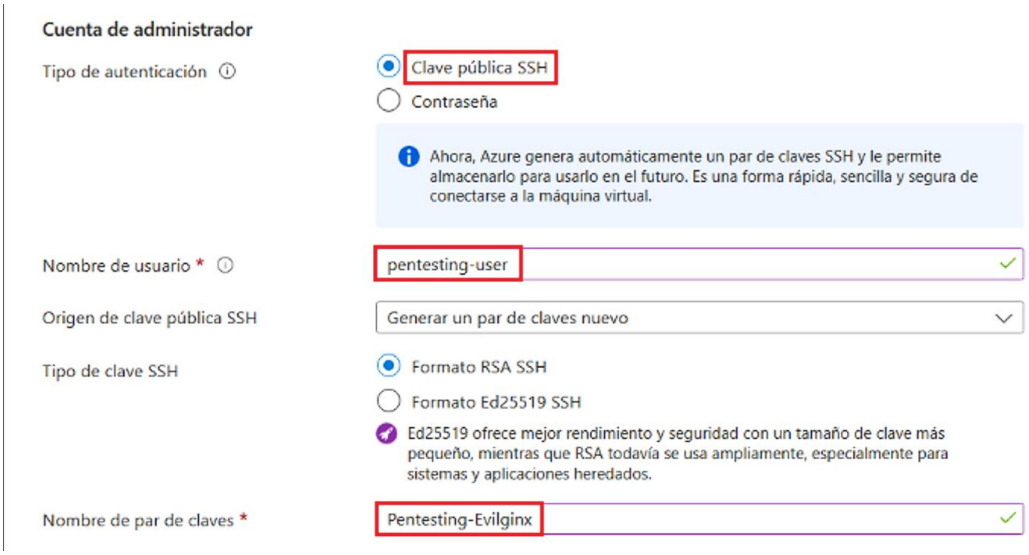


En la sección destinada al administrador, se fija el método de autentificación. En este caso, el inicio de sesión en los servidores remotos se lo realizara por medio de una clave publica SSH. Este método genera dos tipos de claves: una clave publica y una clave privada. La clave publica se comparte con el servidor remoto, mientras que la clave privada se almacena en la máquina del cliente. Solo cuando ambas claves coincidan se produce el inicio de sesión.

La Figura 88 muestra a detalle las opciones disponibles para la selección del método de autentificación asociada a la conexión remota.

Figura 88

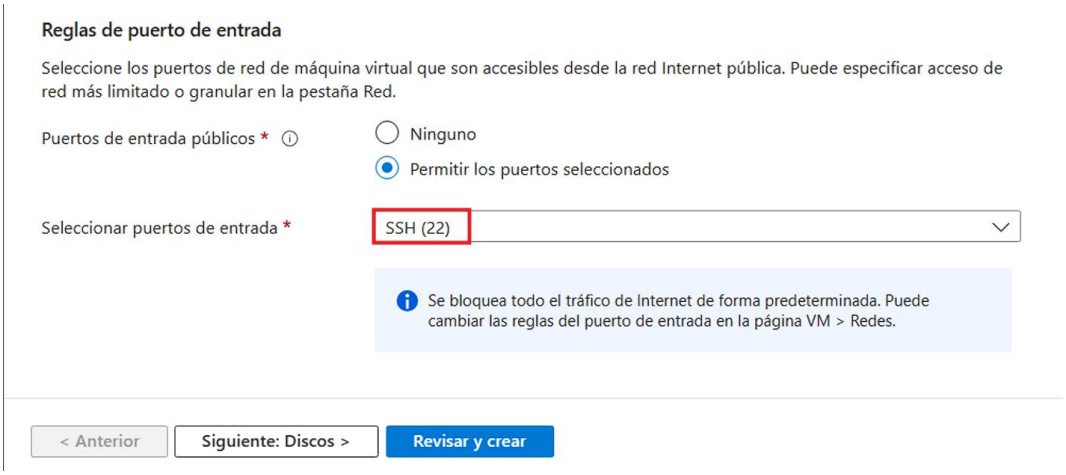
Configuración de la instancia de Evilginx - Parte 3



Anexo al apartado de autentificación, se habilitan los puertos de entrada del protocolo SSH (ver Figura 89).

Figura 89

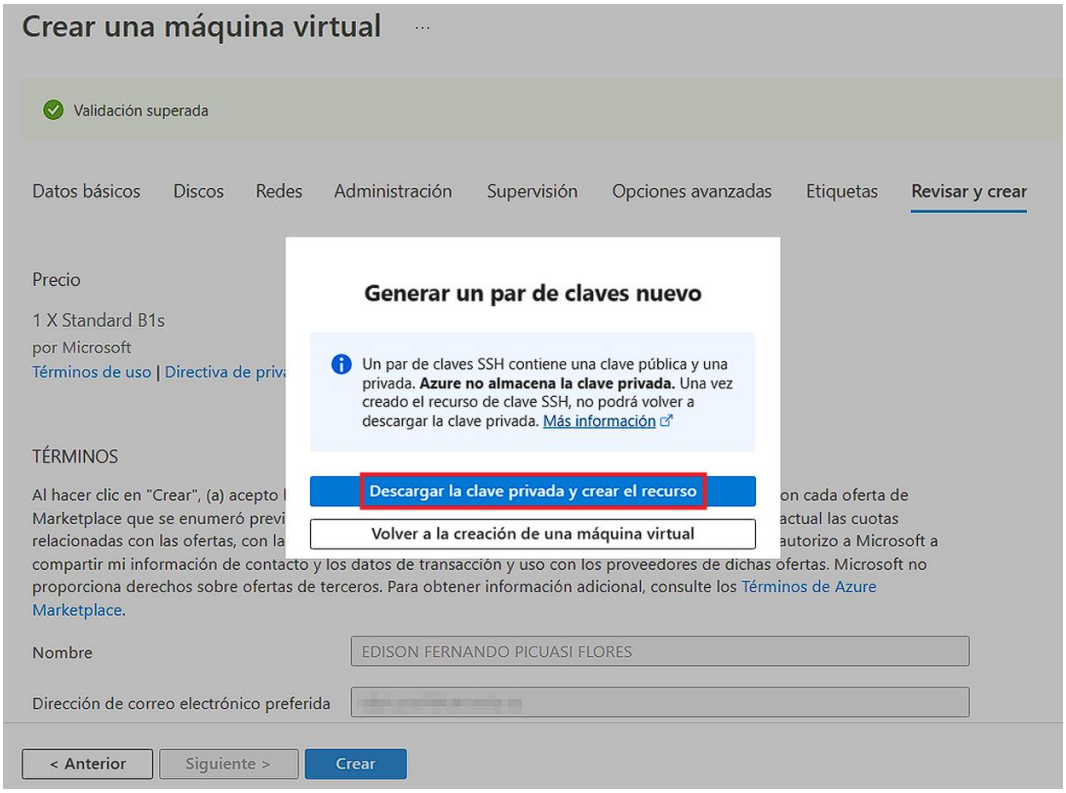
Configuración de la instancia de Evilginx - Parte 4



Una vez finalizado el proceso de configuración de la instancia, se procede a la creación del recurso, lo que incluye la generación de la clave privada necesaria para establecer la conexión remota con el servidor, tal como se muestra en la Figura 90.

Figura 90

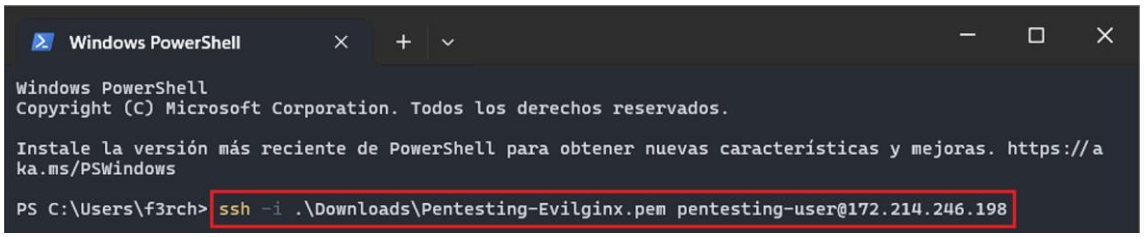
Configuración de la instancia de Evilginx - Parte 5



A partir de la clave privada se inicia el proceso de autentificación e inicio de sesión en el servidor remoto desde la terminal de comandos (ver Figura 91).

Figura 91

Inicio de sesión remota



Una vez accedido al servidor remoto, se procede con la instalación del Evilginx. En donde como primer pasos y través del comando ***wget*** se adquiere el paquete de instalación tal y como se muestra en la Figura 92.

Figura 92

Descarga del archivo de instalación de Evilginx

Posterior a la descarga del archivo zip, se procede a descomprimir el archivo, del conjunto de archivos y directorios resultantes se concede el permiso de ejecución al archivo Evilginx (ver Figura 93 y Figura 94).

Figura 93

Descompresión del archivo de instalación

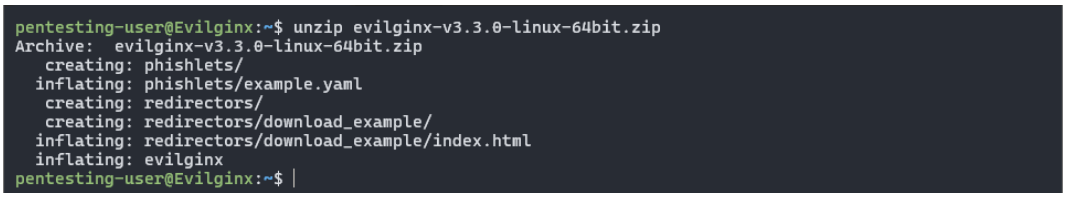
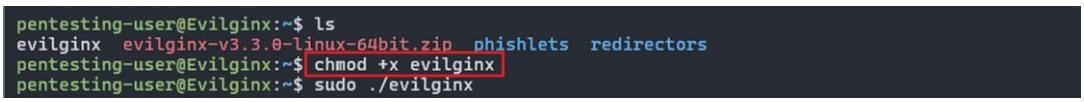


Figura 94

Asignación de permisos de ejecución



Por medio del comando ***sudo ./evilginx*** se inicia el servicio de Evilginx a la vez que se constata la correcta instalación del servicio, como se muestra en la Figura 95.

Figura 95

Interfaz de línea de comandos de Evilginx

4.3.2.2 Instalación de Gophish

De manera similar al apartado anterior se procede a crear la instancia en la que se instalara el servidor de Gophish siguiendo el patrón de configuración realizado para la instancia de Evilginx, se establecen las configuraciones prioritarias asociadas a la ubicación del centro de datos y la plantilla de configuración del sistema operativo (ver Figura 96), las especificaciones de hardware de la instancia (ver Figura 97).

Figura 96

Configuración de la instancia de Gophish - Parte 1

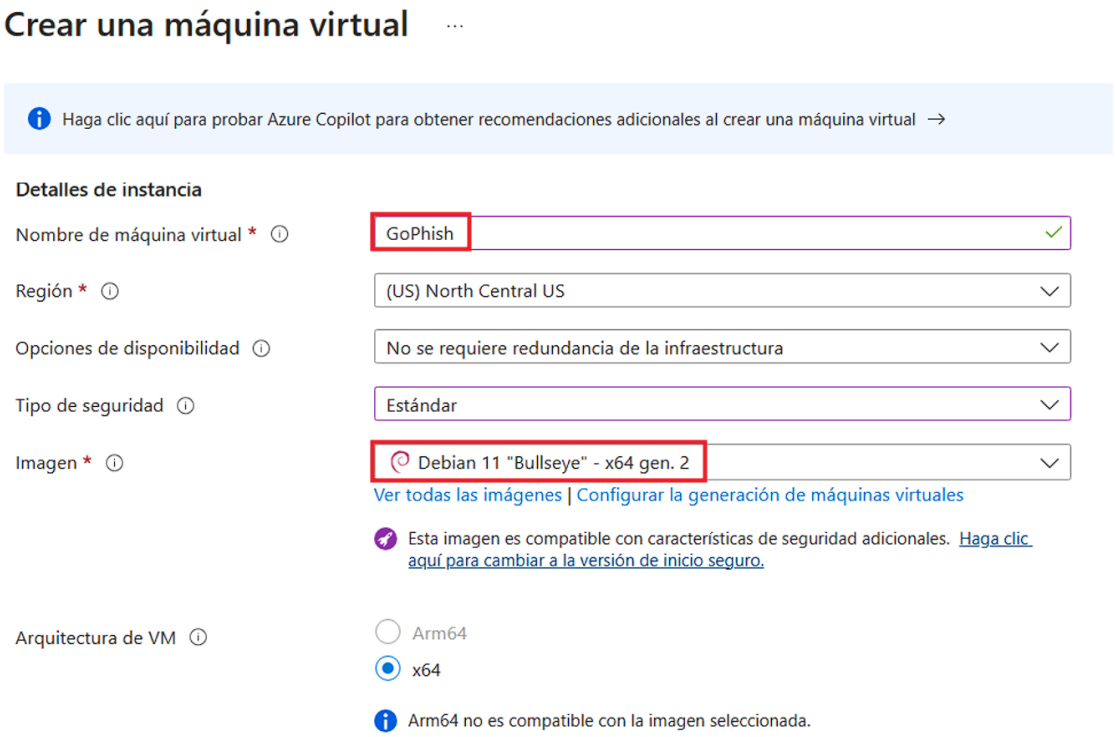
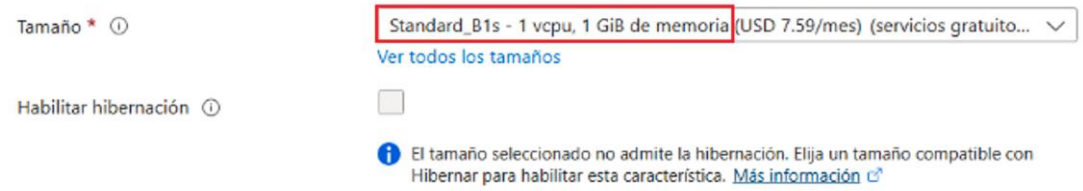


Figura 97

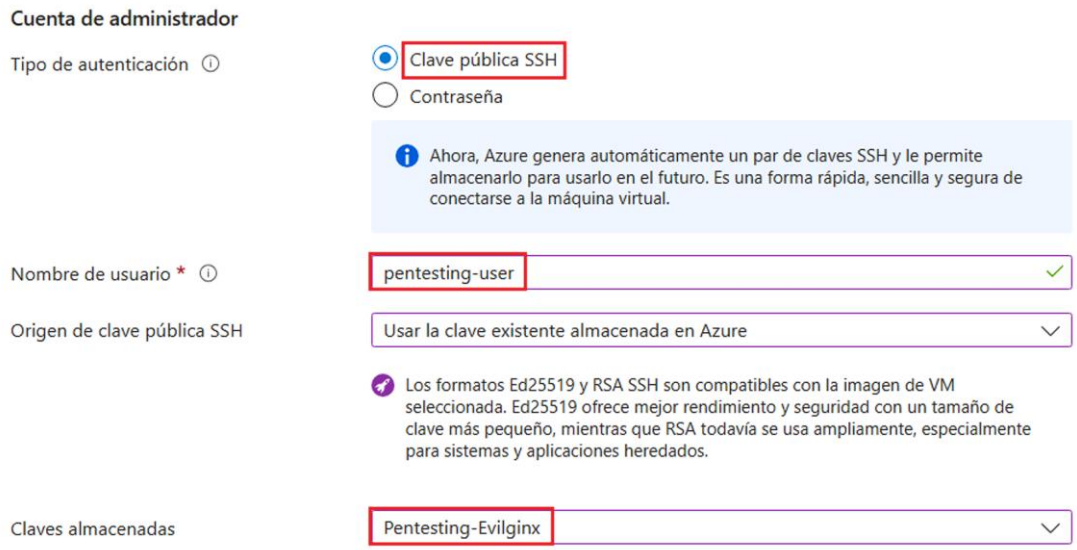
Configuración de la instancia de Gophish - Parte 2



En el aspecto del tipo de autenticación se fija el acceso por medio de una clave publica asociado con el protocolo SSH (ver Figura 98).

Figura 98

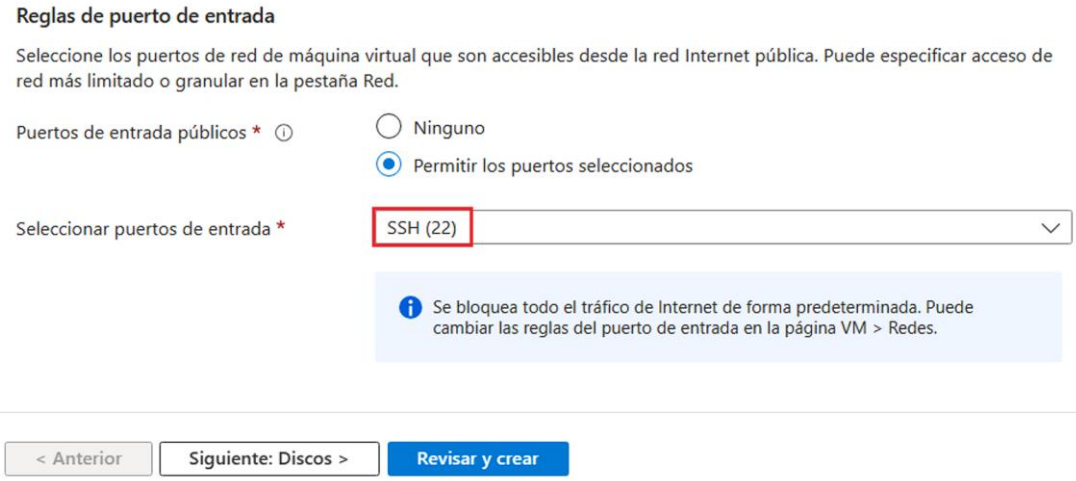
Configuración de la instancia de Gophish - Parte 3



A la vez que se habilita los puertos de entrada para la conexión remota por medio del protocolo SSH (ver Figura 99).

Figura 99

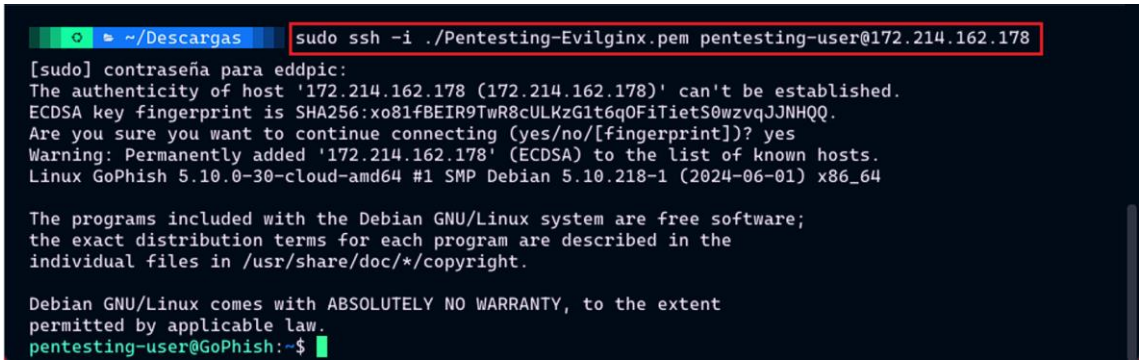
Configuración de la instancia de Gophish - Parte 4



A través de la clave privada obtenida una vez culminada la configuración se inicia el proceso de acceso con el servidor remoto por medio del protocolo SSH (ver Figura 100).

Figura 100

Inicio de sesión remoto



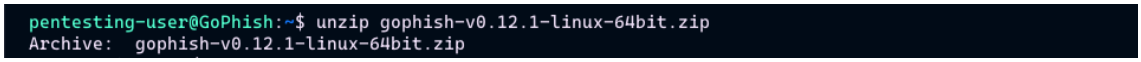
Una vez realizado el inicio de sesión se procede a descargar el archivo de instalación de Gophish (ver Figura 101), para posteriormente obtener los archivos y directorios por medio del comando unzip (ver Figura 102).

Figura 101

Descarga de archivo de instalación de Gophish

Figura 102

Descompresión del archivo de instalación



Finalizado la descompresión del archivo, se localiza el archivo ejecutable del servicio y se concede los permisos de ejecución necesarios, el comando para dicho proceso es presentado en la Figura 103.

Figura 103

Asignación de permisos de ejecución

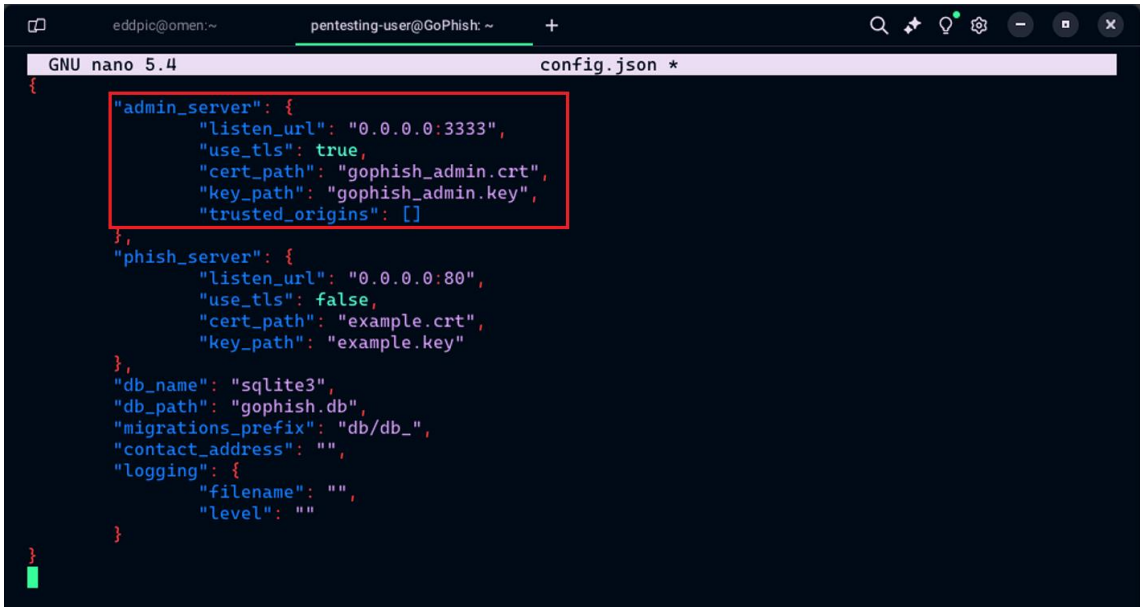
Antes de inicializar el servicio se establecen los parámetros de conexión, de manera que todas interfaces en el puerto sean accesibles, con el de que cualquier dirección

IP pueda establecer conexión con el servidor de Gophish, además de esto se habilita para el acceso el protocolo de seguridad TLS.

La Figura 104 muestra la sección de código a editar en el archivo de configuración de Gophish para habilitar la conexión desde cualquier IP.

Figura 104

Configuración de parámetros de conexión



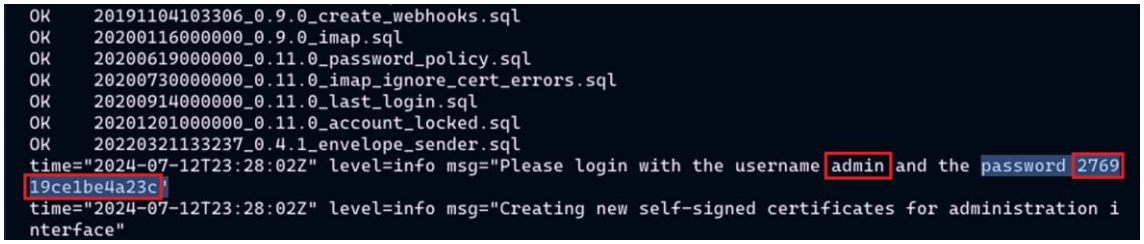
Realizado los cambios pertinentes se inicia y constata la correcta funcionalidad del servicio (ver Figura 105), una vez cumplido los requerimientos se generan las credenciales por defecto para el inicio de sesión (ver Figura 106).

Figura 105

Inicialización del servicio de Gophish

Figura 106

Parámetros de inicio de sesión



A través de las credenciales generadas se accede a la interfaz web del servicio (ver Figura 107), la cual cuenta con varias opciones de configuración para el proceso de creación y despliegue de campanas phishing (Figura 108).

Figura 107

Interfaz web de inicio de sesión de Gophish

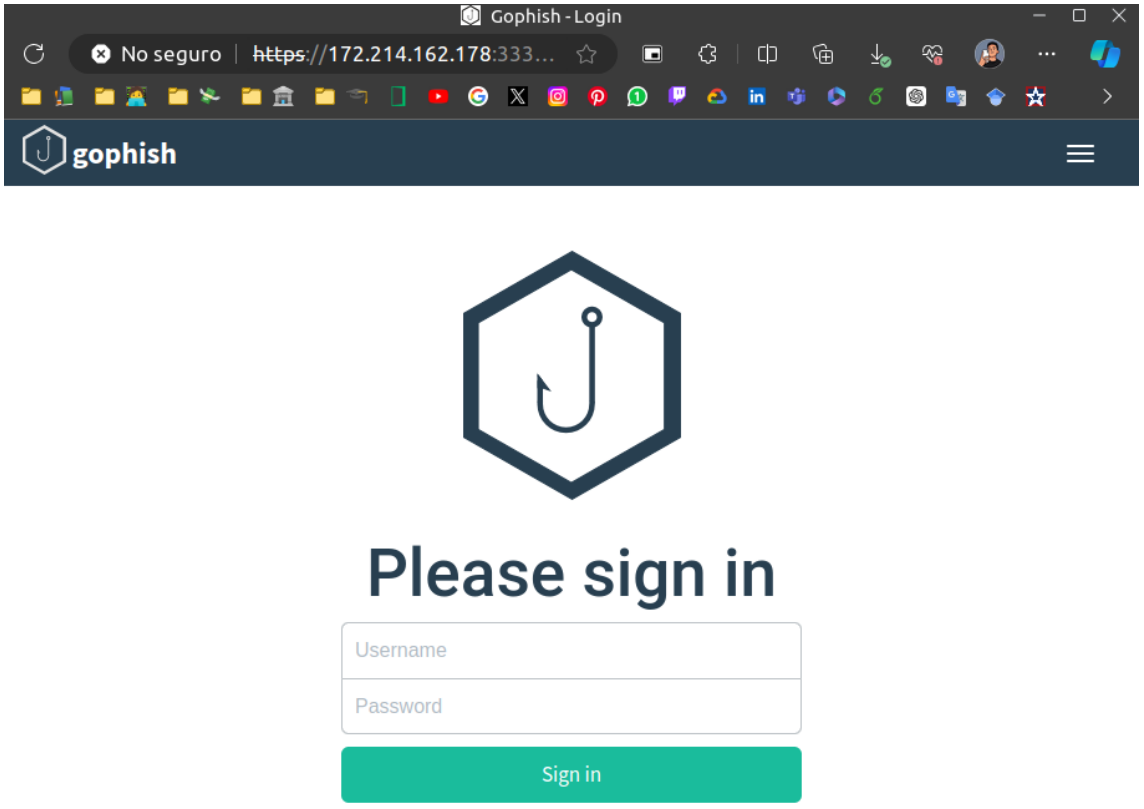
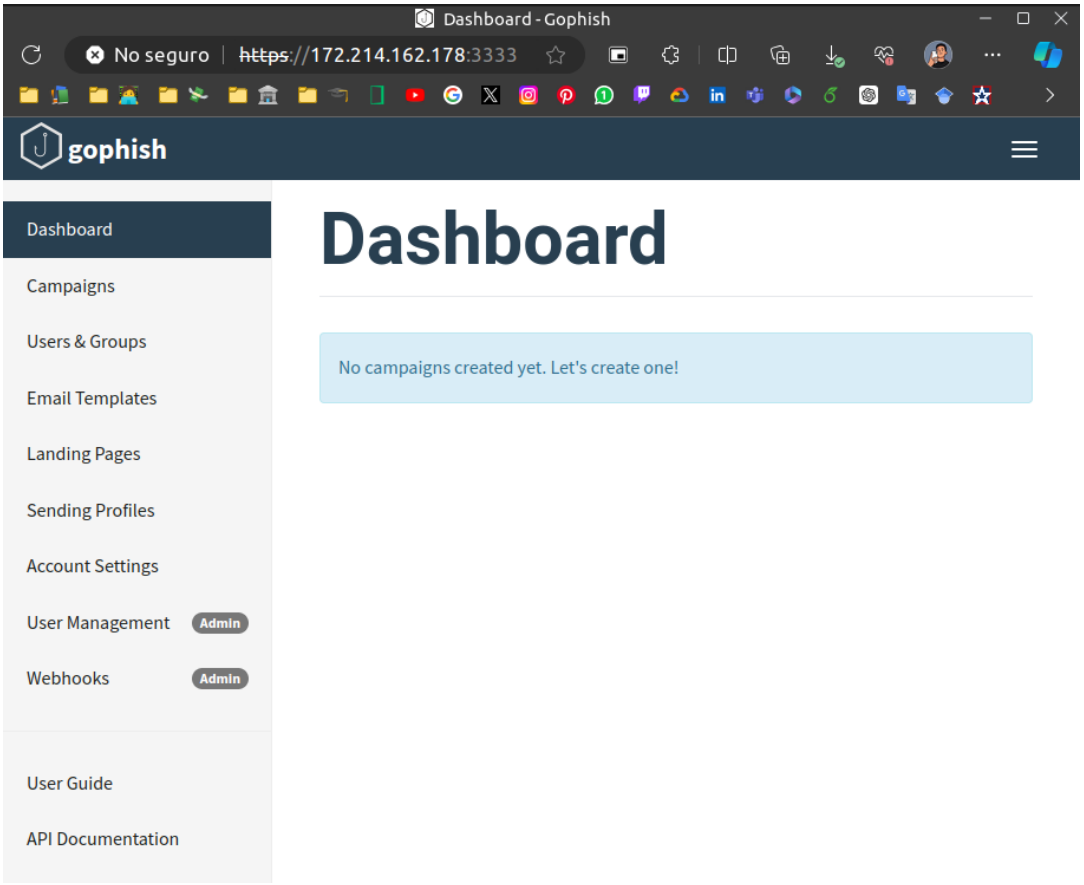


Figura 108

Panel de control de Gophish



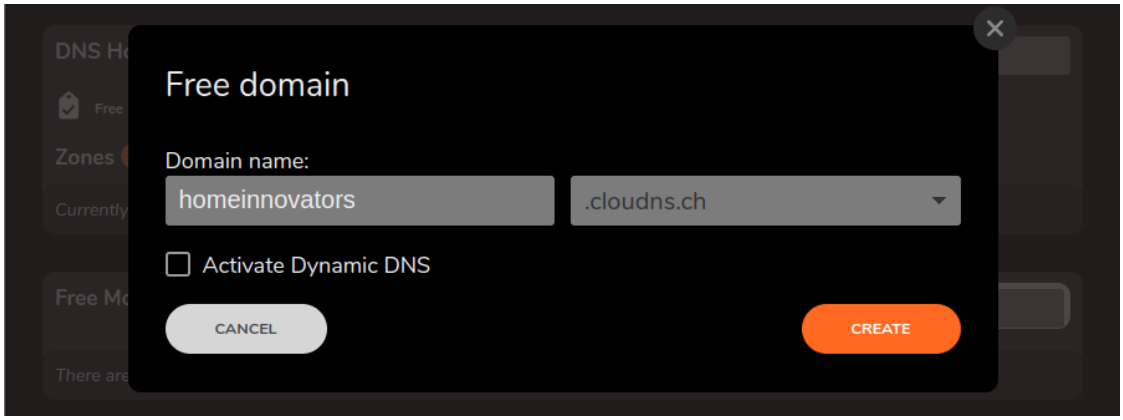
4.3.2.3 Configuración de dominio fraudulento

El siguiente apartado hace uso el Sistema de Nombres de Dominio (DNS) en la nube de CloudDNS, con el fin de que el ataque simulado persuada a la víctima de la legitimidad del correo electrónico.

Una vez ingresado a la plataforma de CloudDNS se procede a generar un nuevo dominio que se empleara posteriormente en los procesos de generación de señuelos por medio de Evilginx (ver Figura 109).

Figura 109

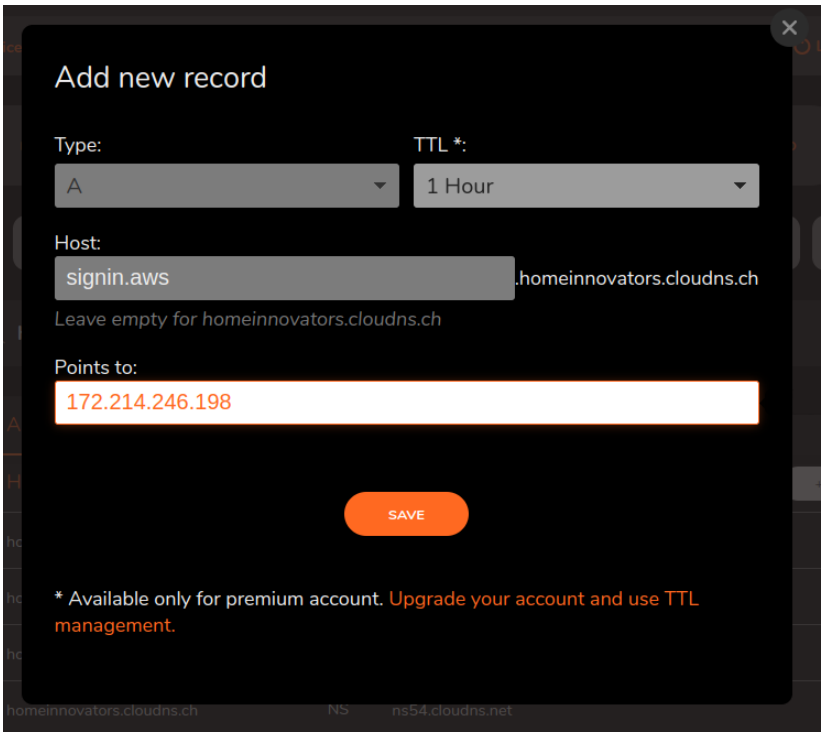
Configuración de dominio



Posterior a la generación del dominio se crean varios registros con subdominios que se adecuen a los procesos de redirección de los sitios legítimos, y que pasen por la dirección IP del servidor de Evilginx para de esta manera realizar la captura de información de la víctima (ver Figura 110).

Figura 110

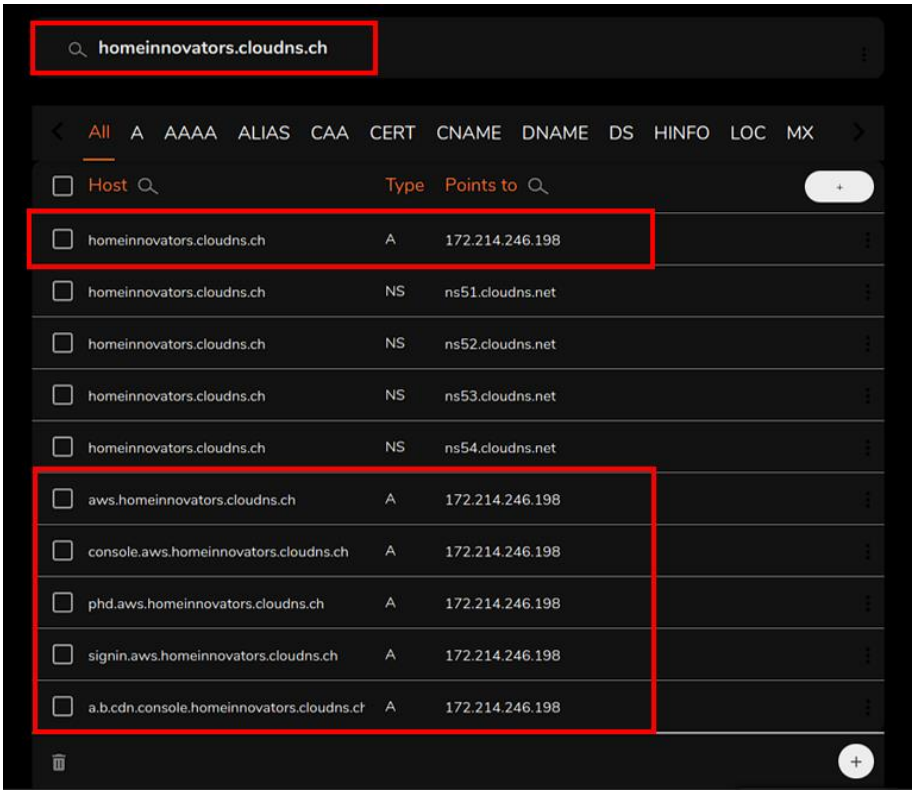
Configuración de registros



La Figura 111 muestra el conjunto de registros asociadas al dominio mediante el cual se realizarán las redirecciones al sitio web malicioso, los registros resaltados corresponden al caso de redirección desde el sitio web de Amazon.

Figura 111

Detalles del dominio y subdominios

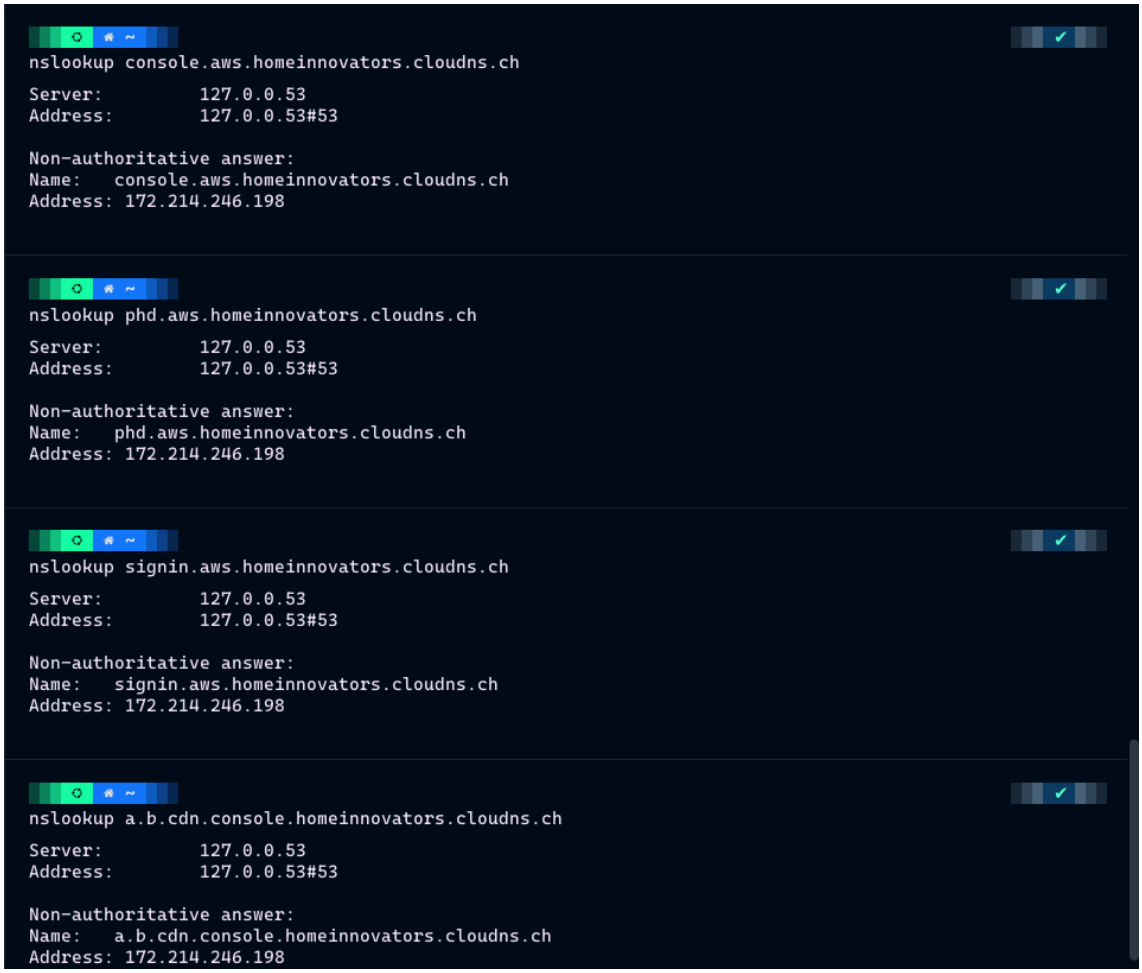


Posterior a la creación del dominio y subdominios se constata el funcionamiento del DNS por medio de consultas especificas hacia los registros configurados en CloudDNS a través del comando ***nsloookup***.

La Figura 112 indica las consultas realizadas mediante el terminal con destino a los registros establecidos para el dominio *homeinnovators.cloudns.ch.*

Figura 112

Verificación de resolución de dominios

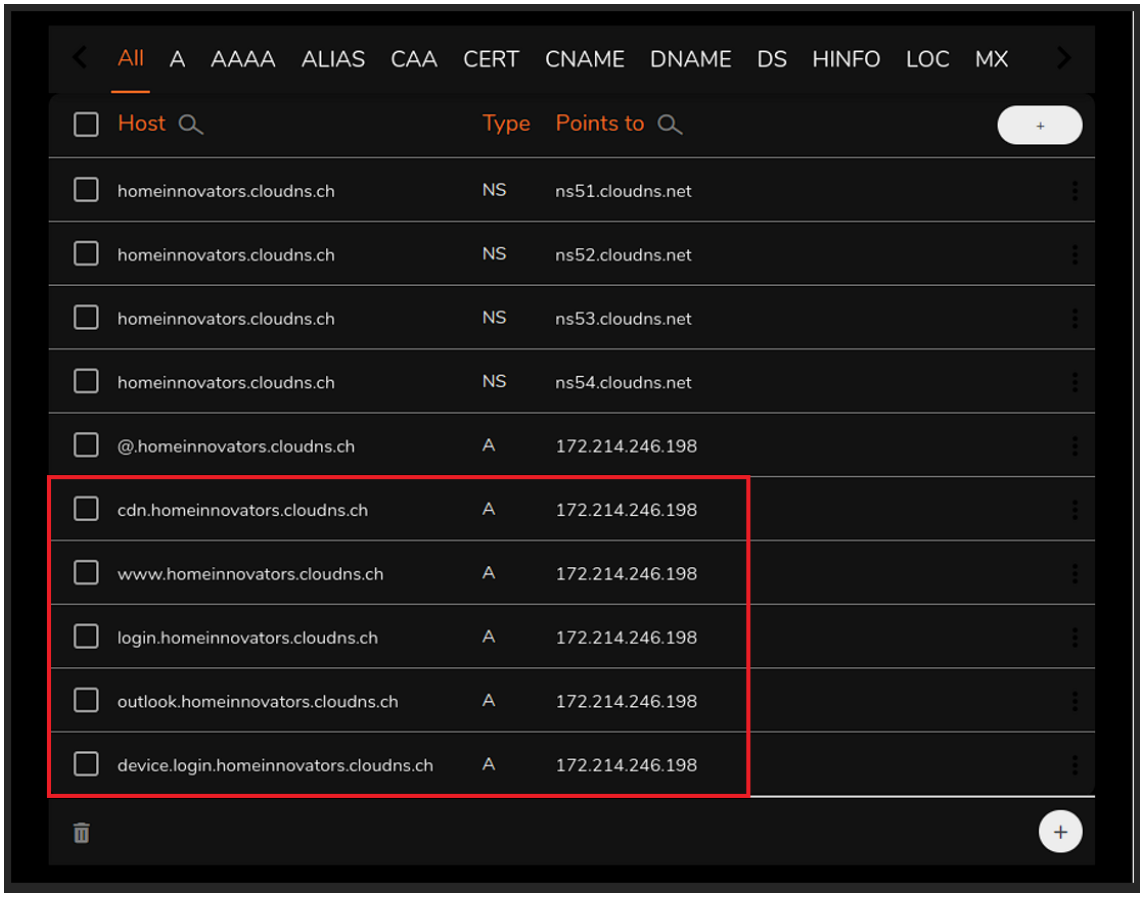


4.3.3 Acceso Inicial

El desarrollo del siguiente apartado precisa de los registros asociados al dominio “homeinnnovators.cloudns.ch”, los cuales fueron configurados en función de los procesos de redirección de Outlook (ver Figura 113).

Figura 113

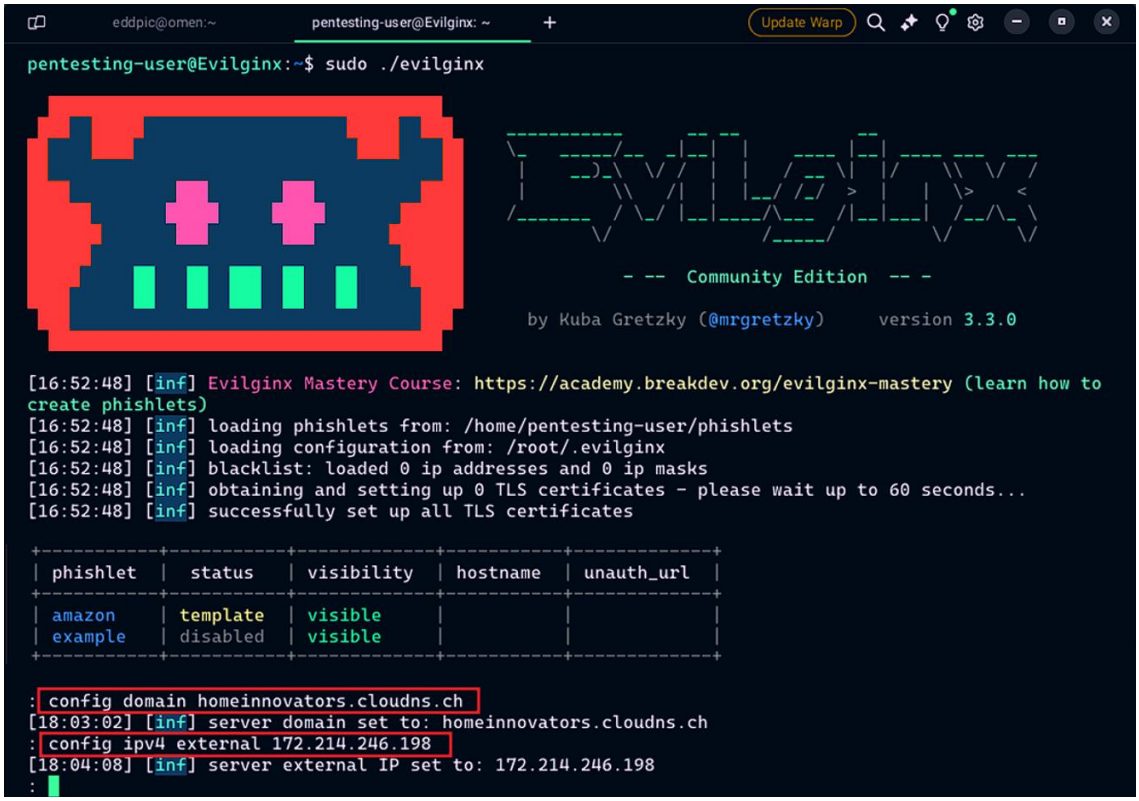
Configuración de registros y subdominios para Outlook



Como primer paso se asocia el dominio fraudulento creado en el apartado anterior y se establece la dirección IP con la cual el servidor Evilginx interactuara con los clientes, por medio de los comandos de ***config domain homeinnovators.cloudns.ch*** y ***config ipv4 external 172.214.246.198*** (ver Figura 114).

Figura 114

Asociación del dominio fraudulento en Evilginx

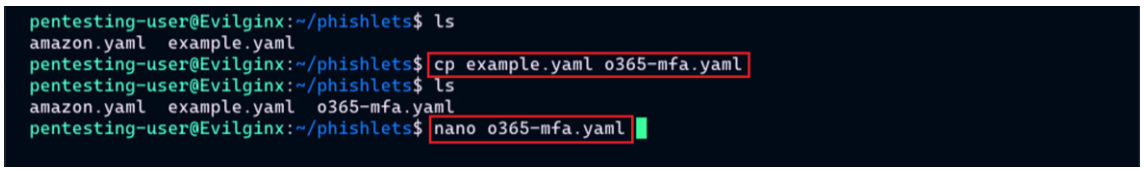


En el siguiente paso se crea un nuevo señuelo para el servicio de correo de Outlook a partir de la plantilla de ejemplo que viene con el archivo de instalación de Evilginx.

La Figura 115 muestra la creación del archivo de configuración que será empleado en los procesos de generación del nuevo señuelo.

Figura 115

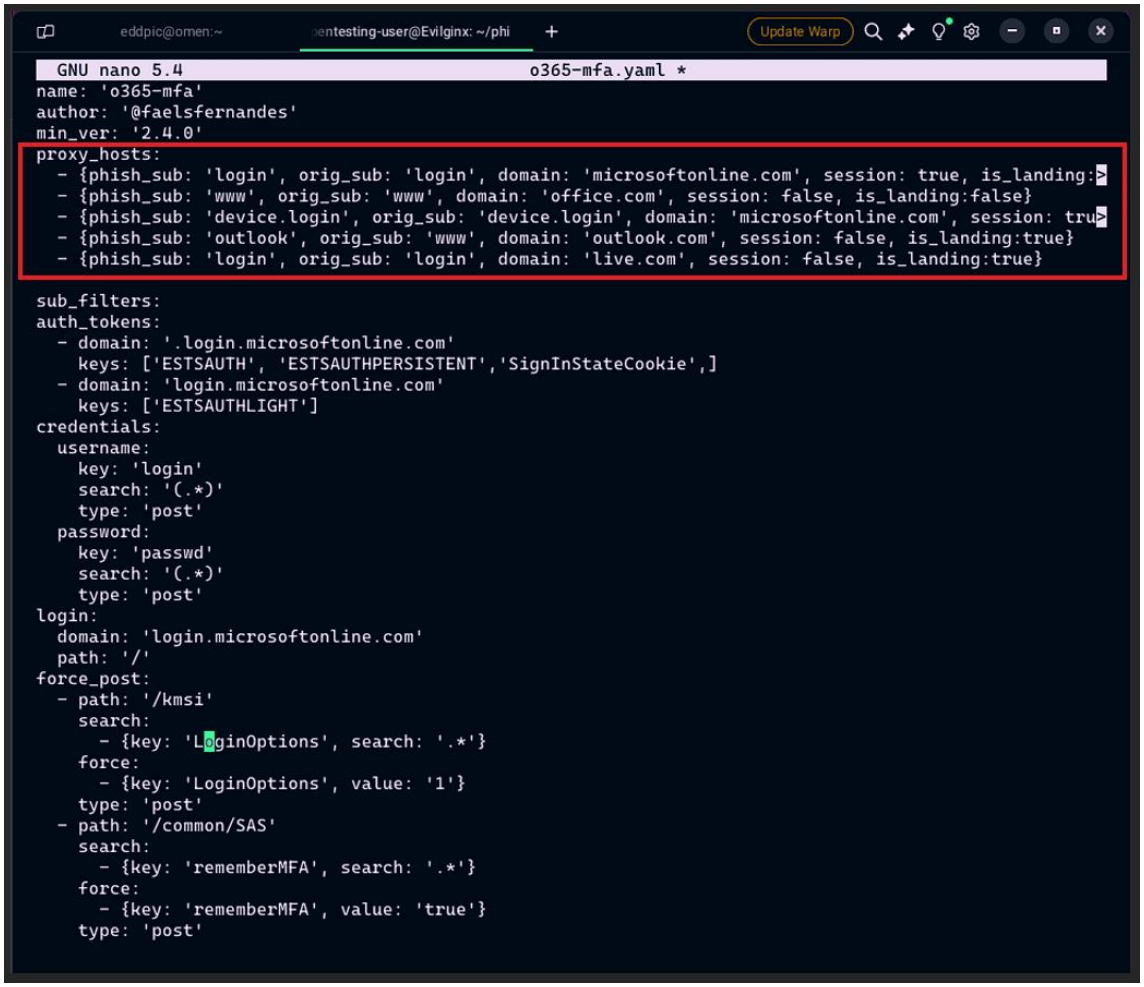
Creación del señuelo



Por medio del editor de texto se anexan los subdominios del DNS, de esta manera se intercepta y redirige el tráfico entre el objetivo y el sitio legítimo sin levantar sospechas (ver Figura 116).

Figura 116

Configuración del señuelo



Una vez configurado el señuelo se enlaza el dominio, de esta manera Evilginx redirigirá, gestionará las solicitudes, capturará las credenciales además de que presentará la réplica del sitio legítimo (ver Figura 117).

Figura 117

Asociación del dominio al señuelo

Finalmente, se pone en funcionamiento el phishlet por medio del comando ***enable***, de esta manera se procede a la verificación de los certificados TLS que serán asociados al enlace malicioso.

Figura 118

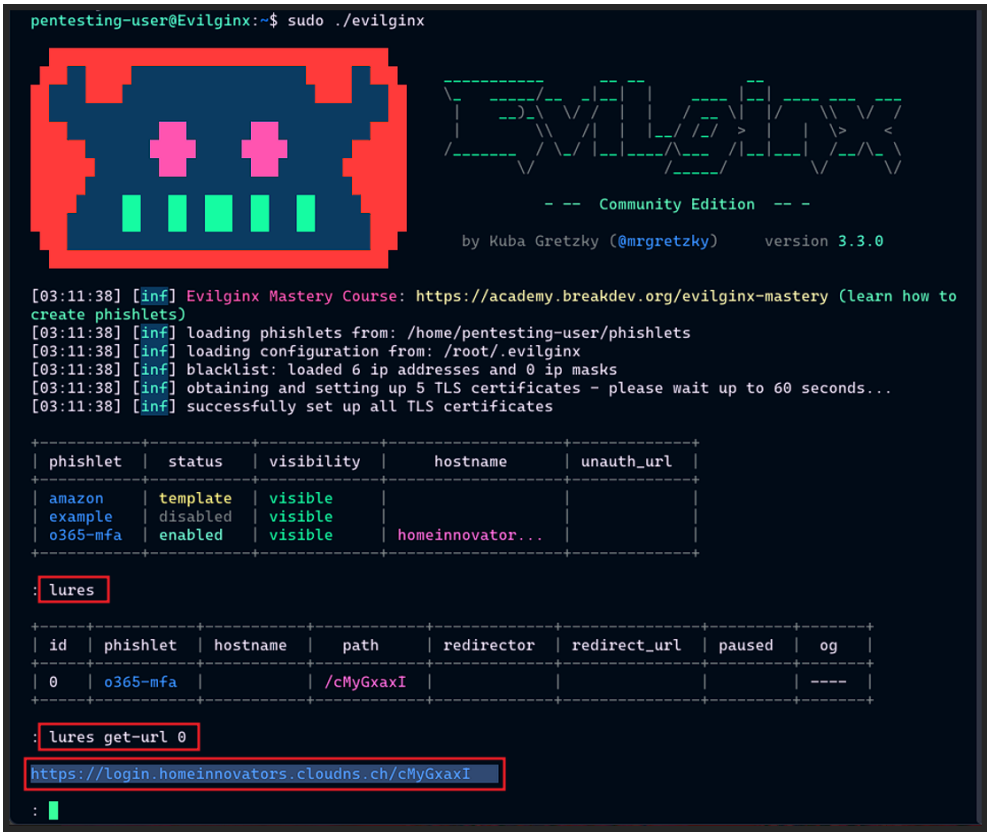
Habilitación del señuelo

Una vez configurado el phishlet se generan los enlaces URL que serán enviados a las víctimas potenciales a través del comando ***lures***.

La Figura 119 muestra el proceso de generación del señuelo y su respectivo enlace malicioso, el cual podrá ser empleado en los procesos de robo de información.

Figura 119

Generación de enlace URL fraudulento

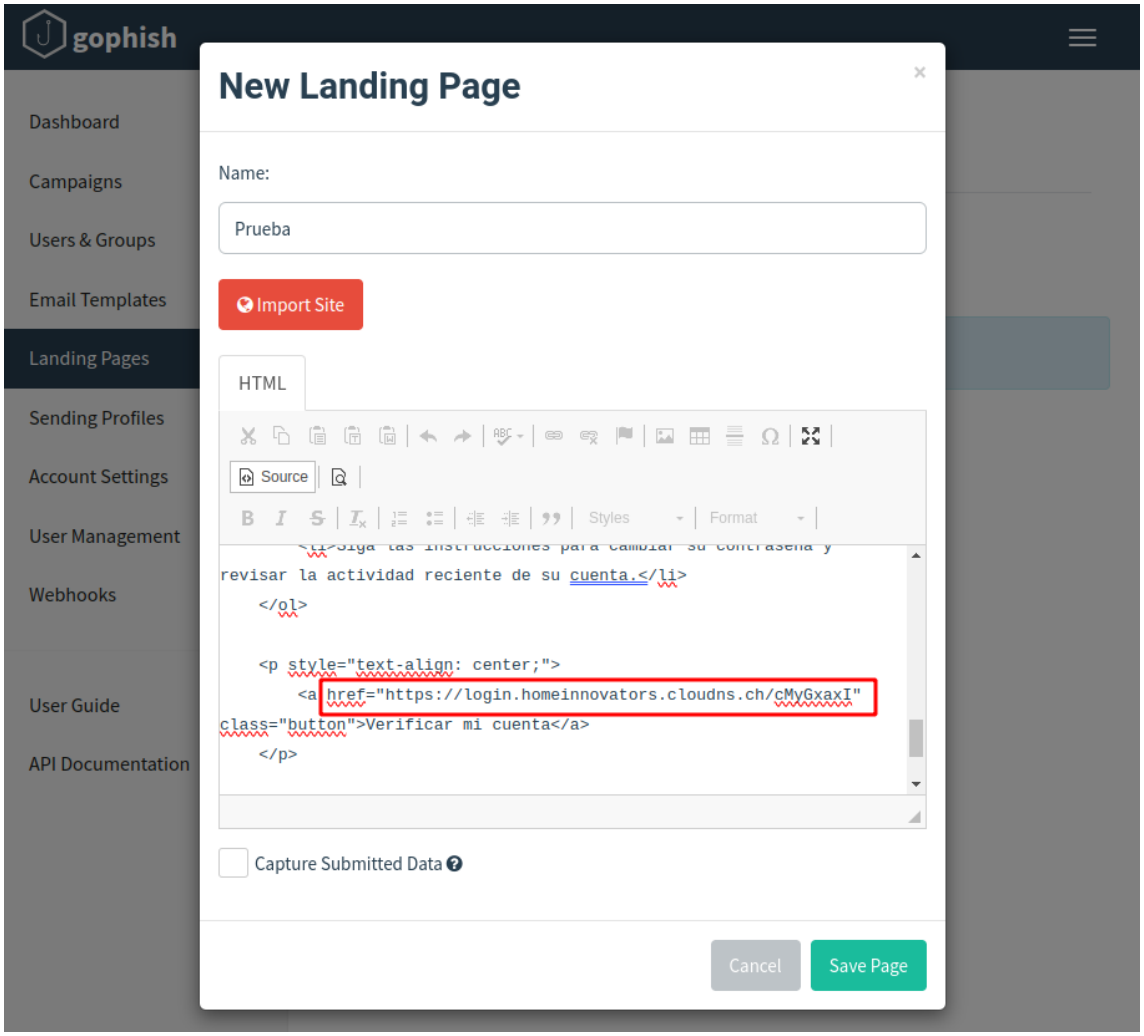


En la siguiente sección se emplea Gophish como intermediario en el proceso de despliegue, se crea el formato HTML del correo en el cual se incrusta el enlace URL generado por Evilginx.

La Figura 120 señala el proceso de adición del enlace malicioso en el código del correo en formato HTML.

Figura 120

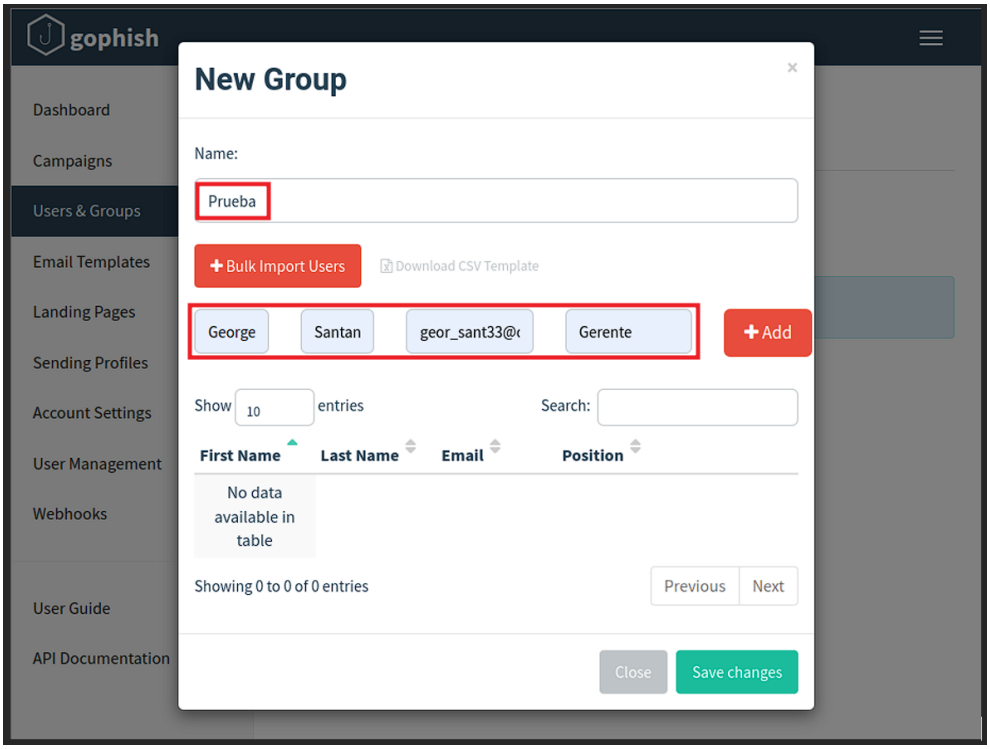
Creación del sitio web del correo electrónico



Posteriormente, se crea el grupo en el que se añadirán las víctimas potenciales junto con su respectivo correo electrónico (ver Figura 121), el cual se empleara en los futuros procesos de despliegue de la campaña de phishing.

Figura 121

Creación de grupo de objetivos

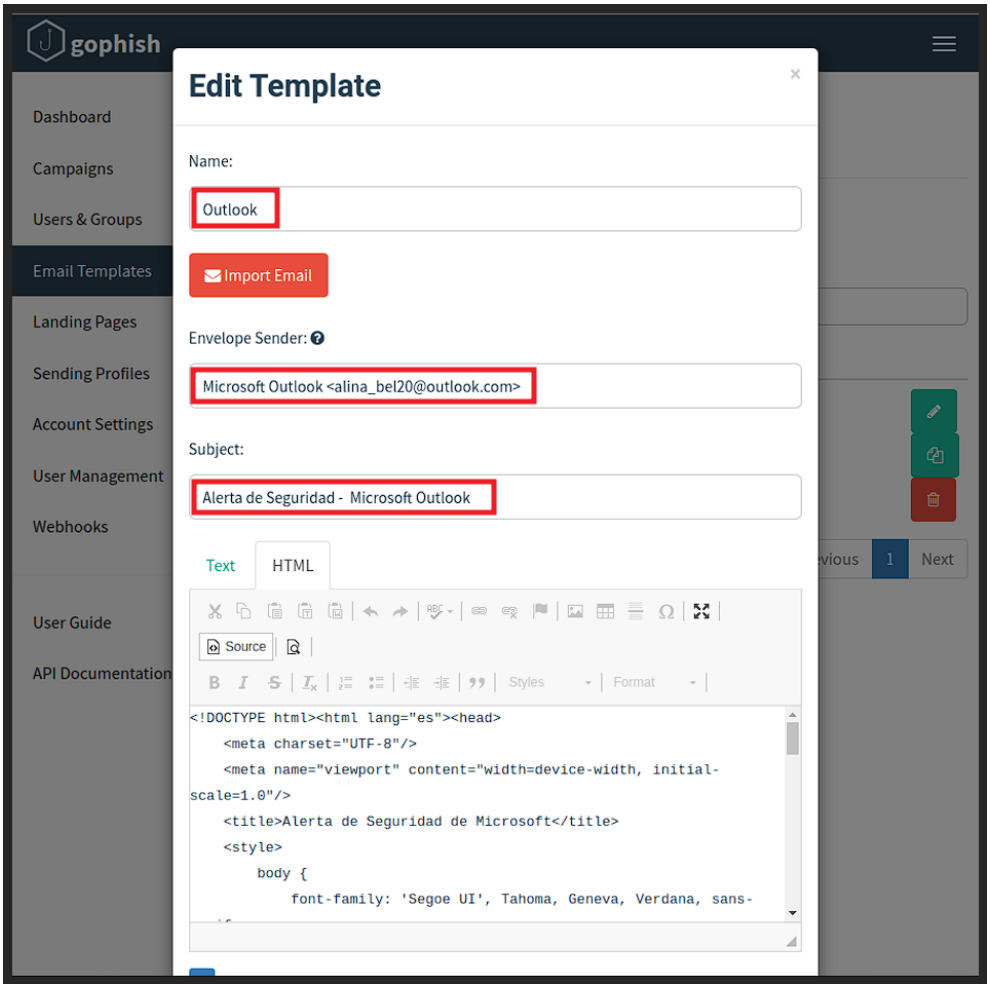


4.3.4 Ejecución

Este apartado materializa el ataque simulado haciendo uso de las herramientas y técnicas desarrolladas anteriormente, en base a las configuraciones y parámetros establecidos en Gophish se modela el patrón del correo electrónico en la cual se establecen los campos de asunto como si se tratase de una advertencia de seguridad acerca de la cuenta personal de la víctima, a la vez que en el campo de texto se anexa el código en lenguaje HTML de la página web fraudulenta (ver Figura 122).

Figura 122

Configuración de la plantilla de correo electrónico

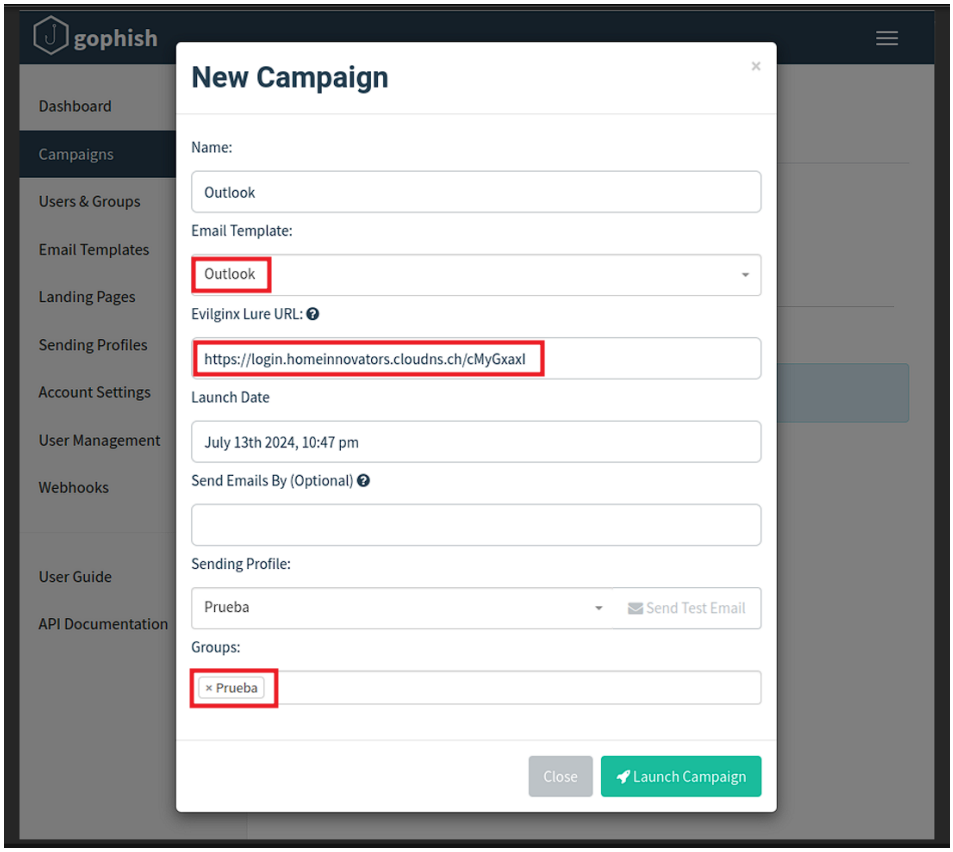


Para el despliegue de la campaña se toma la información y parámetros configurados de Gophish acerca del apartado del patrón de correo electrónico, así como del grupo objetivo, además se vincula de manera externa el enlace URL generado por la herramienta de Evilginx.

La Figura 123 detalla el proceso de generación de la campaña de phishing, en la cual se asocia los parámetros configurados tanto de Gophish como de Evilginx.

Figura 123

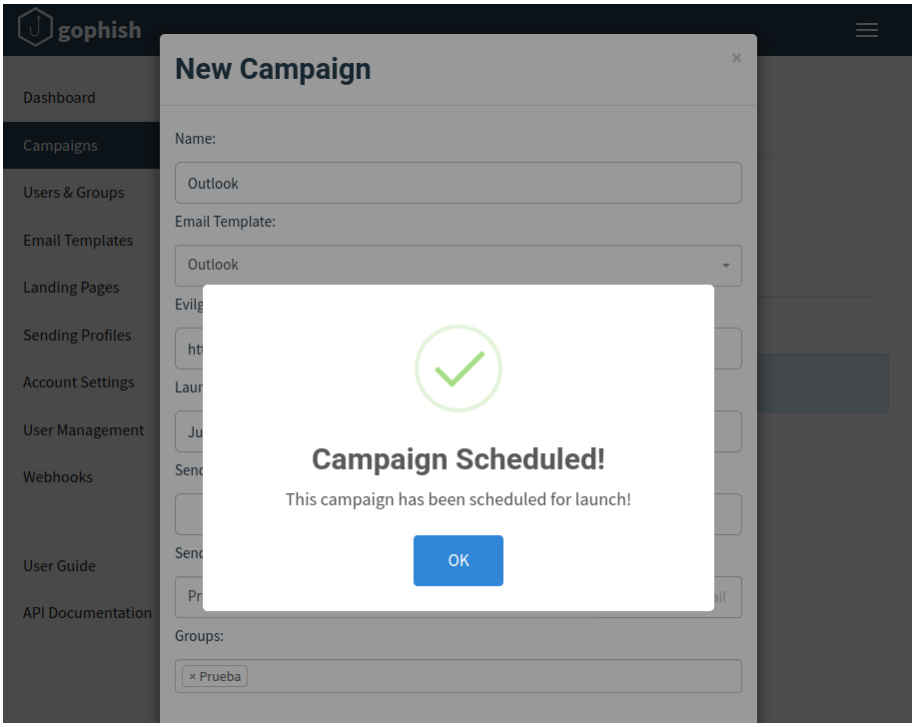
Creación de campaña de phishing



La Figura 124 señala el punto final del proceso de despliegue de la campaña de phishing por medio de correo electrónico malicioso.

Figura 124

Despliegue de campaña de phishing

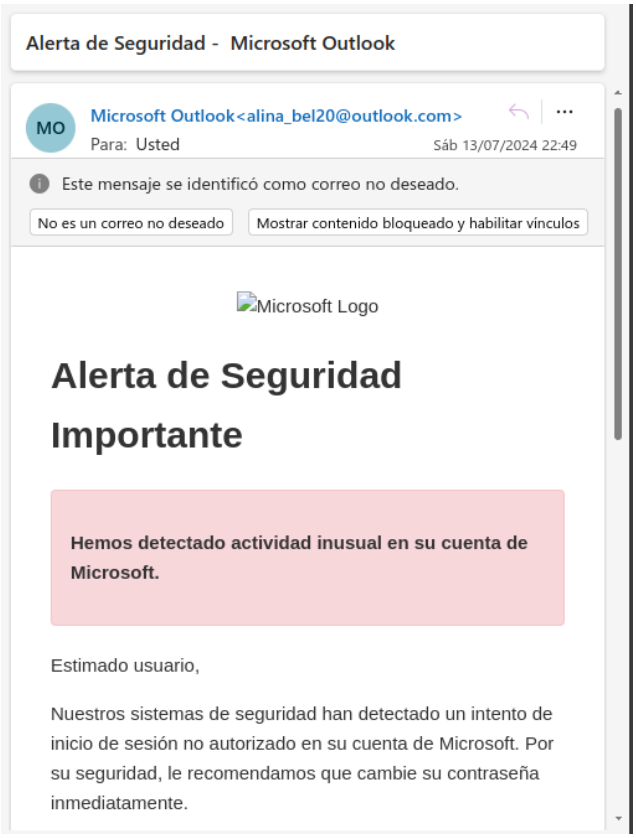


Una vez lanzada la campaña en función de los perfiles, en las cuentas de los objetivos se manifestará el correo fraudulento, en el cual se alerta de la posible irrupción de su cuenta además de que se le aconseja de que ejecute las medidas de prevención necesarias, bajo este concepto se trata de persuadir el ingreso al enlace que lleva como adjunto el correo, para de esta manera realizar el robo de información.

La Figura 125 ejemplifica la llegada del correo electrónico malicioso al buzón de correos de la víctima.

Figura 125

Recepción del correo phishing



4.3.5 Persistencia y Escalada

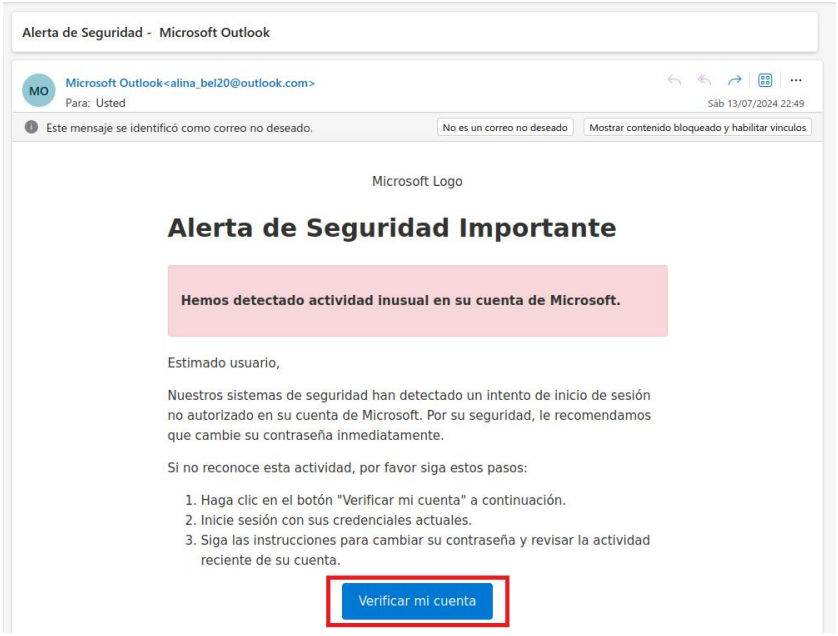
En esta fase se emplea la información del objetivo adquirida tales como usuario y contraseña asociada a los servicios legítimos de correo para de esta manera prolongar la estancia en la red sin que se genera indicios de presencia por parte del atacante.

Este apartado se sitúa en el caso de que la víctima accede al enlace url haciendo caso omiso de que se trata de algo falso y de manera incrédula ingresa la información de la cuenta esto sumado a que la página de login se asemeja casi en su totalidad al del sitio original con excepción del dominio, el cual puede pasar desapercibido si no se es consciente de los pequeños detalles que se asocian a la página oficial.

La Figura 126 muestra el paso inicial correspondiente al ingreso al sitio web falso desde el enlace incrustado en el correo electrónico.

Figura 126

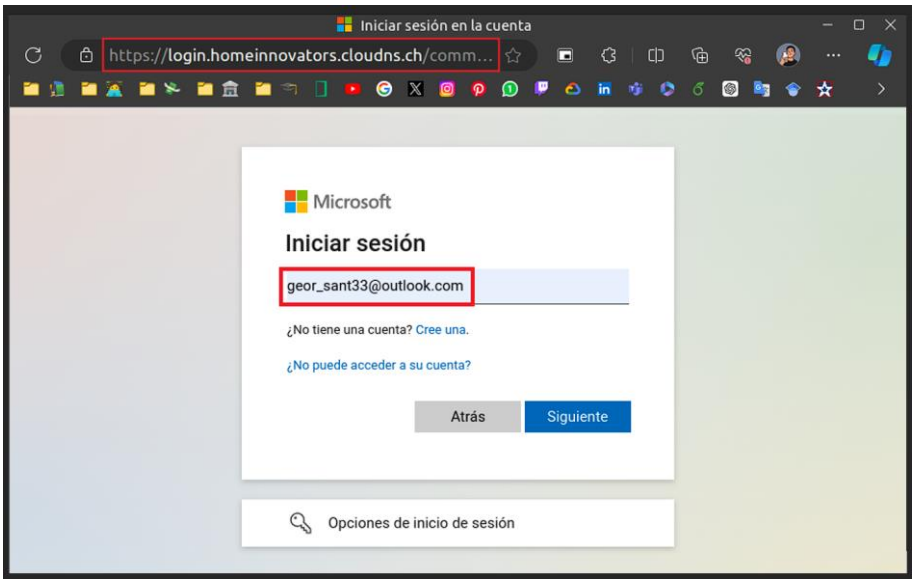
Detalles del correo phishing



La Figura 127 ejemplifica el siguiente paso dentro del proceso de adquisición de credenciales en donde la víctima ingresa las credenciales en este caso del servicio de correo de Outlook.

Figura 127

Ingreso de credenciales

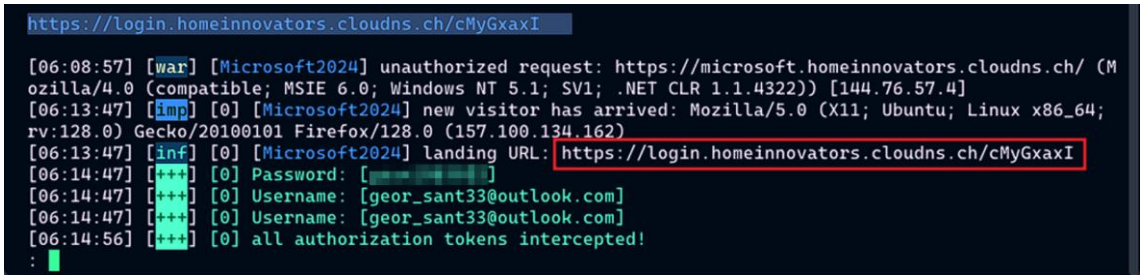


4.3.6 Acceso a Credenciales

En el siguiente paso y posterior al ingreso de credenciales de la víctima, desde el punto de vista del atacante, la herramienta de Evilginx captura las credenciales de acceso del objetivo (ver Figura 128) y de esta manera da rienda suelta a su uso de la manera que le plazca, a la vez que puede hacer uso de esta para aumentar la red de víctimas.

Figura 128

Captura de credenciales de acceso



4.4 Pruebas

En esta sección se desarrolla con el propósito de evidenciar el correcto funcionamiento de la red IoT generada con Mininet-WiFi, así como del modelo de detección phishing, en este sentido se establecen dos tipos de pruebas: pruebas funcionales y pruebas específicas.

En la Tabla 26 se detalla el esquema de pruebas a ejecutar para verificar la funcionalidad de cada uno en función de su tipo.

Tabla 26

Plan de Pruebas del sistema de detección de phishing

Tipo de Prueba	Descripción	Resultado
Pruebas Funcionales	Prueba 01F. Verificar la conectividad de red	Los nodos de la red simulada cuentan con sus

		respectivos canales de comunicación
	Prueba 02F. Conexión de los nodos con el servidor CoAP	Envío y recepción de los datos generados por los nodos cliente (sensores y dispositivos IoT) hacia el nodo servidor CoAP (servidor local)
	Prueba 03F. Acceso al cliente de correo desde el nodo "host1" de la red IoT	Ejecución del cliente de correo con las configuraciones de red del nodo "host1"
	Prueba 04F. Almacenamiento y Representación de Datos	Publicación de las métricas en InfluxDB Cloud, así como su representación en Grafana Cloud por medio de los identificativos.
	Prueba 05F. Conexión y extracción de los detalles del correo (Contenido y Encabezado)	Funcionamiento del inicio de sesión en el servicio de correo por medio de la API de Outlook, a la vez que se adquieren los correos del buzón

	Prueba 06F. Cálculo de la probabilidad de phishing y generación de reglas.	Clasificación del correo en función del tipo para un posterior uso de los enlaces maliciosos en el proceso de generación de reglas de Suricata
Pruebas Específicas	Prueba 01E. Prueba sin el sistema activo	Emulación de acciones que se suscitan en procesos de robo información por medio de correos phishing
	Prueba 02E. Prueba con el sistema activo	Funcionamiento de todos los subprocesos asociados con la detección de phishing

4.4.1 Pruebas Funcionales

El siguiente apartado se enfoca en la verificación de la funcionalidad de la topología de red implementada. Este proceso incluye la evaluación de los procedimientos operativos y las diversas herramientas que se integran en la simulación de la red. El análisis se llevará a cabo considerando los parámetros de configuración establecidos en los capítulos precedentes, asegurando de esta manera la relación y eficacia del sistema en todo su conjunto.

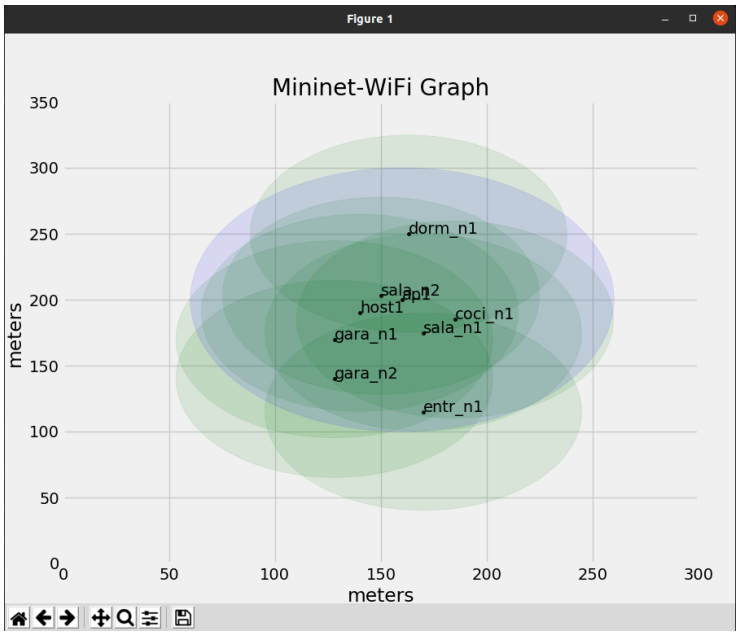
4.4.1.1 Prueba 01F. Verificar la conectividad de red

Una vez configurada la topología y el sistema de detección, se procede a ejecutar del script correspondiente mediante Mininet-WiFi. La comunicación entre los diversos nodos con el punto de acceso se verifica a través de la gráfica de la topología resultante.

La Figura 129 muestra la representación gráfica de los diversos nodos que conforman la topología de red, incluyendo el punto de acceso. El parámetro de verificación empleado corresponde a la cobertura de señal emitida tanto por el punto de acceso, así como por los nodos individuales. La verificación se fundamenta en las variables específicas del modelo de propagación aplicable a espacios interiores. En este contexto, se considera que existe una conexión efectiva cuando los dispositivos se encuentra dentro del rango de proximidad que permita una comunicación fiable entre ellos, de acuerdo con los umbrales que permite el modelo mencionado.

Figura 129

Simulación de red IoT



El apartado de conectividad también se constata por medio de la terminal de comandos de Mininet-WiFi, donde a través de comandos específicos se comprueba la

existencia de los enlaces entre los nodos con el punto de acceso, el cual incluye tanto enlaces inalámbricos como alámbricos.

La Figura 130 presenta los detalles de los enlaces de red entre los dispositivos en la topología de Mininet-WiFi. Asimismo, la Figura 131 ofrece una visión más detallada de la topología red simulada, mostrando los nodos y las conexiones entre ellos.

Figura 130

Detalles generales de la red

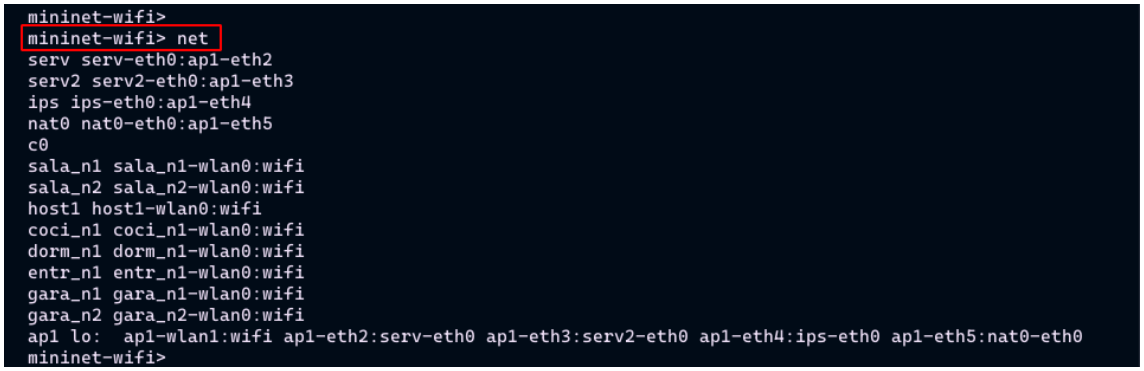
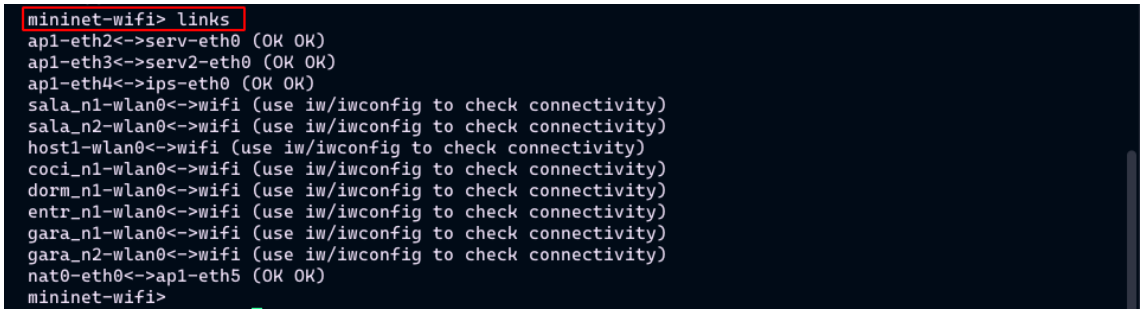


Figura 131

Detalles de enlaces de red



Para continuar con el proceso de verificación de conectividad, se realiza el envío de paquetes ICMP desde cada nodo hacia todos los demás nodos, con el objetivo de constatar la existencia de las rutas de comunicación entre ellos. El procedimiento se lleva a cabo utilizando el comando "***pingall***". Los detalles de este proceso se presentan en la Figura 132 y Figura 133.

Figura 132

Pruebas de interconexión de dispositivos – Parte 1

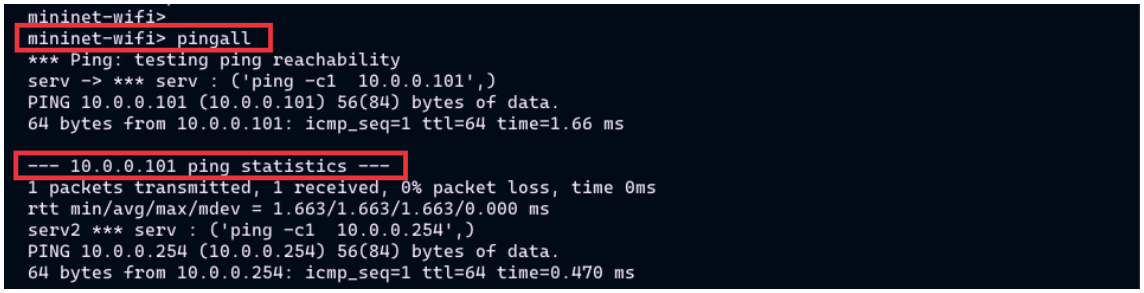
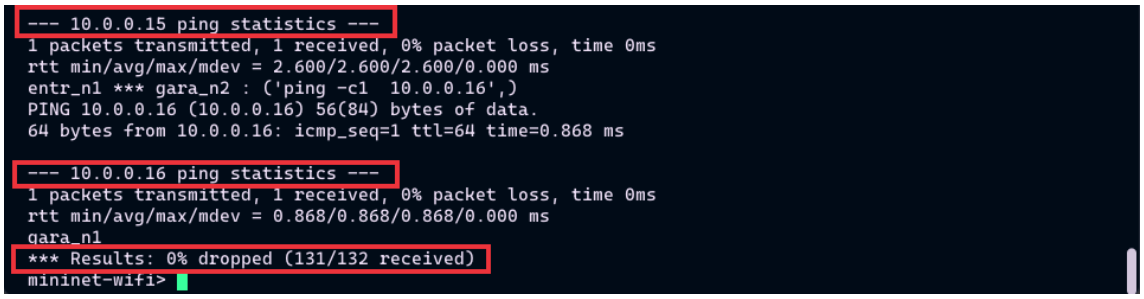


Figura 133

Pruebas de interconexión de dispositivos – Parte 2



4.4.1.2 Prueba 02F. Conexión de los nodos con el servidor CoAP

En la siguiente sección, se muestra el funcionamiento del proceso de comunicación mediante el protocolo CoAP (Constrained Application Protocol). De esta manera es posible evidenciar la transmisión de datos desde el nodo que actúa como sensor en la topología propuesta. La verificación de este proceso se evidencia por medio de dos métodos: la recepción de los mensajes en los terminales y la examinación de los paquetes mediante la herramienta de Wireshark.

La Figura 134 y Figura 135 muestra el proceso de transmisión, así como la recepción de datos entre el servidor y cliente por medio del protocolo CoAP.

Figura 134

Comunicación - Servidor CoAP

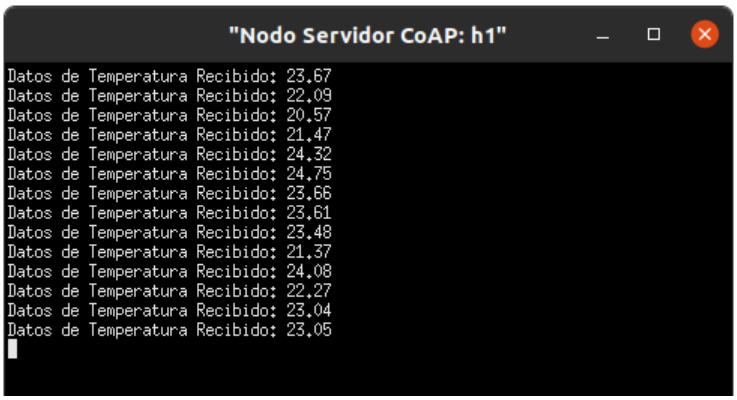
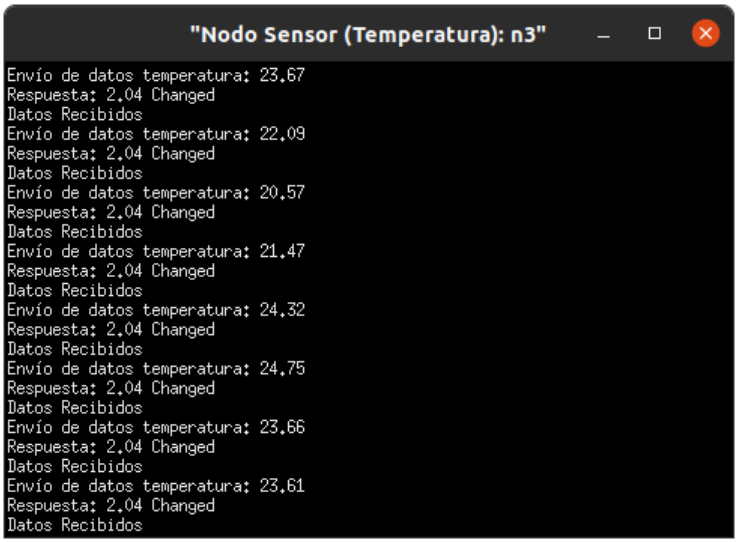


Figura 135

Comunicación - Cliente CoAP



En la siguiente sección se muestran los detalles de los paquetes de tráfico capturados en el proceso de comunicación. El formato utilizado en la comunicación mediante CoAP abarca los tipos de solicitudes y respuestas. La recepción de los mensajes se verifica a través de un identificador único, que se confirma a través de un marcador de confirmación y su correspondiente acuse de recibo. Cabe destacar que el identificador es el mismo tanto en el paquete de solicitud como en el de respuesta.

Los detalles del paquete del mensaje confirmable y del acuse de recibo, mencionados anteriormente, se muestran en la Figura 136 así como en la Figura 137, respectivamente.

Figura 136

Paquete de Mensaje Confirmable (CON)

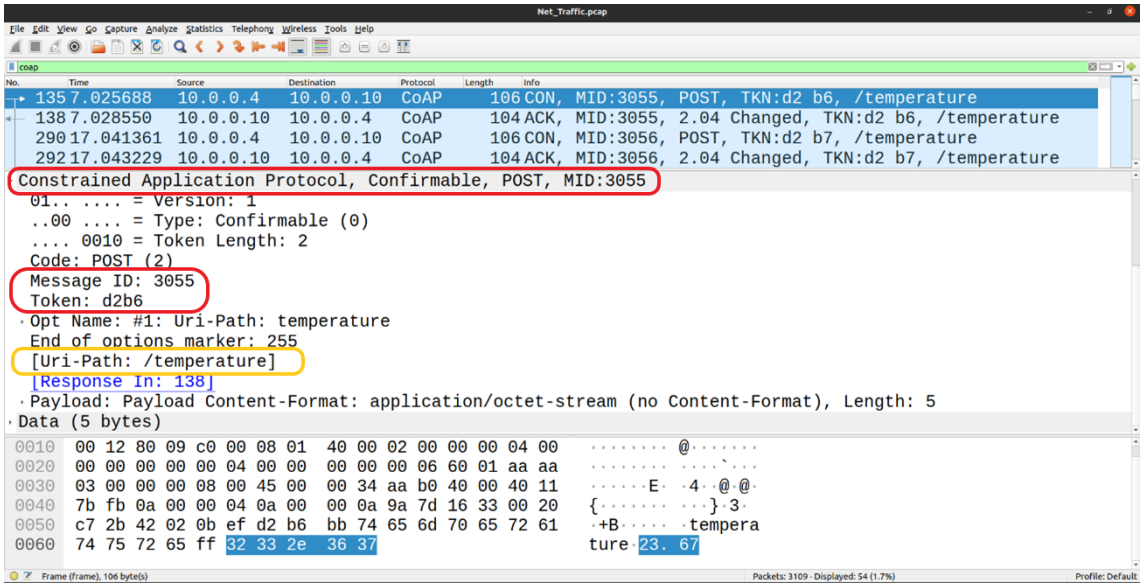
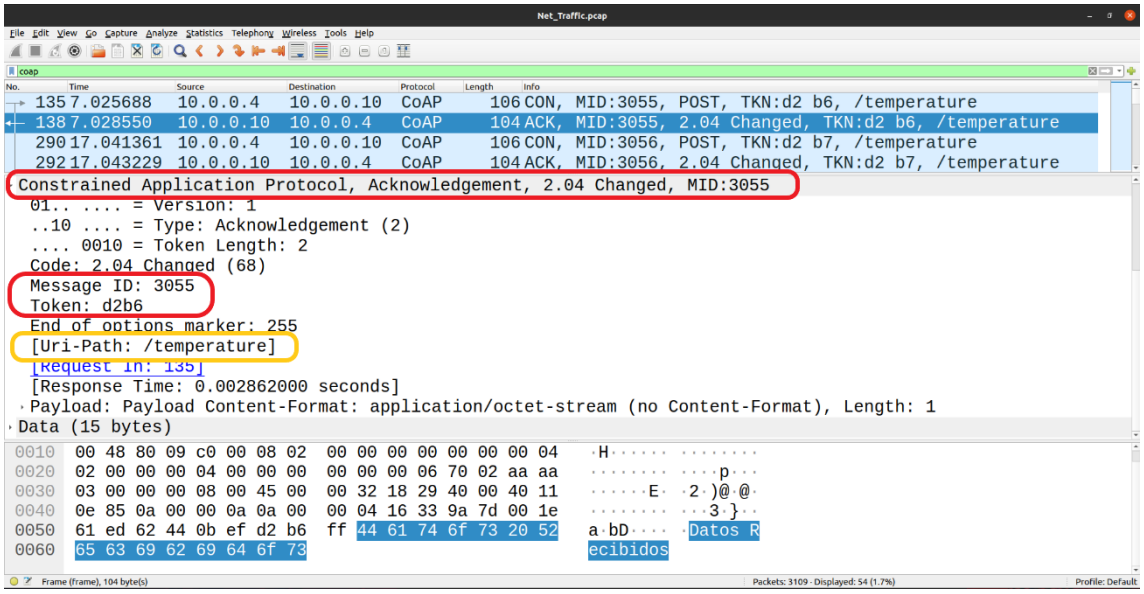


Figura 137

Paquete de Acuse de Recibo (ACK)



4.4.1.3 Prueba 03F. Acceso al cliente de correo desde el nodo “host1” de la red IoT

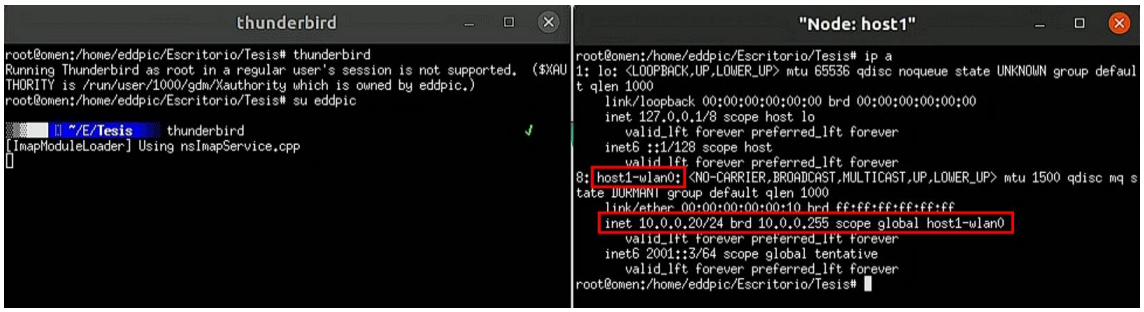
En concordancia con los objetivos del trabajo el despliegue del sistema, así como la interacción entre nodos y la apertura de recursos de software se los ejecuta desde los nodos de la red.

En este sentido se constata que las configuraciones de red de los nodos se asocien a los programas que se ejecutan de manera interna desde la red IoT generada por Mininet-WiFi.

La Figura 138 señala la apertura del cliente de correo de Thunderbird desde el nodo “host1” de la red IoT, de esta manera se mantiene el tráfico generado desde cualquier nodo dentro de la red simulada por Mininet-WiFi.

Figura 138

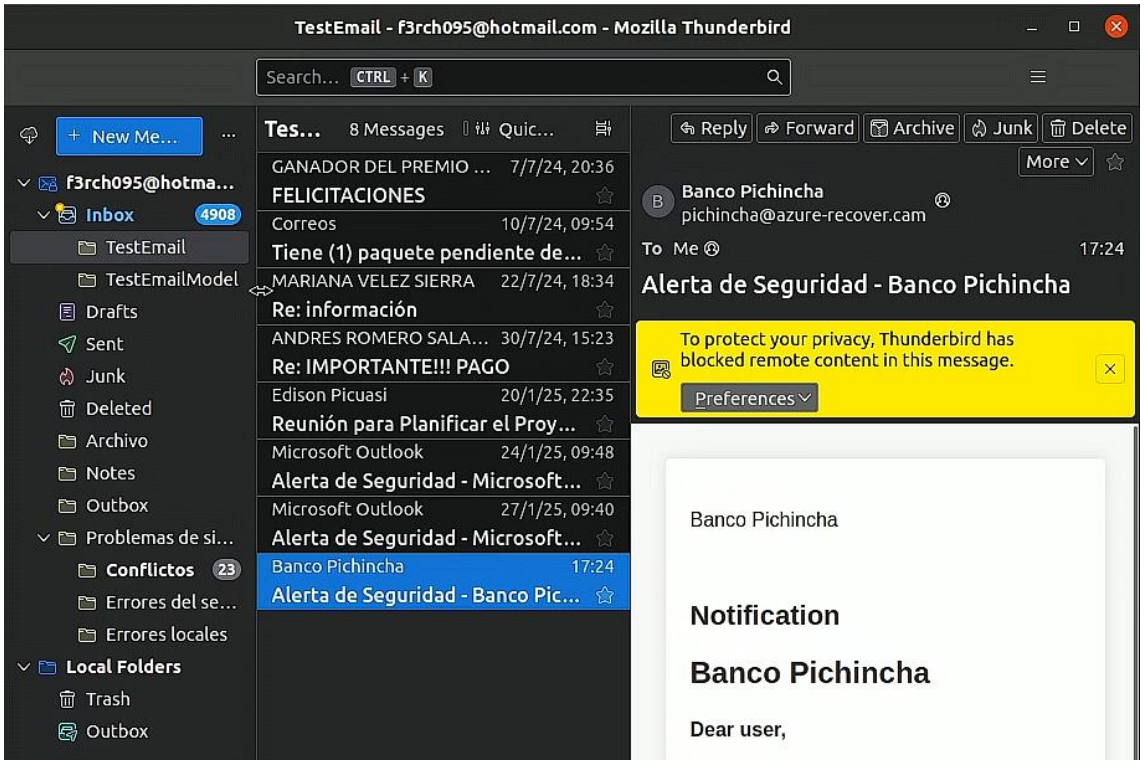
Verificación de dirección IP y apertura de Thunderbird



En el siguiente paso se observa como el cliente empieza a descargar todos los correos alojados en el servidor.

La Figura 139 muestra la ejecución de Thunderbird, desde el punto de vista de la red, el programa emplea las configuraciones del nodo, sin embargo, los procesos de enrutamiento también son orquestados por el protocolo de OpenFlow el cual aísla el plano de datos del resto de planos.

Figura 139
Inicialización del cliente de correo desde el nodo "host1"

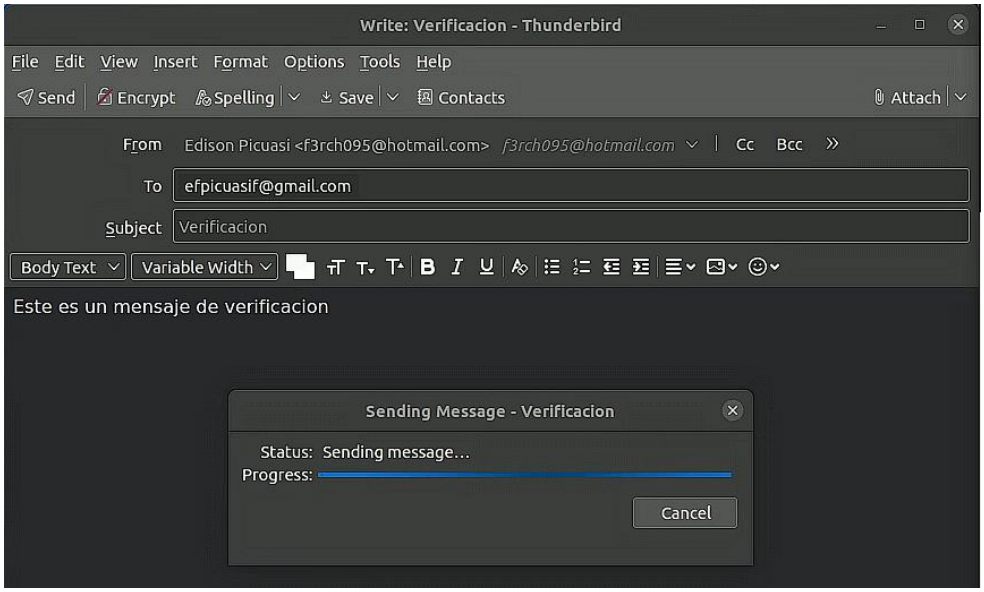


Para constatar que el recurso asociado emplea las configuraciones heredadas por el nodo se realiza un envío de correo desde el cliente, conforme se realice dicha acción los detalles serán capturados por el analizador de paquetes de red Wireshark.

La Figura 140 muestra la creación y envío de correo desde el cliente Wireshark hacia un punto fuera de la red.

Figura 140

Envío de correo

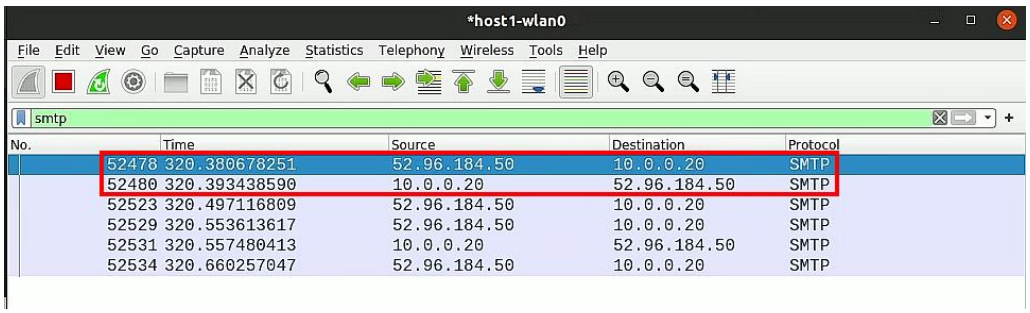


Finalizado el proceso de envío los detalles de trabajo referente al envío de correo serán registrados por Wireshark, en donde se puede observar cómo los detalles de la IP se asocian o corresponden con las del nodo y son empleados con la transferencia de información por medio de SMTP.

La Figura 141 muestra la información de los paquetes de tráfico capturados por Wireshark en relación con el protocolo SMTP.

Figura 141

Captura de paquetes SMTP



4.4.1.4 Prueba 04F. Almacenamiento y Representación de Datos

El apartado de monitorización de los datos de los dispositivos se fundamenta en la representación gráfica de los valores registrados por los diversos nodos de la red simulad a lo largo de múltiples instancias de tiempo. La recopilación de datos se lleva a cabo mediante la transmisión de información desde los nodos hacia InfluxDB Cloud, que posteriormente se visualiza en el servicio de Grafana Cloud. La Figura 142 y Figura 143 muestran los detalles de los parámetros sondeados por los nodos de la red y la representación de dichas métricas respectivamente.

Figura 142

Detalles del almacenamiento de datos del nodo foco inteligente

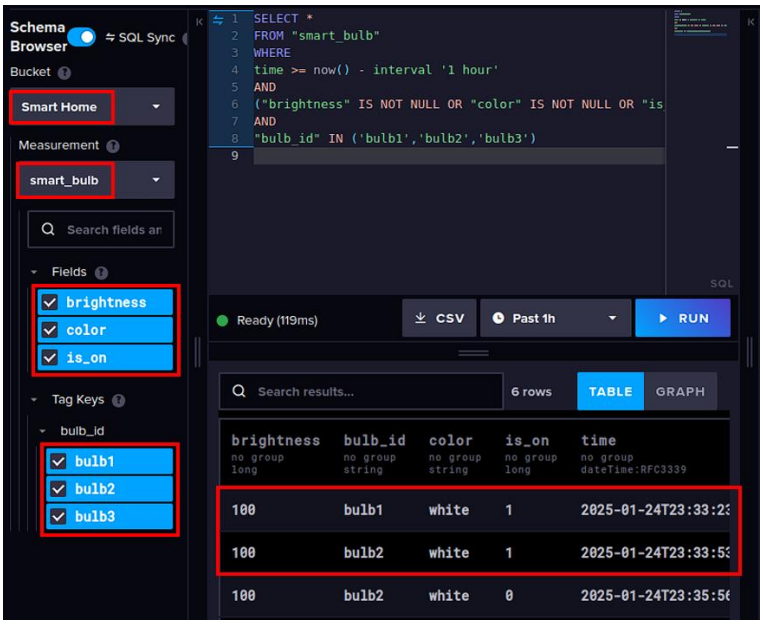


Figura 143

Representación de las métricas generadas por los nodos

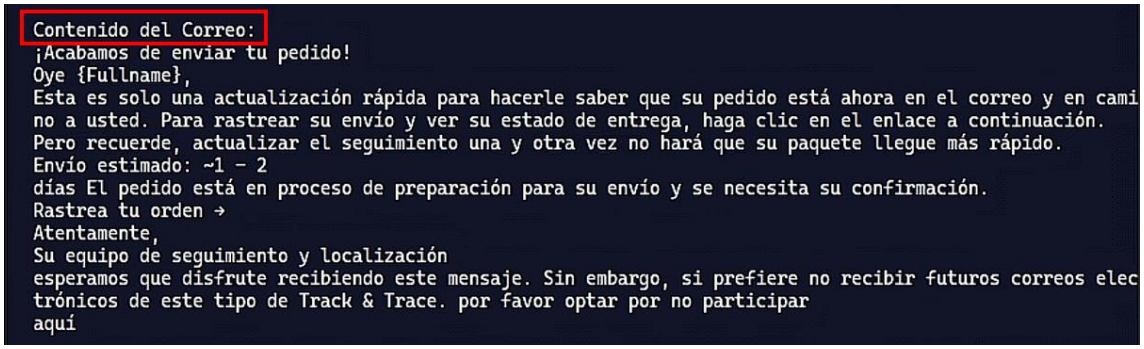
4.4.1.5 Prueba 05F. Conexión y extracción de los detalles del correo (Contenido y Encabezado)

Este apartado se describe el proceso de conexión al servidor de correo electrónico el cual se produce a través de la librería MSAL para Outlook. Tras superar el proceso de autenticación, el sistema accede al directorio establecido para realizar el análisis de los correos. El proceso continúa con la extracción de los detalles de cada correo. En lo referente a la sección del contenido, se realiza un proceso de depuración del texto, de esta manera se aísla el texto plano, eliminando todos los elementos y funciones propias de HTML.

La Figura 144 muestra el resultado del preprocesamiento del contenido del correo, después de aplicar los procesos de eliminación de caracteres especiales, espacios redundantes y etiquetas HMTL.

Figura 144

Detalles del contenido del correo



La Figura 145 resalta los detalles del remitente y asunto extraídos por medio de la librería ***soup*** del correo en formato HTML.

Figura 145

Detalles del encabezado del correo

4.4.1.6 Prueba 06F. Cálculo de la probabilidad de phishing y generación de reglas

Tras la obtención del modelo resultante y configuración completa de la topología, se incorpora el proceso de detección de phishing al nodo que alberga y ejecuta los mecanismos de seguridad, específicamente el Sistema de Detección de Intrusos (IDS). Los procesos de detección, que se ejecutan en segundo plano, se implementan mediante un script cuya ejecución de muestra en la Figura 146. Esta figura presenta los resultados previstos, mostrando los porcentajes de probabilidad de phishing y los detalles del análisis aplicado.

Figura 146

Resultado de detección de ataques phishing

A su vez en la Figura 147 se muestran los detalles de las URLS presentes en el correo clasificado como phishing y que hacen alusión a enlaces maliciosos.

Figura 147

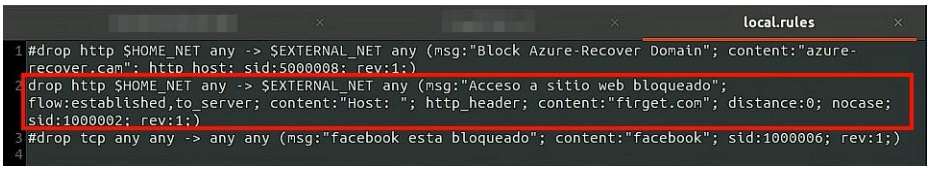
Obtención de enlace maliciosos

Una vez que concluya el análisis de phishing, se genera la regla de Suricata tomando como referencia los enlaces detectados como maliciosos y la plantilla para la regla establecida dentro del script.

La resalta Figura 148 la regla creada durante el proceso de ejecución del script para la detección de phishing, en la que se toma el dominio como parámetro de análisis en el IDS

Figura 148

Creación de regla de Suricata



4.4.2 Pruebas Específicas

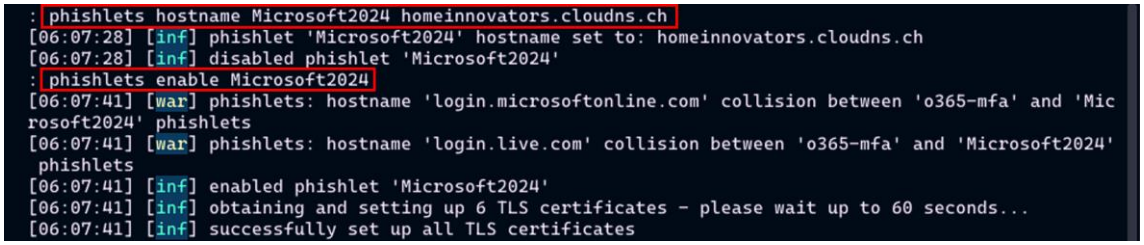
En la sección de pruebas específicas, se evalúa la topología de red en base dos escenarios: con el sistema de detección de phishing activo y sin él. Para llevar a cabo la evaluación de manera satisfactoria, es necesario configurar un entorno que simule el ataque phishing de forma controlada. Esta configuración implica la creación de señuelos mediante herramientas como Evilginx y Gophish. Estas herramientas permiten emular las técnicas empleadas en ataques phishing reales, proporcionando un esquema de pruebas riguroso y realista para evaluar la eficacia del sistema de detección.

4.4.2.1 Configuración del señuelo

El proceso de configuración parte con la creación del señuelo en este caso se empleará el phishlet Microsoft2024 y el dominio alojado en el servicio de DNS, de esta manera se logra emular el servicio de autenticación de Microsoft. Una vez generado el phishlet, se habilita y se obtienen las certificaciones TLS (ver Figura 149).

Figura 149

Configuración de Phishlets

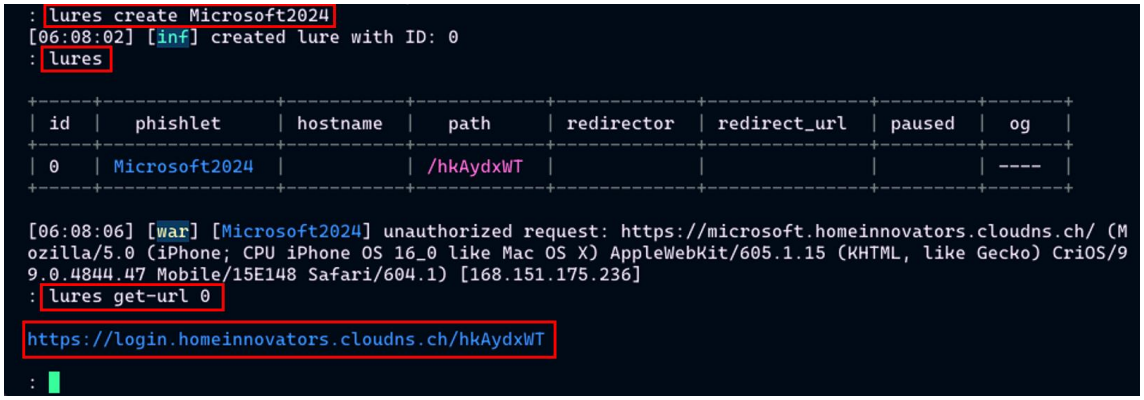


Como paso siguiente se genera el señuelo “lure”, que a breves rasgos son enlaces de phishing que se emplearan en el despliegue de la campaña de phishing. El señuelo enlaza el dominio DNS y el enlace de phishing, por medio del comando lures get-URL [id] se obtiene el enlace completo.

La Figura 150 muestra el procedimiento para generar el enlace malicioso en Evilginx.

Figura 150

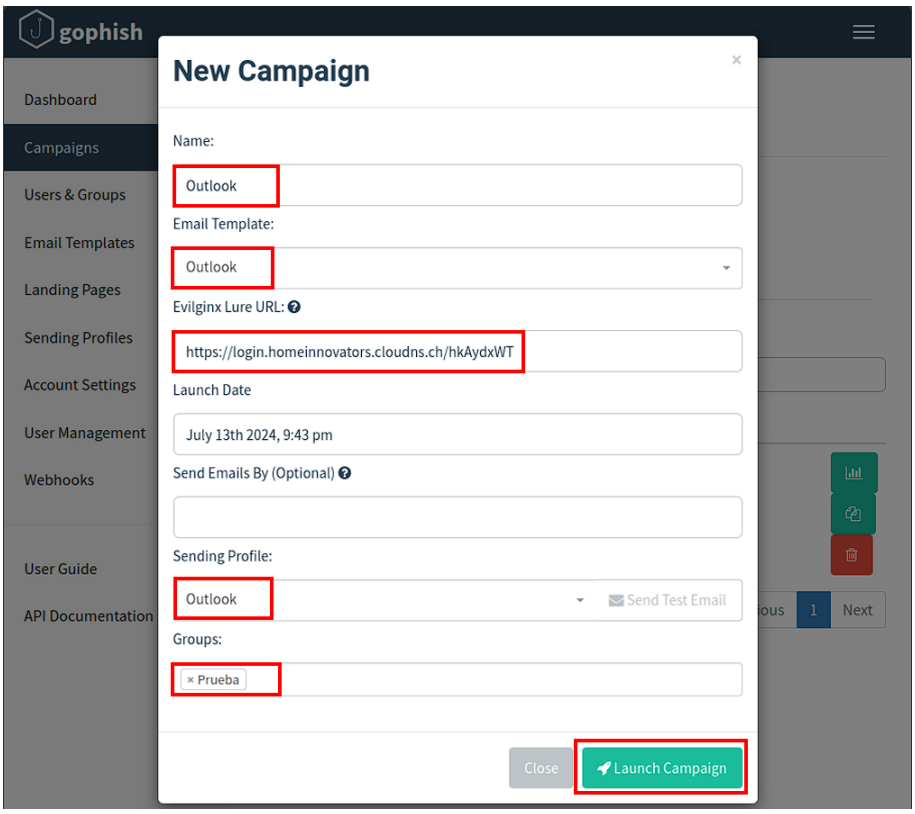
Configuración de Señuelo (Lures)



En el apartado de Gophish se elabora una nueva campaña de phishing en el cual se incrusta la URL obtenida previamente y se rellenan los campos de configuración pertinentes enfocados al objetivo, así como al sitio web falso (ver Figura 151).

Figura 151

Configuración de Campaña de Phishing



En el lado de la víctima se aprecia la recepción del correo phishing que simula una alerta de seguridad típica más realista en el que se hace mención la causa y cómo solucionar el posible problema de seguridad.

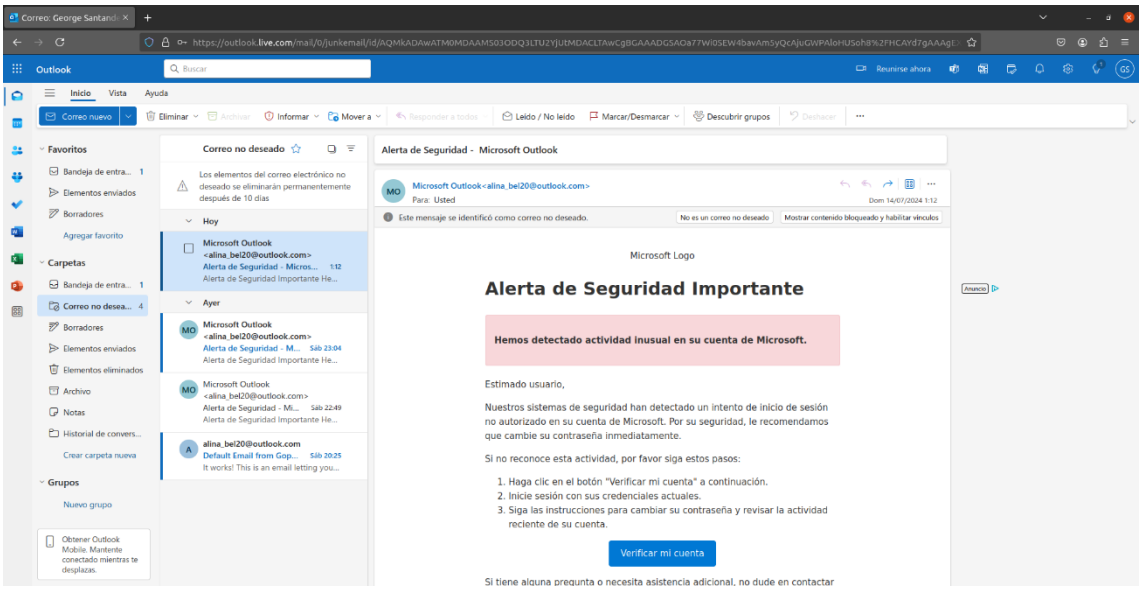
4.4.2.2 Prueba 01E. Prueba sin el sistema activo

Para el caso de prueba se simula la interacción Continuando con la prueba se procede al acceso al URL insertado en el correo, acto seguido se produce una redirección al sitio falso en el cual se solicita las credenciales de acceso (usuario y contraseña).

La Figura 152 ilustra la recepción del correo malicioso desplegado desde la plataforma de Gophish.

Figura 152

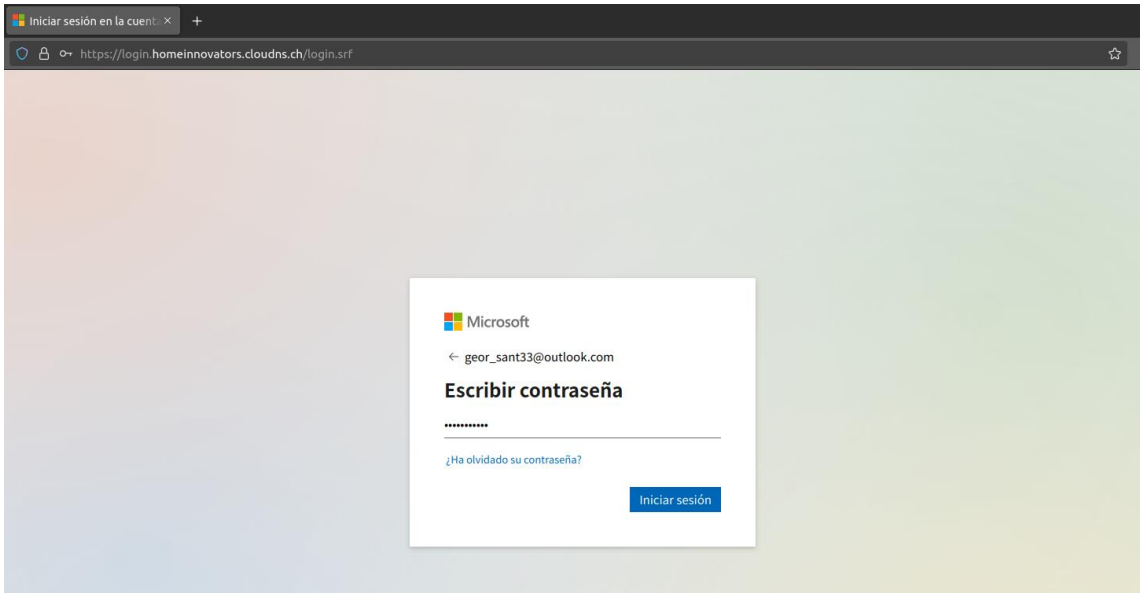
Recepción del correo phishing



En el siguiente paso y en lo que respecta a la simulación de robo de credenciales, se inserta tanto la dirección de correo como la contraseña asociada a dicho servicio (ver Figura 153)

Figura 153

Ingreso de Credenciales - Contraseña



Una vez finalizado el inicio de sesión el atacante una vez más realiza la redirección a la página legítima del servicio de correo, dicho proceso se puede apreciar en la Figura 154.

Figura 154

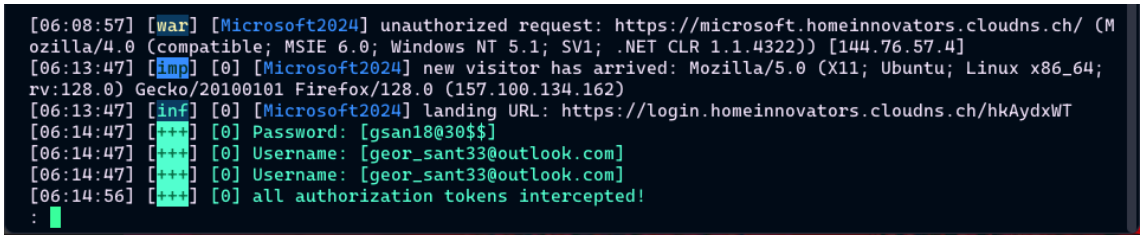
Captura de Credenciales

Posterior al proceso de acceso en el lado del atacante se captura la información de las credenciales de la víctima.

La Figura 155 muestra los valores del nombre de usuario y contraseña capturados en el proceso de acceso por parte de la víctima.

Figura 155

Detalles de Captura de Credenciales



4.4.2.3 Prueba 02E. Prueba con el sistema activo

En este apartado se verifica los resultados de predicción del modelo en 3 escenarios: el primero corresponde a la predicción obtenido durante el proceso de entrenamiento y los dos escenarios restantes reflejan los resultados de las pruebas ejecutadas en base a dos lotes de correos de prueba.

- **Escenario 1 – Entrenamiento**

Por medio de la información de la Tabla 27 se puede determinar que, para el primer escenario, con un total de 2582 muestras analizadas, 1314 fueron clasificadas correctamente como Verdaderos Negativos (VN); lo cual indica que fueron identificadas como correos seguros de manera correcta. Asimismo, se obtuvieron 1202 muestras clasificadas correctamente como Verdaderos Positivos (VP); correspondiente a los correos electrónicos identificados de forma correcta como phishing. Por otro lado, se registraron un número de 45 casos catalogados como Falsos Positivos (FP), es decir correos seguros que fueron identificados erróneamente como phishing, y 21 casos como Falsos Negativos (FN), correspondiente a correos phishing que fueron catalogados incorrectamente como seguros.

Tabla 27

Resultados de la matriz de confusión – Escenario 1

	Predicción	
	Safe	Phishing
Safe	VN: 1314	FP: 45
Phishing	FN: 21	VP: 1202

- Precisión (Precision)

$$P = \frac{VP}{VP + FP} * 100$$

$$P = \frac{1202}{1202 + 45} * 100 = 96{,}39\%$$

- Recuperación (Recall):

$$R = \frac{VP}{VP + FN} * 100$$

$$R = \frac{1202}{1202 + 21} * 100 = 98{,}28\%$$

- Puntuación F1 (F1 score):

$$F1\ score = 2 * \frac{Precisión * Recuperación}{Precisión + Recuperación}$$

$$F1\ score = 2 * \frac{96{,}39 * 98{,}28}{96{,}39 + 98{,}28} = 97{,}33\%$$

- **Escenario 2** – **conjunto de muestras 1:**

El esquema de prueba para este escenario comprende el uso del siguiente dataset "Darito/spanish_spear_phishing", el cual sirve como una base de información para la

generación de los correos en formato eml por medio de un script de Python. Los archivos en formato eml cuyo número se sitúa en alrededor de las 200 muestras son importados al servicio de correo por medio de Thunderbird.

Los correos electrónicos importados son reubicados en directorios específicos en el buzón del usuario, vez configurado todo lo necesario se inicia el proceso de análisis de phishing.

La Tabla 28 detalla los resultados de la predicción de phishing basada en el modelo, los cuales fueron analizados conforme al tipo de correo e idioma.

Tabla 28

Resultados de análisis de detección de phishing – Escenario 2

Cantidad	**Tipo de Correo**	**Resultados del modelo**	
		Seguro	**Phishing**
50	Correo seguro (inglés)	29	21
50	Correo seguro (español)	46	4
50	Correo phishing (inglés)	7	43
50	Correo phishing (español)	15	35

La Tabla 29 muestra la sumatoria de los resultados en función de los 4 posibles casos a suscitarse dentro del análisis, dichos valores se calculan a partir de los datos expuestos en la Tabla 28.

Tabla 29

Resultados de la matriz de confusión – Escenario 2.

Predicción

	Safe	Phishing
Safe	VN: 75	FP: 25
Phishing	FN: 22	VP: 78

- Precisión (Precision)

$$P = \frac{VP}{VP + FP} * 100$$

$$P = \frac{78}{78 + 25} * 100 = 75{,}72\%$$

- Recuperación (Recall):

$$R = \frac{VP}{VP + FN} * 100$$

$$R = \frac{78}{78 + 22} * 100 = 78\%$$

- Puntuación F1 (F1 score):

$$F1\ score = 2 * \frac{Precisión * Recuperación}{Precisión + Recuperación}$$

$$F1\ score = 2 * \frac{75{,}72 * 78}{75{,}72 + 78} = 76{,}84\%$$

- **Escenario 3 – conjunto de muestras 2:**

En este caso, se replica el método de generación de correos por medio del dataset, con un ligero cambio en el número total de muestras empleadas. Se empleará un total de 2000 correos, distribuida equitativamente entre cada tipo de correo e idioma. Los detalles

del lote de correos empleado en la detección de phishing para este escenario se muestran en la Tabla 30.

Tabla 30

Resultados de análisis de detección de phishing - Escenario 3

Cantidad	Tipo de Correo	Resultados del modelo	
		Seguro	**Phishing**
500	Correo seguro (inglés)	452	48
500	Correo seguro (español)	466	34
500	Correo phishing (inglés)	53	447
500	Correo phishing (español)	109	391

La Tabla 31 muestra los resultados de detección para el escenario 3, dichos valores se presentan en función de los 4 posibles casos a suscitarse dentro del análisis.

Tabla 31

Resultados de la matriz de confusión – Escenario 3

	Predicción	
	Safe	Phishing
Safe	VN: 918	FP: 82
Phishing	FN: 162	VP: 838

- Precisión (Precision):

$$P = \frac{VP}{VP + FP} * 100$$

$$P = \frac{838}{838 + 82} * 100 = 91{,}08\%$$

- Recuperación (Recall):

$$R = \frac{VP}{VP + FN} * 100$$

$$R = \frac{838}{838 + 162} * 100 = 83{,}80\%$$

- Puntuación F1 (F1 Score):

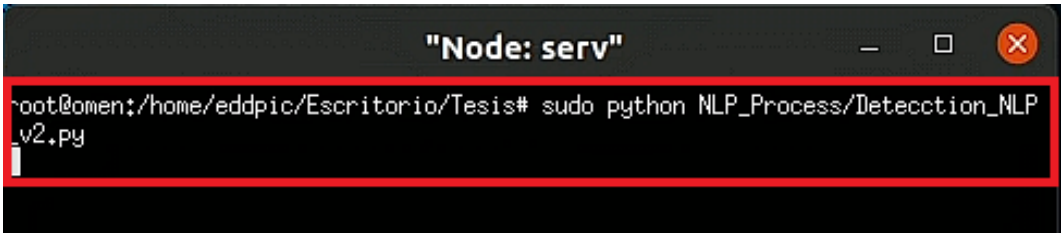
$$F1\ score = 2 * \frac{Precisión * Recuperación}{Precisión + Recuperación}$$

$$F1\ score = 2 * \frac{91{,}08 * 83{,}80}{91{,}08 + 83{,}80} = 87{,}28\%$$

La Figura 156 muestra la ejecución del script en el nodo “serv”, el cual realiza los procesos de conexión, adquisición y cálculo de la probabilidad de phishing, además que en base a los resultados genera acciones de eliminación de correos y la generación de reglas de Suricata.

Figura 156

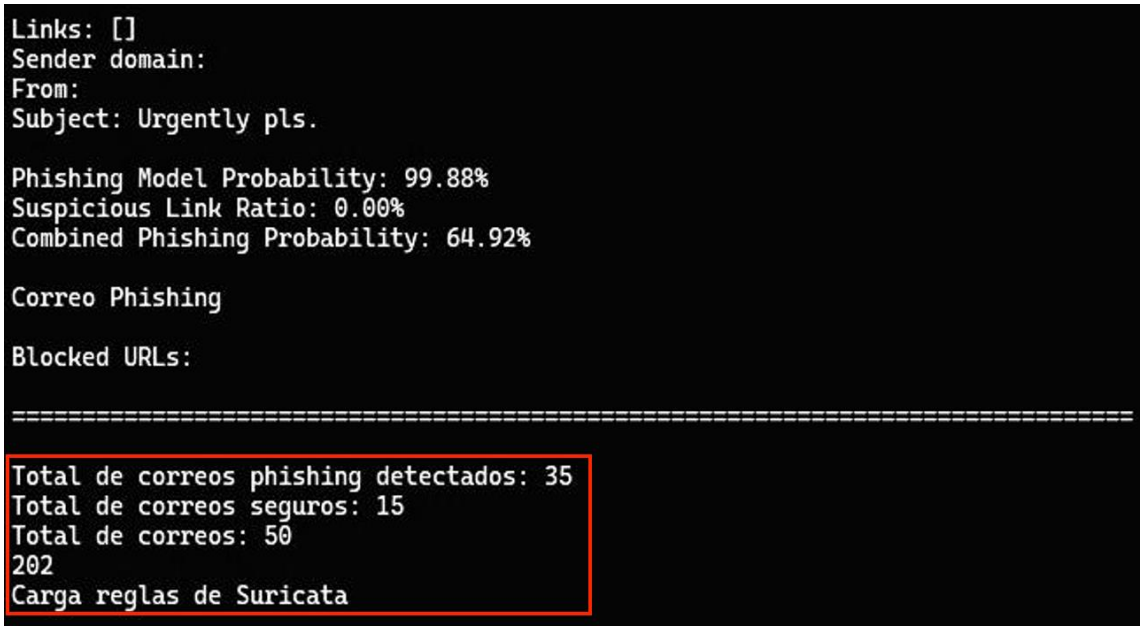
Inicialización del proceso de detección de phishing



Tras culminar el proceso de análisis, se presenta un resumen general en el que se detalla el número total de correos analizados, así como el número de casos detectados, tanto de correos clasificados como phishing como de correos seguros. Los resultados obtenidos pueden observarse en la Figura 157.

Figura 157

Detalles de análisis de phishing



Posteriormente se genera una notificación a manera de correo electrónico con un carácter de urgente en los que se detalla el número de casos de phishing suscitados durante el análisis.

La Figura 159 muestra la notificación de llegada del correo a Thunderbird, el cual se encuentra anexado a un nodo de la red IoT.

Figura 158

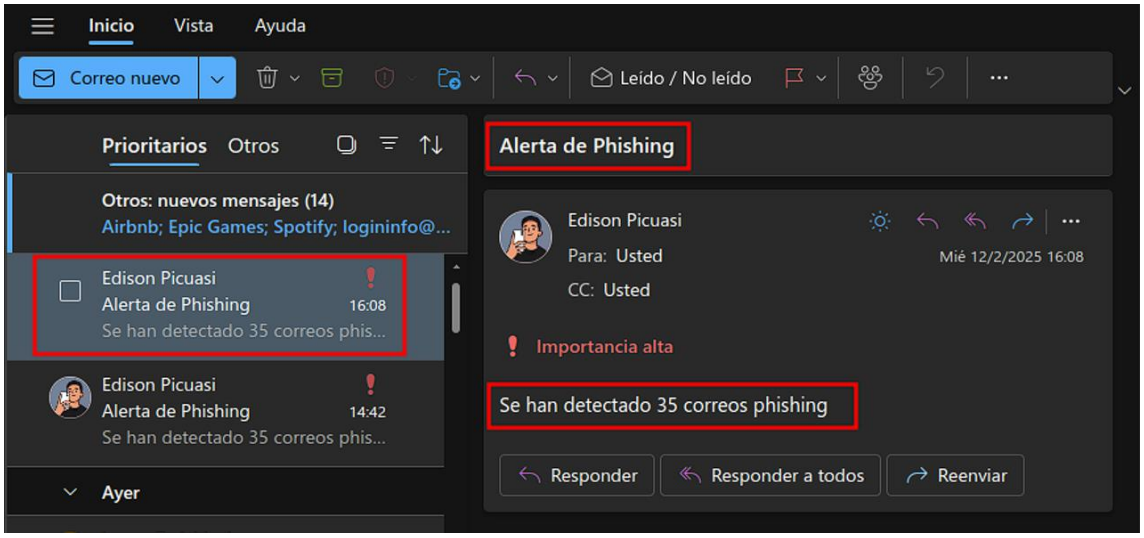
Notificación de correo Thunderbird

En las particularidades del correo electrónico, ilustrada en la Figura 159, se puede apreciar el mensaje que incluye el número de casos detectados, el asunto y el nivel de

prioridad asignado. Este mensaje sigue el patrón de correo estipulado en la función de "draft_message_body", la cual define la estructura y el formato de los correos de alerta.

Figura 159

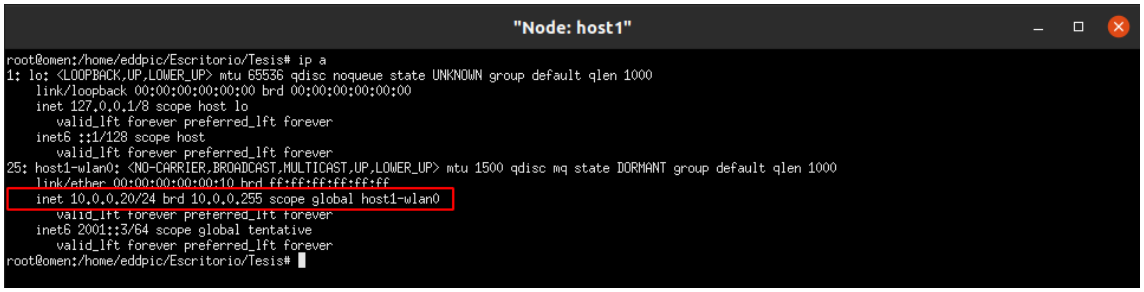
Notificación de resultados de análisis de phishing



La Figura 160 ilustra los detalles de red del nodo "host1", el cual servirá como dispositivo de prueba en el proceso de interacción con enlaces sospechosos de phishing.

Figura 160

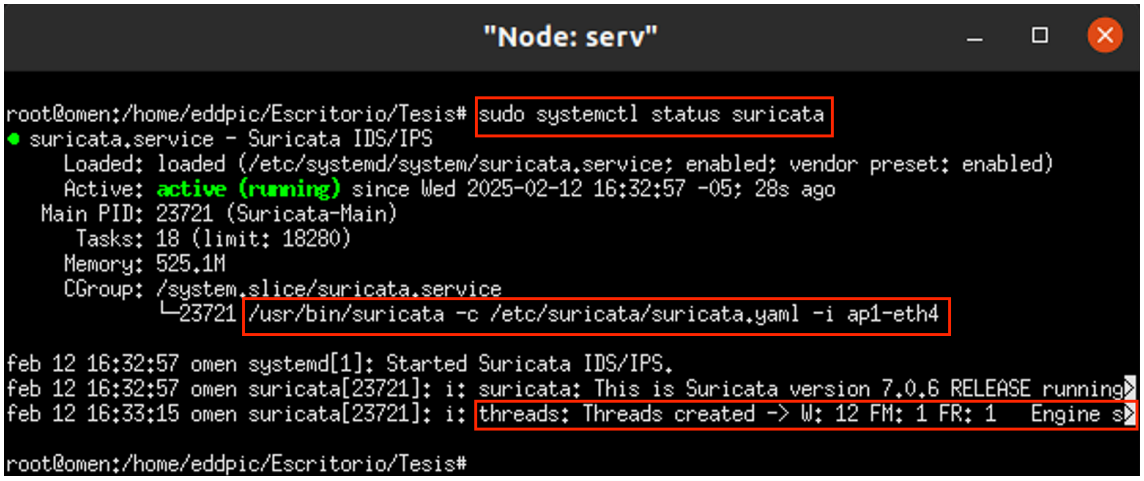
Detalles de red del nodo (Host1)



De igual forma, se corrobora el funcionamiento del IDS Suricata. Como se muestra en la Figura 161, se evidencia el correcto funcionamiento del servicio, así como la monitorización en la interfaz "ap1-eth4".

Figura 161

Detalles de funcionamiento de Suricata

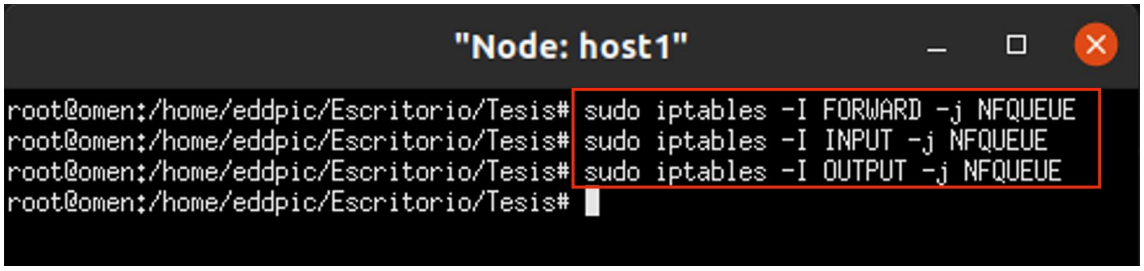


Para que Suricata actúe como un agente activo, es decir que sea capaz de generar acciones en respuesta al tráfico de red, se procede a redirigir el tráfico hacia Suricata desde el cortafuegos de Linux. En ese sentido, se establecen las reglas para gestionar el tráfico entrante, saliente y el del reenvío. Este tráfico es analizado, inspeccionado y sometido a un proceso de toma de decisiones (permitir, descartar o modificar paquetes) por parte Suricata, en un espacio de red denominado colas (NFQUEUE).

En la Figura 162 se presentan los comandos empleados para habilitar el modo IPS en Suricata, el cual opera en conjunto con el cortafuegos y la herramienta de iptables.

Figura 162

Habilitación de modo IPS mediante iptables

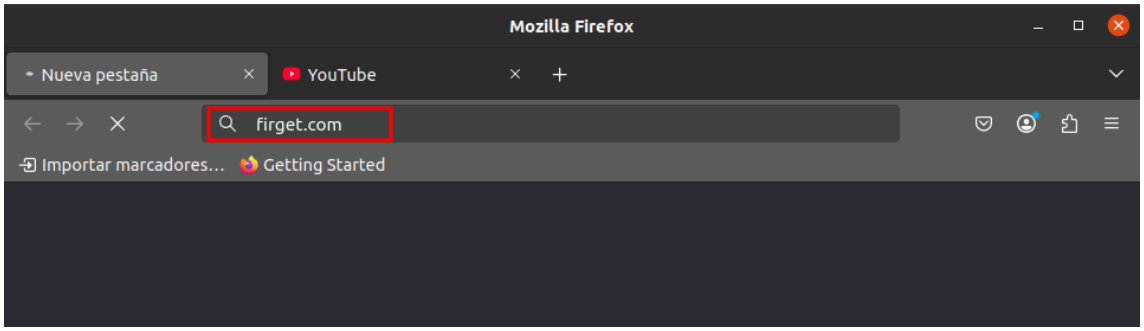


Se evalúa el funcionamiento del modo IPS de Suricata por medio de uno de los nodos de la topología, se lanza el recurso del navegador desde el nodo “host1” para constatar el bloqueo de acceso al enlace malicioso.

La Figura 163 muestra el proceso de petición de acceso y la nula respuesta del sitio web a través del navegador web.

Figura 163

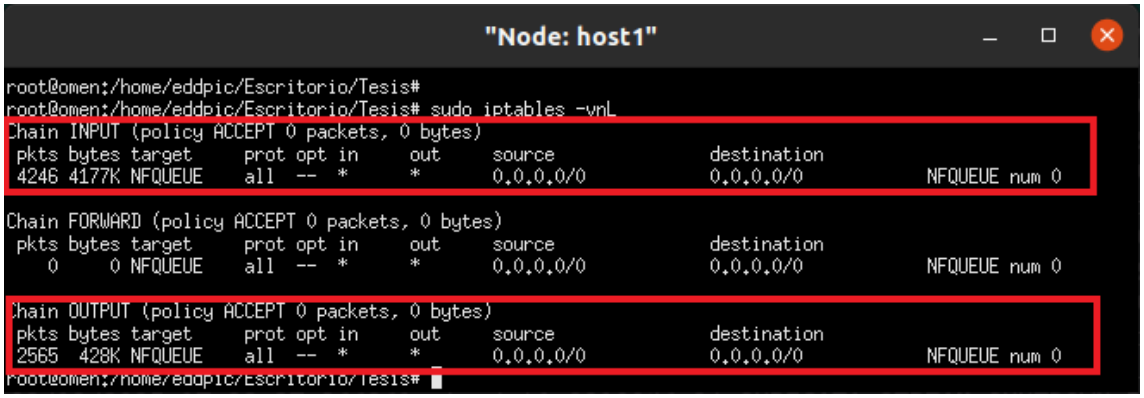
Acceso al enlace malicioso



En el apartado de las reglas de iptables que se encuentran en funcionamiento se evidencia los procesos de redirección del tráfico entrante y saliente hacia Suricata. En la Figura 164 se resalta la redirección del tráfico hacia la cola NFQUEUE y el número de paquetes redireccionados hasta ese momento.

Figura 164

Detalles de monitoreo iptables

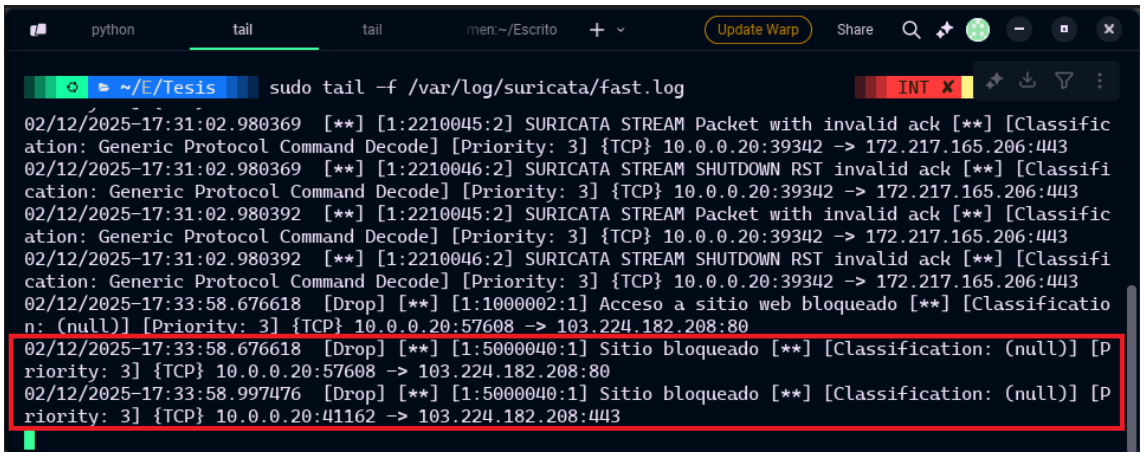


En lo que respecta a los registros de actividad de Suricta se evidencia los procesos de bloqueo al sitio web catalogado como phishing por medio del sistema implementado. En los detalles del registro se observa la información del formato de la regla aplicada y el dispositivo de origen.

La Figura 165 muestra los registros de acceso generados por las reglas de bloqueo presentes en Suricata, que a su vez fueron generadas por el sistema de detección de phishing.

Figura 165

Detalles de registro generadas por Suricata IDS



4.5 Factibilidad

En el siguiente apartado se llevará a cabo un análisis de la viabilidad para la implementación del sistema de detección de phishing. Para ello, se abordarán los aspectos más relevantes que deben considerarse, tales como: el rendimiento del modelo, la facilidad de implementación, los costos y los beneficios.

4.5.1 Rendimiento del modelo

Un análisis de los escenarios muestra como las métricas de evaluación se redujeron cuando el modelo opera dentro del sistema de detección de phishing. Pese a la

disminución de dichas métricas se puede mencionar que el porcentaje mínimo de exactitud del modelo correspondiente al 75,72%, puede catalogarse como aceptable, pero evidencia la necesidad de mejorar la robustez del modelo en entornos operativos. Los detalles de exactitud y el resto de las métricas resultantes para cada escenario se exponen en la Tabla 32.

Tabla 32

Resumen de métricas de evaluación para cada escenario

Escenario	**Métricas de Evaluación**		
	Precisión	Recuperación	Puntuación F1
Escenario 1	96,84%	97,71%	97,27%
Escenario 2	75,72%	78%	76,84%
Escenario 3	91,08%	83,80%	87,28%

4.5.2 Facilidad de implementación

La implementación del sistema de detección de phishing, que utiliza un modelo preentrenado junto con el IDS, puede seguir dos enfoques principales: uno centrado en una infraestructura local y otro basado en una arquitectura híbrida. En ambos casos, la implementación del IDS es similar, ya que este debe instalarse en la red local debido a la necesidad de realizar un análisis constante del tráfico de la red. Sin embargo, en lo que respecta al sistema de detección de phishing, se puede implementar tanto en la red local como en una instancia en la nube. Esto se debe a que los procesos de detección y generación de contramedidas son relativamente independientes entre sí, y solo dependen de la actualización continua de las reglas de Suricata. Así, la elección de implementar el sistema en la red local o en la nube depende de factores como la escalabilidad, la disponibilidad de recursos y las necesidades específicas de la infraestructura.

4.5.3 Beneficios

A continuación, se detallan los beneficios contemplados de la implementación del sistema de detección de phishing en base al procesamiento de lenguaje natural (NLP).

- El sistema puede fortalecer la seguridad en la red, especialmente en entornos críticos como las redes domésticas inteligentes (Smart Home). Al integrar un modelo de detección de phishing con un IDS, se garantiza una vigilancia constante del tráfico de la red, permitiendo detectar y bloquear correos maliciosos antes de que lleguen a los usuarios. Esto reduce significativamente el riesgo de ataques de phishing que podrían comprometer la privacidad y seguridad de los usuarios y dispositivos conectados.
- El sistema no solo detecta amenazas, sino que también genera contramedidas en tiempo real, como reglas de bloqueo para Suricata. Esto permite una respuesta automática y proactiva frente a las amenazas, evitando que los ataques se materialicen, lo que a su vez reduce el impacto en la red y los dispositivos conectados.
- La automatización del proceso de detección y respuesta ante phishing, incluyendo la generación y actualización de las reglas de Suricata, reduce significativamente la carga administrativa. Esto minimiza la necesidad de intervención manual, optimizando los recursos humanos.
- El sistema proporciona una capa adicional de protección para datos sensibles, especialmente en redes domésticas inteligentes conectadas a dispositivos como cámaras, termostatos y sistemas de seguridad. Al impedir la entrada de correos de phishing destinados a robar credenciales o acceder a sistemas de control remoto, el sistema ayuda a preservar la privacidad y seguridad tanto de los usuarios como de sus dispositivos.

Conclusiones y recomendaciones

Conclusiones

En este trabajo se ha desarrollado un sistema funcional para la detección de phishing en entornos IoT, mediante técnicas de Procesamiento de Lenguaje Natural (NLP) y un Sistema de Detección de Intrusos (IDS), con un índice de precisión para la clasificación aceptable de 76,84%, además de que se complementa de métodos que previenen el acceso o interacción con los elementos maliciosos de un correo phishing.

Se determinó los requisitos de software y servicios necesarios para el diseño de la red IoT y el modelo. En cuanto al apartado más relevante correspondiente al modelo se empleó el lenguaje de programación Python debido a que el software para la simulación de redes soporta dicho lenguaje además de que este es ampliamente usado en trabajos orientados son la ciencia de datos.

Es importante mantener actualizado el modelo de detección de phishing, ya que en los últimos sondeos de la situación actual en torno a dichos ataques este sea posicionado entre los principales, en vista de eso sería necesario adaptar el modelo para que este pueda manejar más variables de análisis como urls acortadas, fuentes de imágenes, archivos adjuntos, etc. De manera que su uso e implementación sea sencilla y más amigable con el usuario.

Si bien el sistema se ejecuta en un ambiente totalmente simulado surge la necesidad de evaluar el rendimiento del modelo en condiciones operativas con tráfico en vivo, con número mayor de nodos, así como la adaptabilidad con otra clase de recursos y servicios, ya que como se mencionó anteriormente se están gestando técnicas más

sofisticadas de phishing que están siendo influenciados por el acceso a la inteligencia artificial generativa.

Recomendaciones

- Para trabajos futuros, se recomienda la aplicación de métodos de compresión de modelos con el objetivo de reducir el tamaño del modelo y extender la integración en dispositivos con recursos limitado.

- Se recomienda explorar mecanismos alterativos para acceder a las cuentas de correo electrónico, debido a que los protocolos tradicionales como IMAP o POP3 presentan limitaciones de seguridad y en algunos casos están restringidos o están siendo reemplazados por nuevos mecanismos de acceso.

- Se sugiere implementar un sistema automatizado para el reinicio del servicio de Suricata de manera que garantice la adición de las nuevas reglas. Esto es especialmente importante en sesiones de análisis prolongados, donde las reglas actualizadas podrían no estar presentas por cuestiones asociadas con el reinicio del servicio.

- Extender la compatibilidad para anexar diferentes servicios de correo electrónico, implementando medidas adecuadas de seguridad y métodos de encriptación que eviten el almacenamiento local de datos sensibles como credenciales de usuario. Además de explorar la integración del sistema con servicios de seguridad existentes, como firewalls, así como evaluar su posible incorporación como una funcionalidad de los clientes de correo.

Referencias Bibliográficas

Abbas, S. G., Vaccari, I., Hussain, F., Zahid, S., Fayyaz, U. U., Shah, G. A., Bakhshi, T., & Cambiaso, E. (2021). Identifying and mitigating phishing attack threats in IoT use cases using a threat modelling approach. *Sensors*, *21*(14), 1–25. https://doi.org/10.3390/s21144816

Achary, R. (2021). *Cryptography and Network Security: An Introduction*. Mercury Learning and Information. https://books.google.com.ec/books?id=yCM6EAAAQBAJ

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., & Watteyne, T. (2015). FIT IoT-LAB: A large scale open experimental IoT testbed. *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 459–464. https://doi.org/10.1109/WF-IoT.2015.7389098

Aghaei, E., Niu, X., Shadid, W., & Al-Shaer, E. (2022). Securebert: A domain-specific language model for cybersecurity. *International Conference on Security and Privacy in Communication Systems*, 39–56.

Aguilar Martínez, M. R., Paredes López, J. A., Gordillo Cevallos, D. P., & León Burgos, G. P. (2023). La protección de datos personales en Ecuador. *Estudios Del Desarrollo Social: Cuba y América Latina*, *10*(numero especial 1 SE-Artículos). https://revistas.uh.cu/revflacso/article/view/3594

Alabdan, R. (2020). Phishing Attacks Survey: Types, Vectors, and Technical Approaches. In *Future Internet* (Vol. 12, Issue 10). https://doi.org/10.3390/fi12100168

Alanezi, M. (2021). *Phishing detection methods: a review*.

Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing Attacks: A Recent Comprehensive Study and a New Anatomy . In *Frontiers in Computer Science* (Vol. 3). https://www.frontiersin.org/articles/10.3389/fcomp.2021.563060

Alshattnawi, S., Shatnawi, A., AlSobeh, A. M. R., & Magableh, A. A. (2024). Beyond Word-Based Model Embeddings: Contextualized Representations for Enhanced Social Media Spam Detection. *Applied Sciences*, *14*(6), 2254.

Ameri, K., Hempel, M., Sharif, H., Lopez Jr, J., & Perumalla, K. (2021). Cybert: Cybersecurity claim classification by fine-tuning the bert language model. *Journal of Cybersecurity and Privacy*, *1*(4), 615–637.

Andrea, I., Chrysostomou, C., & Hadjichristofi, G. (2015). Internet of Things: Security vulnerabilities and challenges. *2015 IEEE Symposium on Computers and Communication (ISCC)*, 180–187. https://doi.org/10.1109/ISCC.2015.7405513

Benavides, E., Fuertes, W., Sanchez, S., & Nuñez-Agurto, D. (2020). Caracterización de los ataques de phishing y técnicas para mitigarlos. Ataques: una revisión sistemática de la literatura. *Ciencia y Tecnología*, *13*(1 SE-Ciencias agrotecnológicas), 97–104. https://doi.org/10.18779/cyt.v13i1.357

Chaudhari, B. S., & Zennaro, M. (2020). *LPWAN Technologies for IoT and M2M Applications*. Elsevier Science. https://books.google.com.ec/books?id=68i2DwAAQBAJ

Chung, K., Thaichon, P., & Quach, S. (2022). Types of artificial intelligence (AI) in marketing management. In *Artificial intelligence for marketing management* (pp. 29–40). Routledge.

Cuzme Rodríguez, F. G. (2015). *El internet de las cosas y las consideraciones de seguridad*. Quito/PUCE/2015.

Dahlqvist, F., Patel, M., Rajko, A., & Shulman, J. (2019). Growing opportunities in the Internet of Things. *McKinsey & Company*, 1–6.

Egress. (2024). *Email Security Risk Report 2024*. https://www.egress.com/media/o1sbpq5t/egress_email_security_risk_report_2024.pdf

Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., & Marrón, P. J. (2009). COOJA/MSPSim: interoperability testing for wireless sensor networks. *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 1–7.

Ferrag, M. A., Ndhlovu, M., Tihanyi, N., Cordeiro, L. C., Debbah, M., Lestable, T., & Thandi, N. S. (2024). Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices. *IEEE Access*.

Gartner. (2024). *Emerging Risks Report 2Q24*. https://emt.gartnerweb.com/ngw/globalassets/en/risk-audit/documents/emerging_risks.pdf

Gil, C. M. R. (2023). *Programación de Inteligencia Artificial. Curso Práctico*. Ra-Ma S.A. Editorial y Publicaciones. https://books.google.com.ec/books?id=1jnCEAAAQBAJ

Gualberto, É. S. (2021). *Detecção de phishing: métodos baseados em processamento de linguagem natural*.

Gupta, B. B., Arachchilage, N. A. G., & Psannis, K. E. (2018). Defending against phishing attacks: taxonomy of methods, current issues and future directions. *Telecommunication Systems*, *67*(2), 247–267. https://doi.org/10.1007/s11235-017-0334-z

Islak, S., CRUMPTON, M., ÖZACAR, K., GÜRER, C., ARSOY, S., BABAYİĞİT, B., SATTUF, H., KAYA, V., AKGÜL, İ., & AYDIN, Y. (2022). *Interdisciplinary Engineering Sciences Concepts, Researches and Applications*. Livre de Lyon. https://books.google.com.ec/books?id=LpqnEAAAQBAJ

ISO. (2018). INTERNATIONAL STANDARD ISO / IEC Information technology — Security techniques — Information security management systems — Overview and. *ACM Workshop on Formal Methods in Security Engineering.Washington, DC, USA*, *34*(19), 45–55. https://standards.iso.org/ittf/PubliclyAvailableStandards/c073906_ISO_IEC_27000_2018_E.zip

ISO/IEC/IEEE 29148. (2018). ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, 1–104. https://doi.org/10.1109/IEEESTD.2018.8559686

Jakhar, D., & Kaur, I. (2020). Artificial intelligence, machine learning and deep learning: definitions and differences. *Clinical and Experimental Dermatology*, *45*(1), 131–132.

Kassab, W., & Darabkh, K. A. (2020). A–Z survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications*, *163*, 102663.

Kaynak, O. (2021). The golden age of Artificial Intelligence. *Discover Artificial Intelligence*, *1*(1), 1. https://doi.org/10.1007/s44163-021-00009-x

Kim, J. H. (2017). A Survey of IoT Security: Risks, Requirements, Trends, and Key Technologies. *Journal of Industrial Integration and Management*, *02*(02), 1750008. https://doi.org/10.1142/S2424862217500087

Koide, T., Fukushi, N., Nakano, H., & Chiba, D. (2023). Detecting phishing sites using chatgpt. *ArXiv Preprint ArXiv:2306.05816*.

Lantz, B., Handigol, N., Heller, B., & Jeyakumar, V. (2017). Introduction to mininet. *Mininet Project,[En Línea]*.

Ley Organica de Proteccion de Datos Personales, Gaceta Oficial 1 (2021).

Mehdi, K., Lounis, M., Bounceur, A., & Kechadi, T. (2014). Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool. *Seventh International Conference on Simulation Tools and Techniques*.

Mendsaikhan, O., Hasegawa, H., Yamaguchi, Y., & Shimada, H. (2019). Identification of cybersecurity specific content using the Doc2Vec language model. *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, *1*, 396–401.

Miller, L. C., & Gregory, P. H. (2022). *CISSP For Dummies*. Wiley. https://books.google.com.ec/books?id=U8Z6EAAAQBAJ

Nardini, G., Sabella, D., Stea, G., Thakkar, P., & Virdis, A. (2020). Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks. *IEEE Access*, *8*, 181176–181191. https://doi.org/10.1109/ACCESS.2020.3028550

Oestriecher, K., & Beasley, M. (2020). *Annual Accounting and Auditing Workshop*. Wiley. https://books.google.com.ec/books?id=MhDxDwAAQBAJ

Patel, Z., & Gupta, S. (2018). *Future Internet Technologies and Trends: First International Conference, ICFITT 2017, Surat, India, August 31 - September 2, 2017, Proceedings*. Springer International Publishing. https://books.google.com.ec/books?id=JexHDwAAQBAJ

Pathan, A. S. K. (2014). *The State of the Art in Intrusion Prevention and Detection*. Taylor \& Francis. https://books.google.com.ec/books?id=o39cAgAAQBAJ

Proofpoint. (2024). *2024 State of the Phish*. https://www.proofpoint.com/sites/default/files/threat-reports/pfpt-us-tr-state-of-the-phish-2024.pdf

Rahali, A., & Akhloufi, M. A. (2023). MalBERTv2: Code aware BERT-based model for malware identification. *Big Data and Cognitive Computing*, *7*(2), 60.

Rastenis, J., Ramanauskaitė, S., Suzdalev, I., Tunaitytė, K., Janulevičius, J., & Čenys, A. (2021). Multi-language spam/phishing classification by email body text: Toward automated security incident investigation. *Electronics*, *10*(6), 668.

Sakaoglu, S. (2023). *KARTAL: Web Application Vulnerability Hunting Using Large Language Models: Novel method for detecting logical vulnerabilities in web applications with finetuned Large Language Models*.

Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. K. A. A. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Computer Science*, *171*, 1251–1260. https://doi.org/https://doi.org/10.1016/j.procs.2020.04.133

Senarath, U. S. (2021). Waterfall methodology, prototyping and agile development. *Tech. Rep.*, 1–16.

SlashNext. (2023). *The State of Phishing 2023*. https://slashnext.com/wp-content/uploads/2023/10/SlashNext-The-State-of-Phishing-Report-2023.pdf

Songailaitė, M., Kankevičiūtė, E., Zhyhun, B., & Mandravickaitė, J. (2023). BERT-Based Models for Phishing Detection. *CEUR Workshop Proceedings: IVUS 2023: Proceedings of the 28th International Conference on Information Society and University Studies, Kaunas, Lithuania, May 12, 2023.*, *3575*, 34–44.

Srivastava, T. V., & Purcell, J. (2007). Phishing and pharming–the deadly duo. *Sans Institute*.

Thakkar, A., & Lohiya, R. (2021). A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges. *Archives of Computational Methods in Engineering*, *28*(4), 3211–3243. https://doi.org/10.1007/s11831-020-09496-0

Thakkar, A., & Lohiya, R. (2022). A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review*, *55*(1), 453–563. https://doi.org/10.1007/s10462-021-10037-9

Tidjon, L. N., Frappier, M., & Mammar, A. (2019). Intrusion detection systems: A cross-domain overview. *IEEE Communications Surveys & Tutorials*, *21*(4), 3639–3681.

Torres Ruiz, D. A. (2019). *Sistemas de detección de intrusos basados en redes neuronales artificiales en ambientes con adversarios*. Uniandes. http://hdl.handle.net/1992/43759

Tseng, L., Yao, X., Otoum, S., Aloqaily, M., & Jararweh, Y. (2020). Blockchain-based database in an IoT environment: challenges, opportunities, and analysis. *Cluster Computing*, *23*(3), 2151–2165. https://doi.org/10.1007/s10586-020-03138-7

Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., & Fischer, M. (2015). Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Comput. Surv.*, *47*(4). https://doi.org/10.1145/2716260

Wu, H., Han, H., Wang, X., & Sun, S. (2020). Research on Artificial Intelligence Enhancing Internet of Things Security: A Survey. *IEEE Access*, *8*, 153826–153848. https://doi.org/10.1109/ACCESS.2020.3018170

Yang, L., & Shami, A. (2022). IDS-ML: An open source code for Intrusion Detection System development using Machine Learning. *Software Impacts*, *14*, 100446. https://doi.org/https://doi.org/10.1016/j.simpa.2022.100446

Zhang, C., & Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. *Journal of Industrial Information Integration*, *23*, 100224. https://doi.org/https://doi.org/10.1016/j.jii.2021.100224

Zivkovic, C., Guan, Y., & Grimm, C. (2020). *IoT Platforms, Use Cases, Privacy, and Business Models: With Hands-on Examples Based on the VICINITY Platform*. Springer International Publishing. https://books.google.com.ec/books?id=5JTyDwAAQBAJ

ANEXOS

Anexo 1: Tabla comparativa del software de simulación

Objetivo: Determinar las características principales de las diferentes opciones de software de simulación

Software	Capacidad de Simulación de Redes IoT	Protocolos Compatibles	Rendimiento	Sistemas Operativos Compatibles	Configuración de Parámetros	Fuente Bibliográfica
Cooja	ContikiOS pone a disposición el simulador de redes inalámbricas Cooja, el cual permite simular entornos de dispositivos interconectados, además de aspectos tales como memoria de datos, el tipo de nodo y periféricos de hardware.	Cuenta con soporte de IPV4, IPV6 y los protocolos de la familia 6LoWPAN.	Los parámetros de procesamiento, consumo energético se acrecientan a medida que aumenta el número de dispositivos, es decir la complejidad de la red.	Linux	Es extensible y configurable mediante la incorporación de extensiones que permiten modificar características como el protocolo, aspectos físicos de la comunicación inalámbrica.	Luiz, E. de O. B. M. (n.d.). *Sistema de detecção de intrusão baseado em criptografia simétrica para redes IoT de baixa potência*. Gaur, R., & Prakash, S. (2021). Performance and Parametric Analysis of IoT's Motes with Different Network Topologies. In S. Mekhilef, M. Favorskaya, R. K. Pandey, & R. N. Shaw (Eds.), *Innovations in Electrical and Electronic Engineering* (pp. 787–805). Springer Singapore.

QualNet	Es capaz de modelar redes cableadas, inalámbricas, submarinas, celulares y satelitales.	Admite gran cantidad de protocolos de la capa de protocolos, así como un amplio catálogo de librerías de tecnologías como WiFi, WiMax, GSM, UMTS, LTE, 5G, Link-11, Link-16.	El software precisa de un mayor porcentaje memoria RAM a medida que la topología supera una red de 1000 nodos.	Linux, Windows	El software es flexible y extensible, brinda la opción de crear nuevos protocolos y aplicaciones a partir de la incorporación de modelos en los escenarios de simulación.	Technologies, K. (n.d.). *QualNet Network Simulator Make Networks Work Network modeling software for: Development Analysis*. www.keysight.com
CupCarbon	Permite simular redes inalámbricas de sensores compatibles con escenarios orientados a ciudades inteligentes e Internet de las Cosas.	El software integra varios protocolos LoRaWAN, WiFi, ZigBee, MQTT	El número de nodos que puede incluir la topología depende de manera directa de los recursos de hardware del host.	MacOS, Windows, Linux	El software tiene la capacidad de modificar aspectos relevantes de la red (topología, condiciones del entorno) como de los elementos (posición, consumo de energía, propagación de señal)	Lopez-Pavon, C., Sendra, S., & Valenzuela-Valdés, J. (2018). Evaluation of CupCarbon Network Simulator for Wireless Sensor Networks. *Network Protocols and Algorithms*, *10*. https://doi.org/10.5296/npa.v10i2.13201 Kaur, N., & Deep, G. (2021). IoT-Based brinjal crop monitoring system. *Smart Sensors for Industrial Internet of Things: Challenges, Solutions and Applications*, 231–247.

OMNeT++						
NS-3	Permite simular un amplio segmento de redes tanto cableadas como inalámbricas además de tecnologías y protocolos.	IPv4, IPv6, IEEE 802.11a, 802.11b, 802.11g, OLSR	La dimensión de la topología depende de la complejidad de los protocolos que se integra en la simulación y a su vez de las especificaciones del host, un hardware robusto puede manejar una topología de algunos cientos de nodos.	Linux, MacOS, Windows	El software cuenta con la capacidad para personalizar varios aspectos de la simulación de la manera que se mejor se adecue a lo requerido.	Zárate Ceballos Henry and Parra Amaris, J. E. and J. J. H. and R. R. D. A. and A. R. O. and O. T. J. E. (2021). Network Simulating Using ns-3. In Wireless Network Simulation: A Guide using Ad Hoc Networks and the ns-3 Simulator (pp. 65–95). Apress. https://doi.org/10.1007/978-1-4842-6849-0_4

Mininet-WiFi	Permite emular redes inalámbricas de gran fidelidad basadas en software	Cualquier protocolo de las capas de red y aplicación, IEEE 802.11, IEEE 802.3, OpenFlow.	La dimensionalidad de la topología simulada se ve limitada por los recursos del host (Memoria RAM, CPU), debido a que estos se distribuyen de manera equitativa entre los nodos. Entre otros factores que influencian el rendimiento se consideran el número de nodos y la herramienta empleada.	Linux, MacOS, Windows	Ofrece gran flexibilidad al momento de simular redes, entre los parámetros configurables se sitúan los elementos de la topología, aspectos de los enlaces, de la red, etc.	Chaurasia, A., Mishra, S. N., & Chinara, S. (2019). Performance evaluation of software-defined wireless networks in it-sdn and mininet-wifi. *2019 1st International Conference on Advances in Information Technology (ICAIT)*, 315–319. dos Reis Fontes, R., & Rothenberg, C. E. (2019). Mininet-wifi: Plataforma de emulação para redes sem fio definidas por software. *Anais Estendidos Do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 201–208.
FIT IoT-LAB	Ofrece una plataforma que pone a disposición nodos reales que se localizan en varias zonas geográficas a las que se accede de manera remota.	Es compatible con protocolos y tecnologías relacionadas con redes de sensores inalámbricos e internet de las cosas	La dimensión de la topología depende del tipo de red, además está sujeta a la disponibilidad de los nodos (motes) en la plataforma.	Linux, MacOS, Windows	La plataforma permite modificar una gran variedad de parámetros ya sea de los nodos, protocolos de comunicación, topología de red, seguridad e integración con	Eric Fleury, Nathalie Mitton, Thomas Noel, Cédric Adjih. FIT IoT-LAB: The Largest IoT Open Experimental Testbed. ERCIM News, 2015, 101, pp.4. ⟨hal-01138038⟩

(IoT) como IEEE 802.15.4, 6LoWPAN, Zigbee, Wi-Fi, TCP, UDP, etc.	herramientas externa y APIs.

Anexo 2: Revisión de trabajos relacionados

Objetivos:

- Extraer información relevante para el estudio
- Determinar las técnicas y herramientas que emplea el trabajo

Trabajo 1

Ficha Bibliográfica 1			
Autor	• Denish Omondi Otieno • Akbar Siami Namin • Keith S. Jones	**Editorial**	IEEE
Titulo	The Application of the BERT Transformer Model for Phishing Email Classification		
Año	2023	**Ciudad**	Lubbock

Resumen del contenido:

Metodología de Detección:

Clasifica los correos electrónicos en base a tendencias temáticas resultantes de procesos en torno al análisis de sentimientos, ingeniería de características y preprocesamiento de datos.

Técnicas y Herramientas:

Clasificación en base al modelo BERT, entrenamiento por medio de la plataforma de Google Colab.

Dataset Empleado:

Nazario phishing corpus: Es un conjunto de datos está conformado por 8094 correos electrónicos que fueron recolectados y registrados cada año.

Enron email dataset: Es un conjunto de datos que consta de alrededor de medio millón de correos electrónicos proveniente de 150 usuarios.

Categoría de Phishing:

Phishing mediante email.

Trabajo 2

Ficha Bibliográfica 2			
Autor	• Waafi Abdullah Bin Adam • Ivriah Yue Xuan Tan • Chung Shing Lai • Nurtiara Tania Rahim • Benjamin Yu Bin Tham • Huaqun Guo	**Editorial**	IEEE
Titulo	VishingDefender: An Advanced Vishing Defence System Against Vishing Attacks		
Año	2023	**Ciudad**	Singapur

Resumen del Contenido:

Metodología de Detección:

Detecta intentos de vishing mediante análisis de texto y coincidencia de patrones. Emplea la API de conversión de voz a texto de Google Cloud para transformar la llamada telefónica en texto, la cual posteriormente se preprocesa y analiza con el fin de detectar el ataque. La solución se implementa a manera de aplicación móvil para Android y un bot de Telegram.

Técnicas y Herramientas:

Algoritmos Predictivos:

- Coseno Similarity, Trie y KMP (Knuth-Morris-Pratt)

Bibliotecas de NLP:

- Gensim, SpaCy, NLTK (Natural Language Toolkit) y Scikit-Learn

Dataset Empleado:

Text-classification-of-voice-phishing-transcripts: Es un conjunto de datos creado por el usuario kimdesok, este contiene transcripciones en coreano de mensajes de voz y conversaciones telefónicas categorizadas en llamadas fraudulentas y no fraudulentas.

Categoría de Phishing:

Phishing Telefónico

Trabajo 3

Ficha Bibliográfica 3			
Autor	• Bianca Montes Jones • Marwan Omar	**Editorial**	IEEE

Titulo	Detection of Twitter Spam with Language Models: A Case Study on How to Use BERT to Protect Children from Spam on Twitter		
Año	2023	**Ciudad**	-

Resumen del Contenido:

Metodología de Detección:

Analiza y detecta spam en la red social X anteriormente conocida como Twitter mediante el modelo de lenguaje BERT. Aplica métodos de preprocesamiento de datos, extracción de características basada en TF-IDF y clasificación a través del algoritmo Naive Bayes.

Técnicas y Herramientas:

Modelo NLP

- BERT

Algoritmos:

- TF-IDF (Term Frequency-Inverse Document Frequency)
- Naïve Bayes

Dataset Empleado:

Dataset tweets: Conjunto de datos conformado por 5572 tweets categorizados como spam (747) y no spam (4825).

Categoría de Phishing:

Phishing en redes sociales.

Trabajo 4

Ficha Bibliográfica 4			
Autor	• Rebet Jones • Marwan Omar	**Editorial**	IEEE

	• Derek Mohammed • Calvin Nobels • Maurice Dawson		
Titulo	IoT Malware Detection with GPT Models		
Año	2023	**Ciudad**	Las Vegas

Resumen del Contenido:

Metodología de Detección:

Detecta malware utilizando modelos GPT. A través de la extracción de secuencias de código de máquina del tráfico de red, los cuales se tokenizan por medio de la biblioteca de Transformers y finalmente se vectorizan por medio de la capa de incrustación del modelo GPT.

Técnicas y Herramientas:

Modelo NLP:

- GPT (Generative Pre-trained Transformer)

Bibliotecas:

- Hugging Face Transformers

Dataset Empleado:

IoT-23 dataset: Es un conjunto de datos que contiene información del tráfico de red recopilados de 23 dispositivos IoT tales como cámaras, enrutadores y televisores inteligentes. El dataset incluye tráfico de red clasificado como benigno y malicioso.

Malvis dataset: Es un conjunto de datos que incluye datos de trafico de red proveniente de 10 tipos de malware, además de botnets, ransomware y troyanos.

Categoría de Phishing:

Phishing basado en malware

Trabajo 5

Ficha Bibliográfica 5			
Autor	• Ming-Yang Su • Kuan-Lin Su	**Editorial**	Sensors
Titulo	BERT-Based Approaches to Identifying Malicious URLs		
Año	2023	**Ciudad**	Taoyuan

Resumen del Contenido:

Metodología de Detección:

Detecta URLs maliciosas haciendo uso de BERT, mediante el mecanismo de autoatención del modelo se tokenizan tanto cadenas de URL como sus características. Finalmente, de dichos tokens se extrae el significado semántico para determinar si la URL es maliciosa.

Técnicas y Herramientas:

Modelo NLP:

- bert-base-cased

Dataset Empleado:

Malicius-urls-dataset: Engloba un conjunto de urls categorizados entre benigno (424000), desconfiguracion (95000), phishing (93000) y malware (32000).

Phishing-Dataset: Comprende alrededor de 88000 muestras dividido en 2 categorías, urls benignas (58000) y urls de phishing (30000)

ISCX 2016: Este conjunto de datos se compone de alrededor de 160000 entradas distribuido en 5 categorías, urls benignas (35000), urls de desconfiguración (96000), malware (11000), spam (11000) y casos de phishing (10000).

Categoría de Phishing:

Phishing basado en urls.

Anexo 3: Análisis de referencias para la selección del modelo

Objetivos:

- Extraer información relevante para el estudio
- Determinar las características y elementos que requiere el modelo

Ref.	Artículo	Descripción	Metodología NLP	Conjunto de datos	Medidas de Desempeño	Enfoque NIST
(Ameri et al., 2021)	CyBERT: Cybersecurity Claim Classification by Fine-Tuning the BERT Language Model	Es un clasificador de reclamos de característica de ciberseguridad basado en BERT.	BERT	Documentos y Sitios Web usando web scraping	• Exactitud – (0.954) • Puntuación F1 – (0.93) • Área bajo la curva (AUC) – (0.948) • Tiempo de entrenamiento – (32,970 seg.) • Tiempo de prueba – (97 seg.) • Parámetros de entrenamiento – (108647026)	Detección
(Aghaei et al., 2022)	SecureBERT: A Domain-Specific Language Model for Cybersecurity	Propone un modelo de lenguaje capaz de capturar connotaciones de texto referentes a ciberseguridad aplicables a tareas que dependen del esfuerzo y experiencia humana.	RoBERTa	Artículos, Libros, Encuestas, Blogs/Noticias, Wikipedia, Informes de seguridad, Videos	• Precisión – (85.08) • Recuperación – (88.10) • Puntuación F1 – (86.65)	Detección

(Ferrag et al., 2024)	Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices	Propone una arquitectura que emplea el modelo BERT para la detectar amenazas cibernéticas en redes IoT.	BERT y técnica de codificación Privacy-Preserving Fixed-Length Encoding (PPFLE).	Edge-IIoTset dataset	• Precisión – (0.87) • Recuperación – (0.84) • Puntuación F1 – (0.84) • Soporte – (441371)	Detección
(Rahali & Akhloufi, 2023)	MalBERTv2: Code Aware BERT-Based Model for Malware Identification	Identifica malware en el código fuente de las aplicaciones mediante BERT y técnicas avanzadas de NLP.	MalBERT y Analizador de características	AMD dataset, Drebin dataset, VirusShare dataset, Androzoo dataset	• Exactitud – (0.9813) • Puntuación F1 – (0.9838) • Coeficiente de Correlación de Matthews (MCC) – (0.9684) • Precision – (0.9882) • Recuperación – (0.9807) • Área bajo la curva (AUC) – (0.9791)	Detección
(Mendsaikhan et al., 2019)	Identification of Cybersecurity Specific Content Using the Doc2Vec Language Model	Propone un Sistema para extraer información acerca de amenazas cibernéticas mediante un filtro de dominio especifico basado en Doc2Vec	Filtro de lenguaje natural basado en Doc2Vec	Discusiones de Reddit, Discusiones de StackExchange, Comentarios de Hackernews, Canales RSS de noticias de seguridad, Archivos de noticias Slashdot, Base de datos nacional de vulnerabilidades.	• Precisión – (0.8745) • Recuperación – (0.4503) • Puntuación F1 – (0.5945) • Exactitud – (0.8217)	Detección

(Alshattnawi et al., 2024)	Beyond Word-Based Model Embeddings: Contextualized Representations for Enhanced Social Media Spam Detection	Realiza un análisis en relación con la detección de spam en plataformas de redes sociales.	BERT, ELMo, Word2Vec - GLoVe	Twitter dataset, YouTube dataset	• Exactitud – (ELMo: 90%, BERT: 83%) • Precisión – (ELMo: 91%, BERT: 87%) • Recuperación – (ELMo: 89%, BERT: 79%) • Puntuación F1 – (ELMo: 90%, BERT: 84%)	Detección
(Koide et al., 2023)	Detecting Phishing Sites Using ChatGPT	Propone un Sistema que emplea GPT-4V para analizar el contenido de sitios web y URLs con el fin de detectar sitios phishing	GPT-4, GPT-4V y web crawler	OpenPhish, PhishTank,	• Precisión – (98.7%) • Recuperación – (99.6%) • Exactitud – (99.2%) • Puntuación F1 – (99.2%)	Detección
(Sakaoglu, 2023)	KARTAL: Web Application Vulnerability Hunting Using Large Language Models	Propone un método para la detección de vulnerabilidades de aplicaciones web mediante un modelo de lenguaje de gran tamaño (LLM)	GPT-3.5 Turbo	OWASP, MITRE	• Exactitud – (87.19%) • Recuperación – (83.36%) • Precisión – (82.20%) • Coeficiente de Correlación de Matthews (MCC) – (0.70) • Puntuación F1 – (0.82)	Detección
(Songailaitė et al., 2023)	BERT-Based Models for Phishing Detection	Propone un método de detección de correos electrónicos de phishing por medio de la afinación de modelos basados en BERT y la metodología de transferencia de aprendizaje.	DistilBERT, TinyBERT, RoBERTa	Phishingdata-Analysis, Text_Phishing_Email_ML_Classificaction	• Exactitud – (0.985) • Precisión – (0.985) • Recuperación – (0.985) • Puntuación – (0.985)	Detección

Anexo 4: Planteamiento del Escenario de Internet de las Cosas

Escenario Smart Home

El escenario planteado comprende un entorno de casa inteligente en el que se emulan varios dispositivos que permiten controlar varios parámetros en diversas áreas de la residencia.

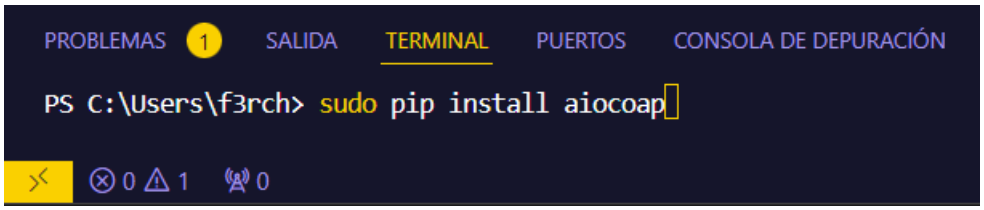
Sala: foco inteligente, termostato.
Cocina: foco inteligente.
Dormitorio: foco inteligente.
Entrada: enchufe inteligente.
Garaje: enchufe inteligente, sensor.

Los dispositivos emulados manejan uno o varios parámetros que se pueden controlar mediante una aplicación con el objetivo de proporcionar una interacción más real dentro del escenario, los datos que se originan en los dispositivos se envían a un servidor que ejecuta el protocolo CoAP y a su vez se reenvían hacia la base de datos de InfluxDB, de esta manera se realiza la representación de los datos en un dashboard mediante la conexión entre Grafana e InfluxDB. La siguiente tabla muestra la relación de los datos y el dispositivo.

Nodo/Dispositivo	Variables de registro
Bombilla Inteligente	Luminosidad, Estado, Color
Termostato	Temperatura, Estado
Foco Inteligente	Luminosidad, Estado, Color
Enchufe Inteligente	Estado
Sensor	Temperatura

Programación de nodos para el almacenamiento de datos servidor InfluxDB

El siguiente apartado precisa de la instalación del paquete de Python “aicoap”, el cual habilita la comunicación entre los dispositivos emulados con el servidor que actúa a manera de concentrador de datos haciendo uso de CoAP.



Se importa el paquete aiocoap junto con los submódulos que permiten definir aspectos relacionados al servidor y cliente, junto con los parámetros de manejo de las peticiones y respuestas.

```
import asyncio
from aiocoap import resource, Context, Message, CHANGED
import json
from InfluxDB import InfluxDBManager
```

El siguiente proceso establece los métodos de interacción entre los nodos a través del servidor de CoAP, permitiendo la recepción de los datos desde los dispositivos conectados y su posterior almacenamiento en la base de datos de InfluxDB. Esta metodología define la estructura para la recepción de los datos de cada uno de los dispositivos, y a su vez, gestiona la conexión con la base de datos de InfluxDB para efectuar el almacenamiento de la información. En este contexto, se establece los procesos para la administración de las solicitudes POST originadas por los dispositivos en base a identificativo “ID”, con el fin de registrar los valores monitorizados por cada uno de los dispositivos.

- **Enchufe Inteligente**

```
class PlugResource(resource.Resource):
    async def render_post(self, request):
        payload = json.loads(request.payload.decode('utf8'))
        plug_id = payload.get("plug_id")

        if plug_id not in server.plugs:
            server.plugs[plug_id] = {
                "is_on": False
            }

        print(f"Datos de Enchufe {plug_id} Recibidos: {payload}")

        if "turn_on" in payload:
            server.plugs[plug_id]["is_on"] = payload["turn_on"]
        if "turn_off" in payload:
            server.plugs[plug_id]["is_on"] = not payload["turn_off"]

        return Message(code=CHANGED, payload=f'Datos de Enchufe {plug_id} Procesados'.encode('utf-8'))
```

- **Foco Inteligente**

```
class BulbResource(resource.Resource):
    async def render_post(self, request):
        payload = json.loads(request.payload.decode('utf8'))
        bulb_id = payload.get("bulb_id")

        if bulb_id not in server.bulbs:
            server.bulbs[bulb_id] = {
                "is_on": False,
                "color": "white",
                "brightness": 100
            }

        print(f"Datos de Bombilla {bulb_id} Recibidos: {payload}")

        if "turn_on" in payload:
            server.bulbs[bulb_id]["is_on"] = payload["turn_on"]
        if "turn_off" in payload:
            server.bulbs[bulb_id]["is_on"] = not payload["turn_off"]
        if "color" in payload:
            server.bulbs[bulb_id]["color"] = payload["color"]
        if "brightness" in payload:
            server.bulbs[bulb_id]["brightness"] = payload["brightness"]

        response_payload = f'Datos de Bombilla {bulb_id} Recibidos y Procesados'.encode('utf-8')
        return Message(code=CHANGED, payload=response_payload)
```

- **Termostato**

```
class ThermostatResource(resource.Resource):
    async def render_post(self, request):
        payload = json.loads(request.payload.decode('utf8'))
        print(f"Datos de Termostato Recibidos: {payload}")

        if "turn_on" in payload:
            server.thermostat["is_on"] = payload["turn_on"]
        if "turn_off" in payload:
            server.thermostat["is_on"] = not payload["turn_off"]
        if "temperature" in payload:
            server.thermostat["temperature"] = payload["temperature"]

        return Message(code=CHANGED, payload=b'Datos de Termostato Recibidos y Procesados')
```

- **Sensor de Temperatura**

```
class TemperatureSensorResource(resource.Resource):
    async def render_post(self, request):
        payload = request.payload.decode('utf8')
        print(f"Datos de Sensor de Temperatura Recibidos: {payload}")
        server.temperature_sensor = float(payload)

        return Message(code=CHANGED, payload=b'Datos de Sensor de Temperatura Recibidos y
Procesados')
```

Código para la generación de datos en los nodos

El siguiente apartado tiene como objetivo la generación de datos, los cuales pueden ser generados de manera aleatoria o en base a las interacciones del usuario con las interfaces graficas de los dispositivos simulados.

El proceso general establece un método para el envío de información relacionado con el estado de las métricas o atributos evaluadas por los dispositivos. Esta información se estructura a manera de mensaje en formato JSON que contiene detalles específicos de cada dispositivo y los cuales se envían al servidor CoAP para su procesamiento.

- **Sensor de Temperatura**

```
import asyncio
from aiocoap import Context, Message, POST
import random

async def send_data():
    protocol = await Context.create_client_context()
    while True:
        temperature = round(random.uniform(15.0, 25.0), 2)  # Simulación de datos de temperatura
        payload = str(temperature).encode('utf8')
        request = Message(code=POST, uri='coap://10.0.0.100/temperature', payload=payload)
        response = await protocol.request(request).response
        print(f"Envío de datos temperatura: {temperature}")
        print(f"Respuesta: {response.code}\n{response.payload.decode('utf8')}")
        await asyncio.sleep(15)  # Enviar datos cada 10 segundos

if __name__ == "__main__":
    asyncio.run(send_data())
```

- **Foco Inteligente**

```
import asyncio
import aiocoap
import json
import customtkinter as ctk
from CTkColorPicker import *
import sys

SERVER_ADDRESS = "10.0.0.100"  # Dirección IP del servidor

class SmartLightBulb:
    def __init__(self, bulb_id):
        self.bulb_id = bulb_id
        self.color = 'white'
        self.brightness = 100
        self.is_on = False

    async def send_data(self, data):
        context = await aiocoap.Context.create_client_context()
        payload = json.dumps({"bulb_id": self.bulb_id, **data}).encode('utf-8')
        request = aiocoap.Message(code=aiocoap.POST, payload=payload)
        request.set_request_uri(f'coap://{SERVER_ADDRESS}/bulb')
        response = await context.request(request).response
        print(f"Respuesta del servidor: {response.code}, {response.payload.decode('utf-8')}")
```

- **Enchufe Inteligente**

```
import asyncio
import aiocoap
import json
import customtkinter as ctk
import sys

SERVER_ADDRESS = "10.0.0.100"  # Dirección IP del servidor

class SmartPlug:
    def __init__(self, plug_id):
        self.plug_id = plug_id
        self.is_on = False

    async def send_data(self, data):
        context = await aiocoap.Context.create_client_context()
        payload = json.dumps({"plug_id": self.plug_id, **data}).encode('utf-8')
        request = aiocoap.Message(code=aiocoap.POST, payload=payload)
        request.set_request_uri(f'coap://{SERVER_ADDRESS}/plug')
        response = await context.request(request).response
        print(f"Respuesta del servidor: {response.code}, {response.payload.decode('utf-8')}")
```

- **Termostato**

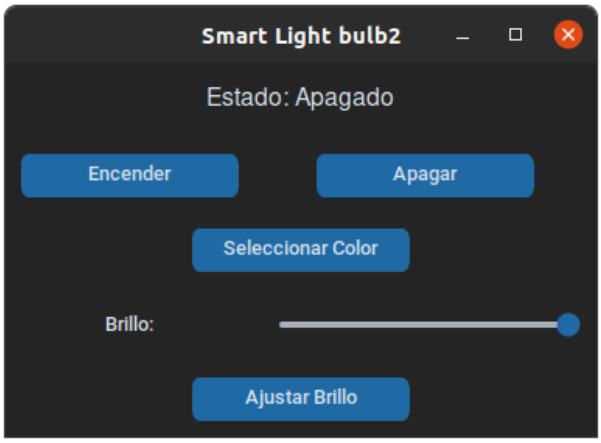
```
import asyncio
import aiocoap
import json
import customtkinter as ctk
from tkinter import messagebox

class SmartThermostat:
    def __init__(self):
        self.temperature = 22
        self.is_on = False

    async def send_data(self, data):
        context = await aiocoap.Context.create_client_context()
        request = aiocoap.Message(code=aiocoap.POST, payload=json.dumps(data).encode('utf-8'))
        request.set_request_uri('coap://10.0.0.100/thermostat')
        response = await context.request(request).response
        print(f"Respuesta del servidor: {response.code}, {response.payload.decode('utf-8')}")
```

Configuración de las interfaces de interacción con el usuario

En complemento al apartado de generación de datos, este proceso incluye la creación de las interfaces graficas que permite al usuario controlar los parámetros o valores monitoreados por los dispositivos. Estas interfaces se desarrollan por medio de la librería de customtkinter, la cual proporciona widgets especializadas, como botones, cuadros de dialogo y controles deslizantes entre otros. De esta manera se crea una interfaz intuitiva capaz de modificar los parámetros de cada dispositivo.

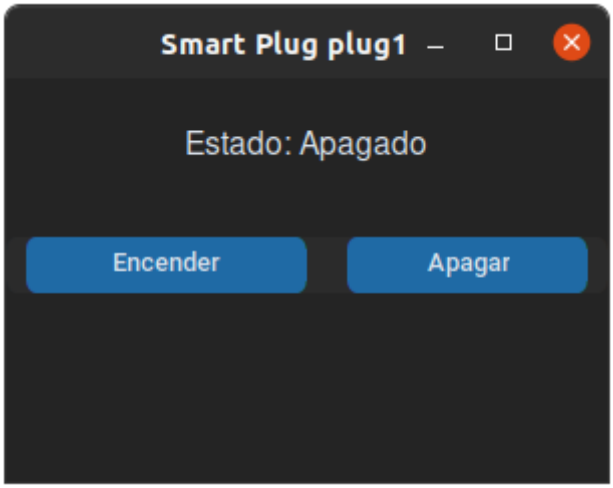
- Foco Inteligente

```python
class SmartBulbApp:
    def __init__(self, root, bulb_id):
        self.bulb = SmartLightBulb(bulb_id)
        self.root = root
        self.root.title(f"Smart Light {bulb_id}")
        self.setup_ui()

    def setup_ui(self):
        self.status_label = ctk.CTkLabel(self.root, text="Estado: Apagado", font=("Helvetica", 16))
        self.status_label.grid(row=0, column=0, columnspan=2, pady=10)

        ctk.CTkButton(self.root, text="Encender", command=lambda: self.toggle_bulb(True)).grid(row=1, column=0, padx=10, pady=10)
        ctk.CTkButton(self.root, text="Apagar", command=lambda: self.toggle_bulb(False)).grid(row=1, column=1, padx=10, pady=10)

        self.color_button = ctk.CTkButton(self.root, text="Seleccionar Color", command=self.choose_color)
        self.color_button.grid(row=2, column=0, columnspan=2, pady=10)

        ctk.CTkLabel(self.root, text="Brillo:").grid(row=3, column=0, padx=10, pady=10)
        self.brightness_slider = ctk.CTkSlider(self.root, from_=0, to=100)
        self.brightness_slider.set(100)
        self.brightness_slider.grid(row=3, column=1, padx=10, pady=10)

        ctk.CTkButton(self.root, text="Ajustar Brillo", command=self.set_brightness).grid(row=4, column=0, columnspan=2, pady=10)

    def toggle_bulb(self, state):
        self.bulb.is_on = state
        data = {"turn_on": state} if state else {"turn_off": True}
        asyncio.run(self.bulb.send_data(data))
        self.update_status()

    def update_status(self):
        state = "Encendido" if self.bulb.is_on else "Apagado"
        self.status_label.configure(text=f"Estado: {state}")

    def choose_color(self):
        color = AskColor().get()
        if color:
            self.bulb.color = color
            asyncio.run(self.bulb.send_data({"color": color}))
            self.color_button.configure(fg_color=color)

    def set_brightness(self):
        brightness = int(self.brightness_slider.get())
        self.bulb.brightness = brightness
        asyncio.run(self.bulb.send_data({"brightness": brightness}))
```



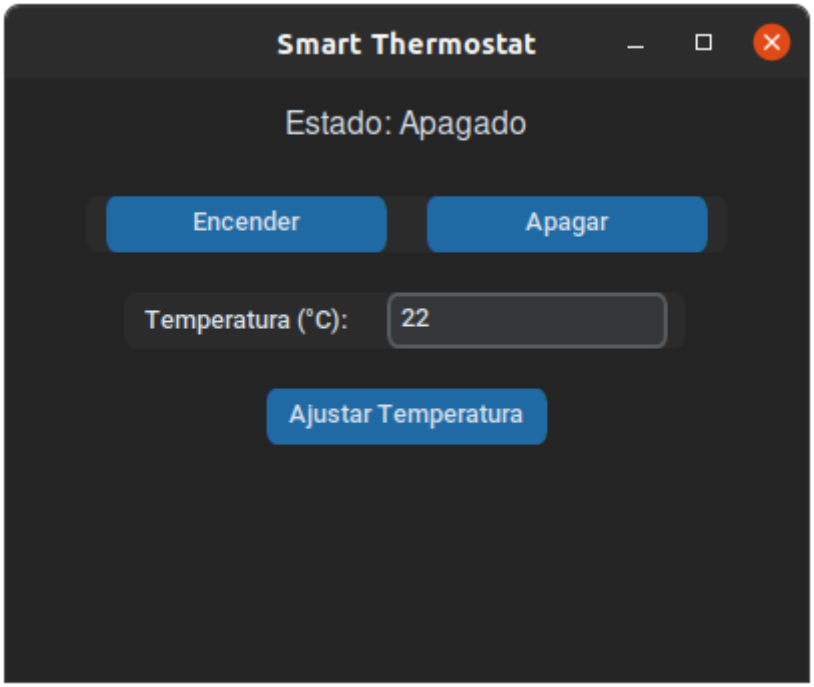
- Enchufe inteligente

```python
class PlugApp:
    def __init__(self, root, plug_id):
        self.plug = SmartPlug(plug_id)
        self.root = root
        self.root.title(f"Smart Plug {plug_id}")
        self.root.geometry("300x200")
        self.setup_ui()

    def setup_ui(self):
        ctk.set_appearance_mode("dark")
        ctk.set_default_color_theme("blue")

        self.status_label = ctk.CTkLabel(self.root, text="Estado: Apagado", font=("Helvetica", 16))
        self.status_label.pack(pady=20)

        button_frame = ctk.CTkFrame(self.root)
        button_frame.pack(pady=10)

        ctk.CTkButton(button_frame, text="Encender", command=self.turn_on).pack(side="left", padx=10)
        ctk.CTkButton(button_frame, text="Apagar", command=self.turn_off).pack(side="left", padx=10)

    def toggle_plug(self, state):
        self.plug.is_on = state
        data = {"turn_on": state} if state else {"turn_off": True}
        asyncio.run(self.plug.send_data(data))
        self.update_status()

    def update_status(self):
        state = "Encendido" if self.plug.is_on else "Apagado"
        self.status_label.configure(text=f"Estado: {state}")

    def turn_on(self):
        self.toggle_plug(True)

    def turn_off(self):
        self.toggle_plug(False)
```



- Termostato

```python
class ThermostatApp:
    def __init__(self, root):
        self.thermostat = SmartThermostat()
        self.root = root
        self.root.title("Smart Thermostat")
        self.root.geometry("400x300")
        self.setup_ui()

    def setup_ui(self):
        ctk.set_appearance_mode("dark")
        ctk.set_default_color_theme("blue")

        self.status_label = ctk.CTkLabel(self.root, text="Estado: Apagado", font=("Helvetica", 16))
        self.status_label.pack(pady=10)

        button_frame = ctk.CTkFrame(self.root)
        button_frame.pack(pady=10)

        ctk.CTkButton(button_frame, text="Encender", command=lambda: self.toggle_thermostat(True)).pack(side="left", padx=10)
        ctk.CTkButton(button_frame, text="Apagar", command=lambda: self.toggle_thermostat(False)).pack(side="left", padx=10)

        temp_frame = ctk.CTkFrame(self.root)
        temp_frame.pack(pady=10)

        ctk.CTkLabel(temp_frame, text="Temperatura (°C):").pack(side="left", padx=10)
        self.temp_entry = ctk.CTkEntry(temp_frame)
        self.temp_entry.insert(0, "22")
        self.temp_entry.pack(side="left", padx=10)

        ctk.CTkButton(self.root, text="Ajustar Temperatura", command=self.set_temperature).pack(pady=10)

    def toggle_thermostat(self, state):
        self.thermostat.is_on = state
        data = {"turn_on": state} if state else {"turn_off": True}
        asyncio.run(self.thermostat.send_data(data))
        self.update_status()

    def update_status(self):
        state = "Encendido" if self.thermostat.is_on else "Apagado"
        self.status_label.configure(text=f"Estado: {state}")

    def set_temperature(self):
        try:
            temperature = int(self.temp_entry.get())
            self.thermostat.temperature = temperature
            asyncio.run(self.thermostat.send_data({"temperature": temperature}))
        except ValueError:
            messagebox.showerror("Error", "Por favor, introduce un valor numérico válido para la temperatura.")
```



Configuración de InfluxDB

Se procede con la instalación de InfluxDB a manera de servicio, con el fin de que este disponible como un recurso para los nodos de la topología generada a partir de Mininet-WiFi.

Para esto se agrega las claves del software y a su vez se incluye el repositorio de InfluxDB a la lista de fuentes del sistema operativo.

```
~/Escritorio/Tesis
echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c influxdata-archive_compat.key'
| sha256sum -c && cat influxdata-archive_compat.key | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/
influxdata-archive_compat.gpg > /dev/null

influxdata-archive_compat.key: La suma coincide
```

```
~/Escritorio/Tesis
echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.co
m/debian stable main' | sudo tee /etc/apt/sources.list.d/influxdata.list

deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com/debi
an stable main
```

Una vez instalada la fuente del repositorio y actualizada la base de datos de los repositorios se procede a instalar el servicio de InfluxDB.

```
~/Escritorio/Tesis
sudo apt-get install influxdb2
```

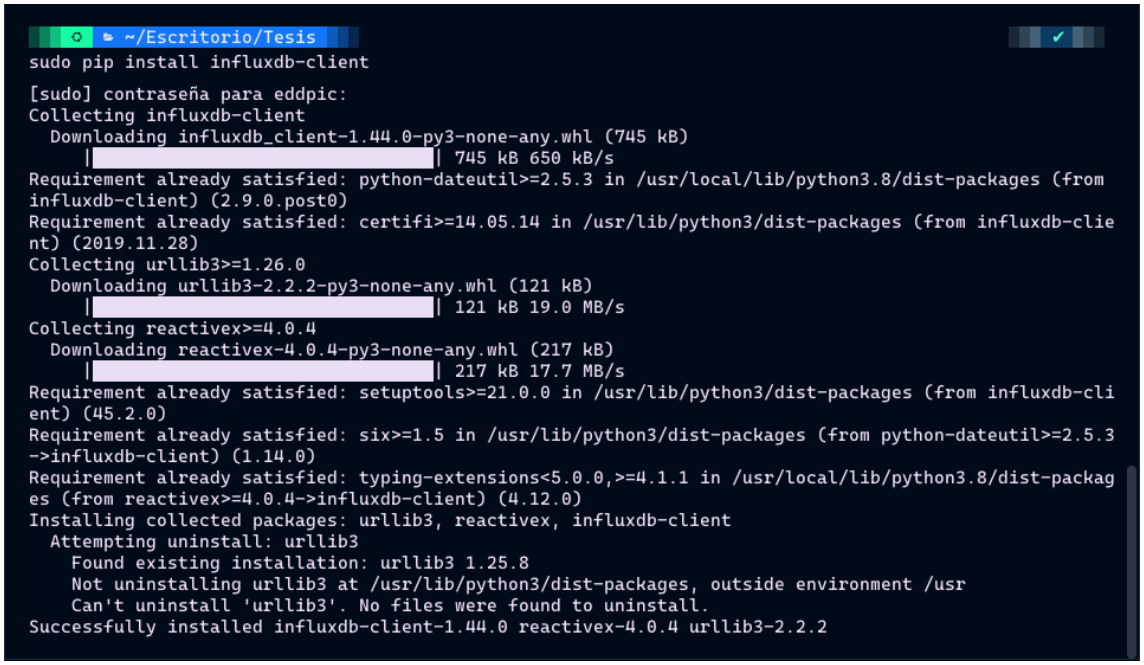
```
~/Escritorio/Tesis
influx version

Influx CLI dev (git: a79a2a1b825867421d320428538f76a4c90aa34c) build_date: 2024-04-16T14:34:32Z
```

Se inicia y habilita el servicio de InfluxDB de manera que entre en acción al momento en que se inicia el equipo anfitrión.

Con el fin de evitar problemas en torno al funcionamiento del servicio se habilitan reglas de firewall que permiten su correcto funcionamiento.

```
~/Escritorio/Tesis
sudo ufw allow 8086/tcp
Reglas actualizadas
Reglas actualizadas (v6)
```

De igual manera se instala el paquete de Python que permite enlazar el servicio mediante código.



Programación de nodos para del envío de datos servidor InfluxDB

En la siguiente sección se describen los scripts involucrados en el proceso de envío de datos desde los nodos hacia el servidor. El proceso de envío de datos opera en base a la siguiente función la cual proporciona una interfaz para interactuar con la base de datos de InfluxDB. Esta permite la escritura de datos por medio de la API de InfluxDB, utilizando como parámetros de conexión las variables de URL, token, organización y bucket. Este último hace referencia al lugar en donde se almacenan los datos de series temporales.

```python
from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS

class InfluxDBManager:
    def __init__(self, url, token, org, bucket):
        self.client = InfluxDBClient(url=url, token=token, org=org)
        self.write_api = self.client.write_api(write_options=SYNCHRONOUS)
        self.bucket = bucket

    def write_to_influxdb(self, measurement, tags, fields):
        point = Point(measurement)
        for key, value in tags.items():
            point = point.tag(key, value)
        for key, value in fields.items():
            point = point.field(key, value)
        self.write_api.write(bucket=self.bucket, record=point)

    def close(self):
        self.client.close()
```

Una vez establecida la función principal, se procede a configurar los procesos de envío de datos hacia la base de datos de InfluxDB de manera individual para cada nodo o dispositivo. El proceso en cuestión emplea un sistema de etiquetas para identificar de forma única cada dispositivo dentro del escenario, además de registrar las mediciones recopilados por cada uno de estos. Este enfoque permite una mayor organización y recuperación eficiente de los datos almacenados.

- Sensor de Temperatura

```python
server.influxdb_manager.write_to_influxdb(
    measurement="temperature_sensor",
    tags={},
    fields={
        "temperature": server.temperature_sensor
    }
)
```

- Enchufe Inteligente

```
server.influxdb_manager.write_to_influxdb(
    measurement="smart_plug",
    tags={"plug_id": plug_id},
    fields={
        "is_on": int(server.plugs[plug_id]["is_on"])
    }
)
```

- Termostato

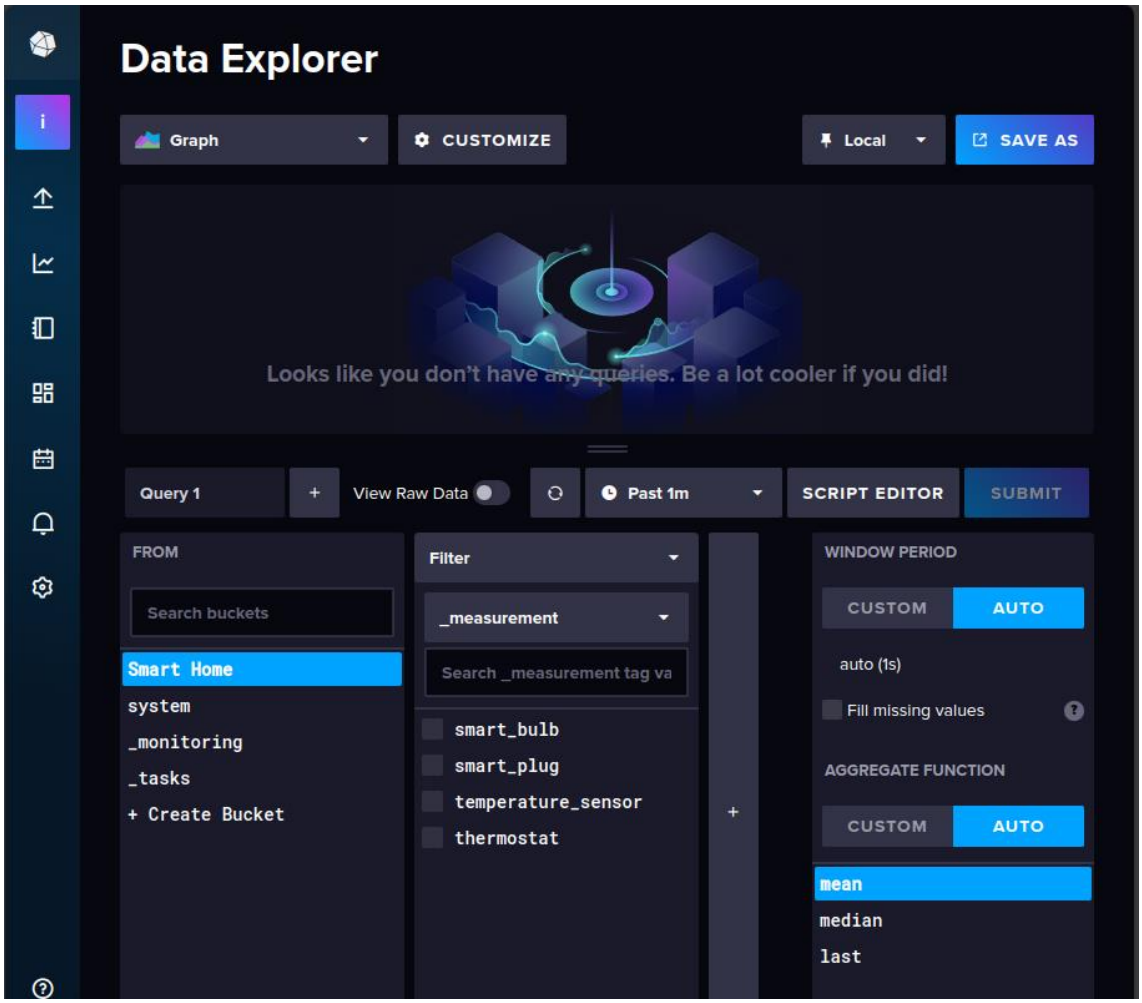
```
server.influxdb_manager.write_to_influxdb(
    measurement="thermostat",
    tags={},
    fields={
        "is_on": int(server.thermostat["is_on"]),
        "temperature": server.thermostat["temperature"]
    }
)
```

- Foco inteligente

```
server.influxdb_manager.write_to_influxdb(
    measurement="smart_bulb",
    tags={"bulb_id": bulb_id},
    fields={
        "is_on": int(server.bulbs[bulb_id]["is_on"]),
        "color": server.bulbs[bulb_id]["color"],
        "brightness": server.bulbs[bulb_id]["brightness"]
    }
)
```

Validación de envío de datos hacia InfluxDB

En el código de la topología generada mediante Mininet, se incorpora el script correspondiente a los procesos de envío de datos. Este paso permite iniciar el envío de información hacia InfluxDB. Las etiquetas asociadas a las mediciones de los dispositivos pueden visualizarse en la interfaz web de InfluxDB, donde se agrupan en función del bucket asignado y los identificativos únicos de cada dispositivo.



Configuración de Grafana

El proceso de instalación inicia con la adquisición de los claves de validación del software Grafana y la adición del repositorio de descarga a la lista de fuentes del sistema operativo Linux.

```
sudo wget -q -O /usr/share/keyrings/grafana.key https://apt.grafana.com/gpg.key
--accept-regex=REGEX
```

```
echo "deb [signed-by=/usr/share/keyrings/grafana.key] https://apt.grafana.com stable main" | sudo tee -
a /etc/apt/sources.list.d/grafana.list
-a
```

Posteriormente, se actualiza la lista de fuentes y procede a la instalación de los paquetes esenciales para el servicio de Grafana.

```
sudo apt-get install grafana
-B
```

```
grafana-server -v

Version 11.1.0 (commit: 5b85c4c2fcf5d32d4f68aaef345c53096359b2f1, branch: HEAD)
```

Finalmente, se inicia y habilita el servicio de Grafana, de modo que este se ejecute automáticamente al arrancar el sistema operativo.

```
sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-07-04 21:52:33 -05; 5s ago
       Docs: http://docs.grafana.org
   Main PID: 157988 (grafana)
      Tasks: 17 (limit: 18280)
     Memory: 47.9M
     CGroup: /system.slice/grafana-server.service
             └─157988 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfi>

jul 04 21:52:36 omen grafana[157988]: logger=ngalert.scheduler t=2024-07-04T21:52:36.033841499-05:00 >
jul 04 21:52:36 omen grafana[157988]: logger=ticker t=2024-07-04T21:52:36.033935017-05:00 level=info >
jul 04 21:52:36 omen grafana[157988]: logger=http.server t=2024-07-04T21:52:36.035491081-05:00 level=>
jul 04 21:52:36 omen grafana[157988]: logger=provisioning.dashboard t=2024-07-04T21:52:36.078954646-0>
jul 04 21:52:36 omen grafana[157988]: logger=provisioning.dashboard t=2024-07-04T21:52:36.078985166-0>
jul 04 21:52:36 omen grafana[157988]: logger=plugins.update.checker t=2024-07-04T21:52:36.266720873-0>
jul 04 21:52:36 omen grafana[157988]: logger=grafana.update.checker t=2024-07-04T21:52:36.26705681-05>
jul 04 21:52:36 omen grafana[157988]: logger=plugin.angulardetectorsprovider.dynamic t=2024-07-04T21:>
jul 04 21:52:36 omen grafana[157988]: logger=grafana-apiserver t=2024-07-04T21:52:36.456883076-05:00 >
jul 04 21:52:36 omen grafana[157988]: logger=grafana-apiserver t=2024-07-04T21:52:36.45725561-05:00 l>
lines 1-20/20 (END)
```

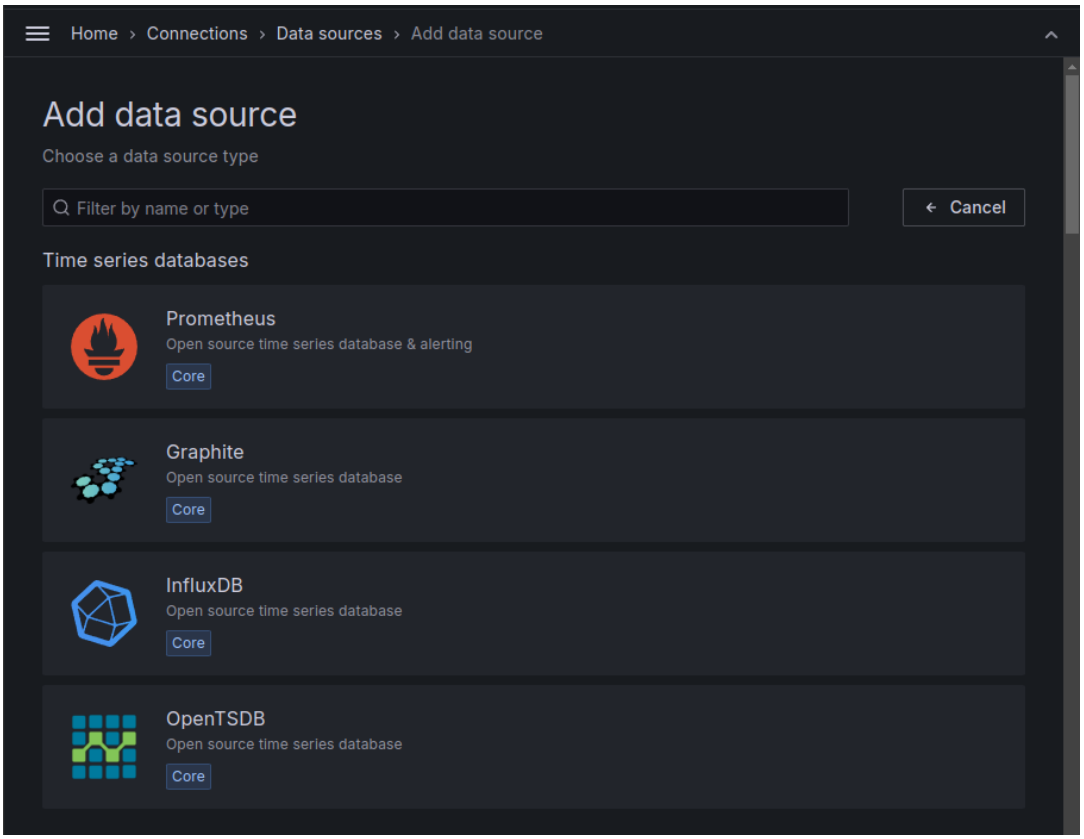
Para evitar problemas de comunicación en torno al servicio se añaden reglas de firewall que permiten las conexiones entrantes por el puerto 3000 correspondiente al servicio de Grafana.

```
sudo ufw allow 3000/tcp
Reglas actualizadas
Reglas actualizadas (v6)
```

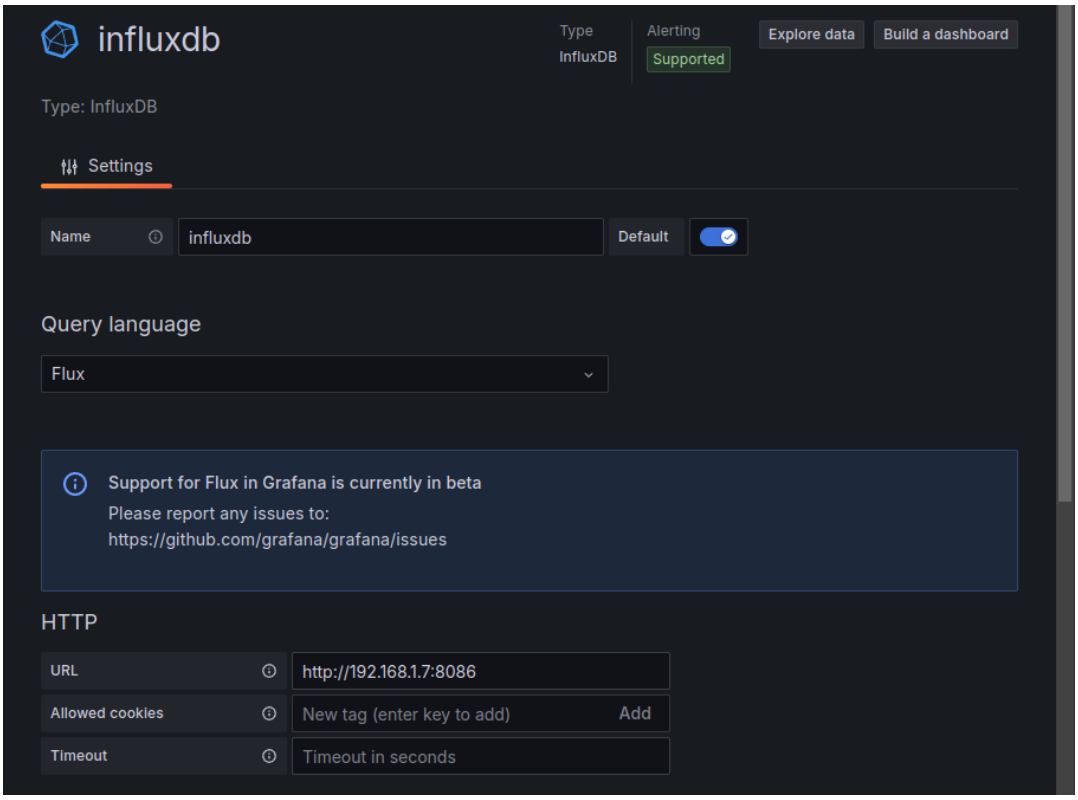
Configuración de Dashboards en Grafana

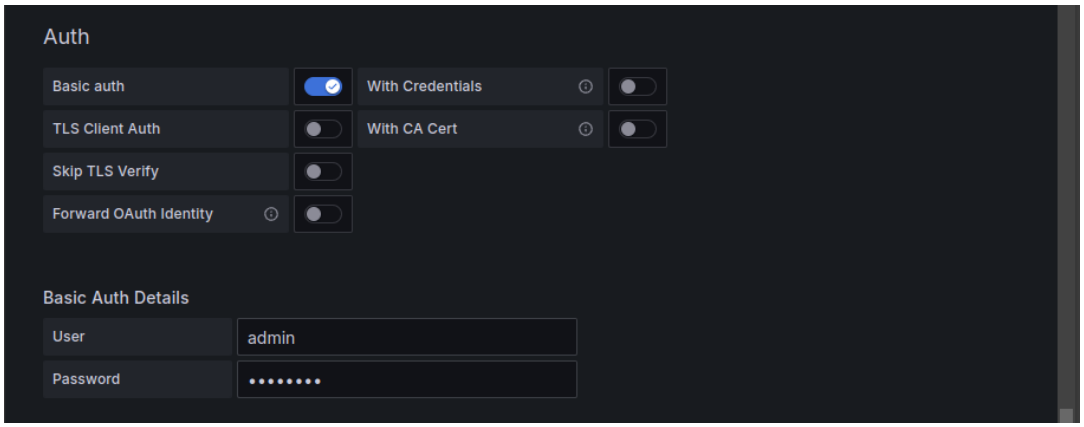
Culminado el apartado de instalación, se continua con la creación de las gráficas que permitirán la visualización de los datos enviados desde los nodos de la red simulada por medio de los siguientes pasos.

Se accede a la interfaz gráfica del servicio y, en el apartado de fuentes de información, se selecciona InfluxDB.

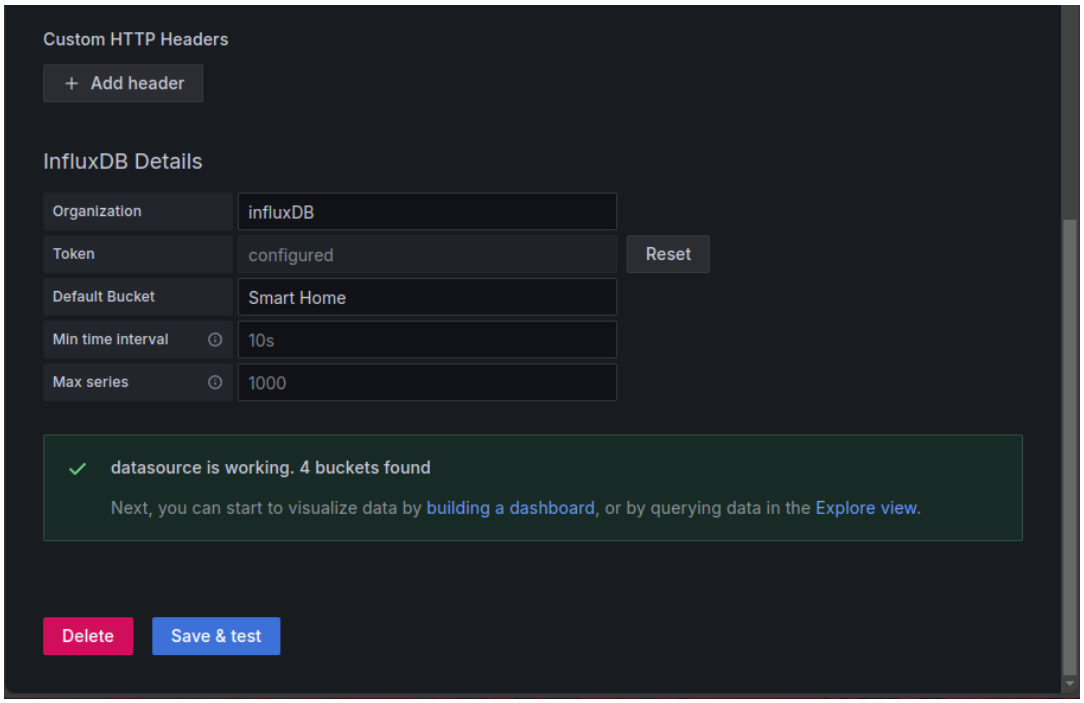


En el apartado de configuración, se establecen los parámetros de comunicación. En este caso, se define el nombre, la URL y las credenciales de autenticación para el proceso de consultas.



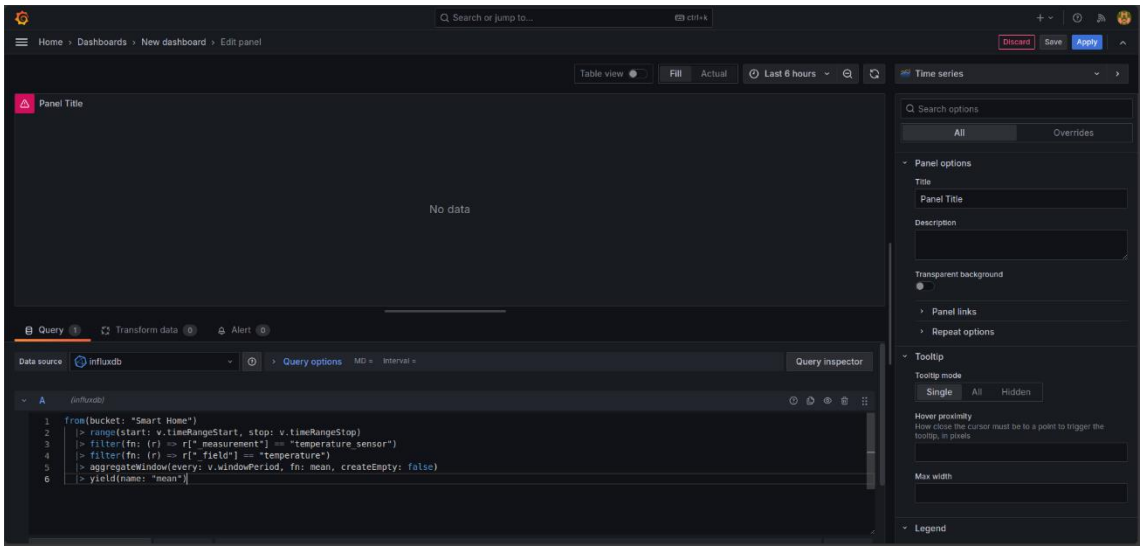


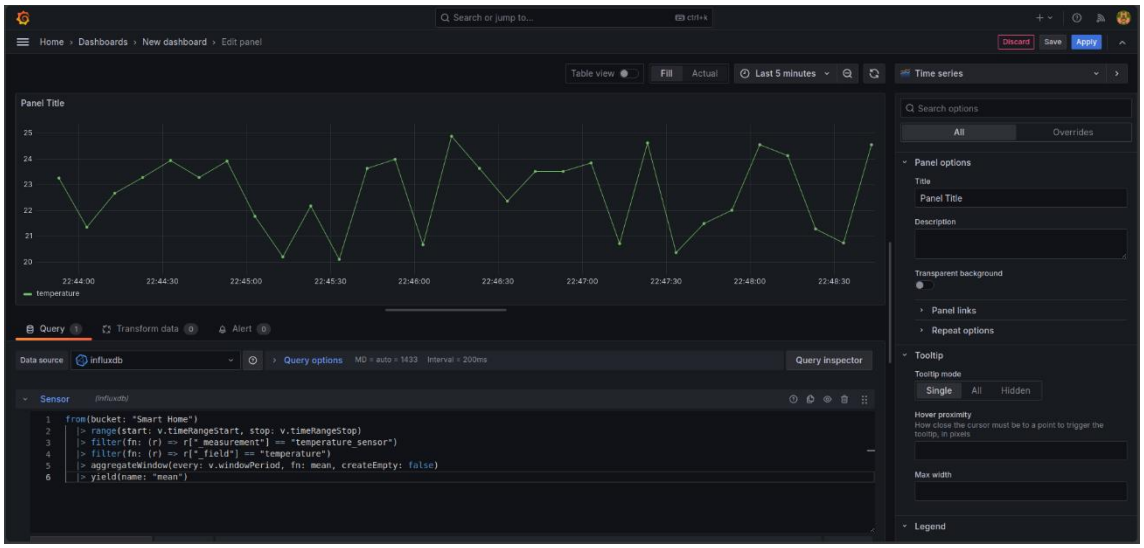
En esta misma sección, también se especifican los detalles de la base de datos de InfluxDB, donde se indica la organización a la que pertenece y el contenedor (bucket) en el que almacenan los datos de series temporales.

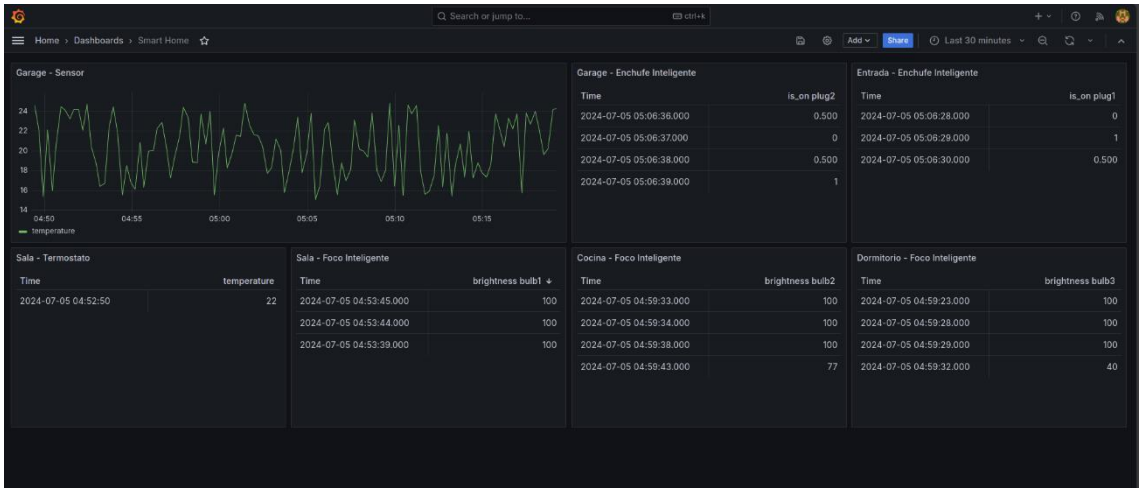


En base a la información del bucket de InfluxDB, se genera una consulta para recuperar los datos de series temporales albergados en él.

Se importa a Grafana el formato de la consulta generada en InfluxDB para cada uno de los dispositivos (nodos). De esta manera, se generan las gráficas de monitoreo que muestran las mediciones y valores detectados por los dispositivos.







Configuración de Suricata

La instalación y configuración del sistema de detección de intrusos (IDS) Suricata inicia con la adición del repositorio del paquete de software a la lista de fuentes del sistema operativo, procediendo luego con la instalación el paquete Suricata.

```
~/Escritorio/Tesis  sudo add-apt-repository ppa:oisf/suricata-stable
```

```
~/Escritorio/Tesis  sudo apt install suricata                    33s
```

Tras la instalación, se habilita el servicio de manera que este se ejecute automáticamente una vez se inicie el sistema operativo. A su vez se verifica el estado del servicio para corroborar el correcto funcionamiento de este.

```
~/Escritorio/Tesis  sudo systemctl enable suricata.service                    17s
[sudo] contraseña para eddpic:
suricata.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable suricata
```

```
~/Escritorio/Tesis  sudo systemctl status suricata                    5s
● suricata.service - LSB: Next Generation IDS/IPS
     Loaded: loaded (/etc/init.d/suricata; generated)
     Active: active (exited) since Wed 2024-07-17 00:34:16 -05; 44min ago
       Docs: man:systemd-sysv-generator(8)
      Tasks: 0 (limit: 18280)
     Memory: 0B
     CGroup: /system.slice/suricata.service

jul 17 00:34:16 omen systemd[1]: Starting LSB: Next Generation IDS/IPS...
jul 17 00:34:16 omen suricata[11926]: Starting suricata in IDS (af-packet) mode... done.
jul 17 00:34:16 omen systemd[1]: Started LSB: Next Generation IDS/IPS.
```

Una vez completada la instalación y habilitación del servicio, se procede a configurar los parámetros de Suricata de manera que este monitoree la red generada por Mininet. El desarrollo de este apartado requiere de la identificación previa de la interfaz por la cual transita el tráfico de red en Mininet.

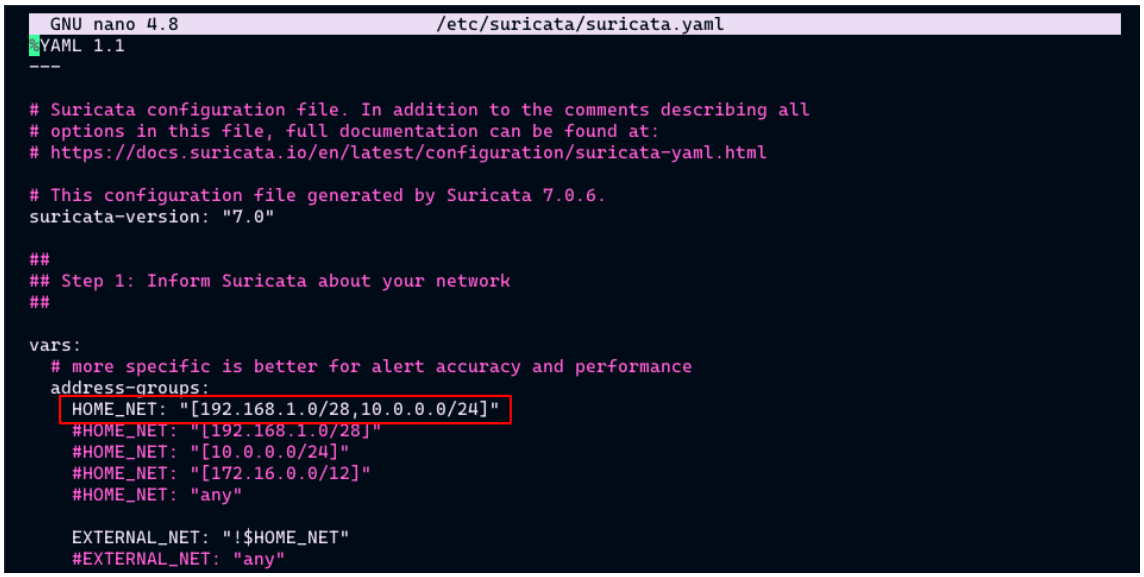
```
mininet-wifi> links
ap1-eth2<->serv-eth0 (OK OK)
ap1-eth3<->serv2-eth0 (OK OK)
ap1-eth4<->ips-eth0 (OK OK)
sala_n1-wlan0<->wifi (use iw/iwconfig to check connectivity)
sala_n2-wlan0<->wifi (use iw/iwconfig to check connectivity)
host1-wlan0<->wifi (use iw/iwconfig to check connectivity)
coci_n1-wlan0<->wifi (use iw/iwconfig to check connectivity)
dorm_n1-wlan0<->wifi (use iw/iwconfig to check connectivity)
entr_n1-wlan0<->wifi (use iw/iwconfig to check connectivity)
gara_n1-wlan0<->wifi (use iw/iwconfig to check connectivity)
gara_n2-wlan0<->wifi (use iw/iwconfig to check connectivity)
nat0-eth0<->ap1-eth5 (OK OK)
```

```
32: ap1-eth2@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP
group default qlen 1000
    link/ether 8e:f9:b1:61:90:42 brd ff:ff:ff:ff:ff:ff link-netnsid 8
    inet6 fe80::8cf9:b1ff:fe61:9042/64 scope link
       valid_lft forever preferred_lft forever
33: ap1-eth3@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP
group default qlen 1000
    link/ether 82:e5:98:d9:d6:4f brd ff:ff:ff:ff:ff:ff link-netnsid 9
    inet6 fe80::80e5:98ff:fed9:d64f/64 scope link
       valid_lft forever preferred_lft forever
34: ap1-eth4@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system state UP
group default qlen 1000
    link/ether 9a:7b:1b:44:5f:29 brd ff:ff:ff:ff:ff:ff link-netnsid 10
    inet6 fe80::987b:1bff:fe44:5f29/64 scope link
       valid_lft forever preferred_lft forever
35: ap1-eth5@nat0-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-system sta
te UP group default qlen 1000
    link/ether 86:1a:e0:5a:26:7c brd ff:ff:ff:ff:ff:ff
    inet6 fe80::841a:e0ff:fe5a:267c/64 scope link
       valid_lft forever preferred_lft forever
36: nat0-eth0@ap1-eth5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group defaul
t qlen 1000
    link/ether be:ce:09:9c:71:23 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.12/8 brd 10.255.255.255 scope global nat0-eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::bcce:9ff:fe9c:7123/64 scope link
       valid_lft forever preferred_lft forever
```

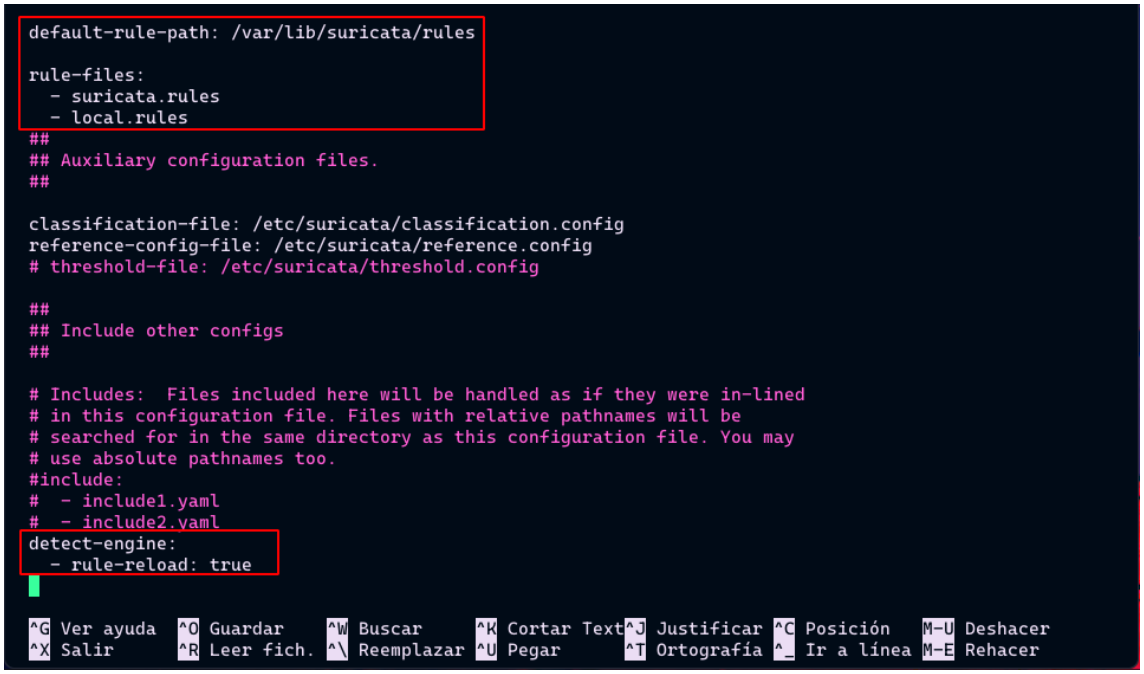
Una vez identificada la interfaz, se modifica el archivo de configuración de Suricata. En esta etapa, se asocia la interfaz y se designa el segmento de red a monitorear. Adicionalmente, se establecen las rutas de los directorios donde se almacenan las reglas y se habilita la opción de recarga automática de estas.

```
~/Escritorio/Tesis  sudo nano /etc/suricata/suricata.yaml
```

```
 GNU nano 4.8                    /etc/suricata/suricata.yaml

# Linux high speed capture support
af-packet:
  #- interface: wlo1
  - interface: ap1-eth4
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
```


GNU nano 4.8 /etc/suricata/suricata.yaml
%YAML 1.1

Suricata configuration file. In addition to the comments describing all
options in this file, full documentation can be found at:
https://docs.suricata.io/en/latest/configuration/suricata-yaml.html
This configuration file generated by Suricata 7.0.6.
suricata-version: "7.0"
##
Step 1: Inform Suricata about your network
##
vars:
more specific is better for alert accuracy and performance
address-groups:
HOME_NET: "[192.168.1.0/28,10.0.0.0/24]"
#HOME_NET: "[192.168.1.0/28]"
#HOME_NET: "[10.0.0.0/24]"
#HOME_NET: "[172.16.0.0/12]"
#HOME_NET: "any"
EXTERNAL_NET: "!$HOME_NET"
#EXTERNAL_NET: "any"


default-rule-path: /var/lib/suricata/rules
rule-files:
- suricata.rules
- local.rules
##
Auxiliary configuration files.
##
classification-file: /etc/suricata/classification.config
reference-config-file: /etc/suricata/reference.config
threshold-file: /etc/suricata/threshold.config
##
Include other configs
##
Includes: Files included here will be handled as if they were in-lined
in this configuration file. Files with relative pathnames will be
searched for in the same directory as this configuration file. You may
use absolute pathnames too.
#include:
- include1.yaml
- include2.yaml
detect-engine:
- rule-reload: true
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Text ^J Justificar ^C Posición M-U Deshacer
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^T Ortografía ^_ Ir a línea M-E Rehacer