ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS Y DE COMPUTACIÓN

HERRAMIENTAS DE SEGURIDAD DEFENSIVA Y OFENSIVA PARA REDES LORAWAN

DESARROLLO DE UN PROTOTIPO DE MÓDULO DE ANÁLISIS DE TRÁFICO BASADO EN WIRESHARK PARA DETECCIÓN DE ATAQUES DE DENEGACIÓN USANDO INSPECCIÓN DE TRAMAS LORAWAN QUE PROVEA UNA CAPA DE INTEGRACIÓN API REST.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

CHRISTIAN JAVIER AMAGUAÑA AGUILAR

christian.amaguana@epn.edu.ec

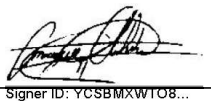
DIRECTOR: JHONATTAN JAVIER BARRIGA ANDRADE

Jhonattan.barriga@epn.edu.ec

DMQ, SEPTIEMBRE 2022

CERTIFICACIONES

Yo, Christian Javier Amaguaña Aguilar declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



Signer ID: YCSBMXWTO8...

Christian J. Amaguaña

Certifico que el presente trabajo de integración curricular fue desarrollado por Christian Javier Amaguaña Aguilar, bajo mi supervisión.

Jhonattan Barriga

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Christian Javier Amaguaña Aguilar

DEDICATORIA

Dedico este mi trabajo de integración curricular primeramente a Dios quien me ha permitido mantenerme firme en todo este tiempo de estudio y sus bendiciones en esta carrera, luego a mi Madre quien con su esfuerzo y dedicación me ha formado el hombre que soy y que a pesar de las dificultades a travesadas durante todo este tiempo se mantuvo firme y me dio la oportunidad de estudiar y formarme profesionalmente, a mis hermanos quienes han estado junto a mi en todo momento y su apoyo emocional ha sido de gran valor en este tiempo de estudio, el cual me ha ayudado a levantarme en los momentos difíciles en este camino y finalmente a mis amigos quienes al igual que mis hermanos me apoyaron en todo momento en este camino para poder cumplir con mi objetivo.

Christian J. Amaguaña

AGRADECIMIENTO

Agradezco al estimado MSc. Jhonattan Barriga por su motivación y mentoría en todo momento de esta etapa del desarrollo de este proyecto, el cual ha sido fundamental por su guía para el avance y éxito de este proyecto, que por las dificultades que se mostraron también durante el desarrollo del mismo supo cómo motivarme y aconsejarme para continuar adelante con este proyecto.

A mi familia por su amor, apoyo y ejemplo de fortaleza en todo momento brindado hacia mí para poder avanzar en este camino.

Y finalmente a mis amigos y compañeros quienes a lo largo de la carrera han sido de apoyo y ejemplo en este camino como también su apoyo emocional para salir adelante.

Christian J. Amaguaña

ÍNDICE DE CONTENIDO

INDICE DE FIGURAS

RESUMEN

Cada día la tecnología sigue avanzando y con ello los dispositivos inteligentes para nuestro hogar, con esto llega el nacimiento del IoT o el internet de las cosas que han permitido conectar a miles de personas, ahora que en la actualidad ha permitido implementar casi todo lo que hay en el hogar, sin embargo, por esto se tiene la responsabilidad que buscar soluciones para la seguridad de cada uno de los dispositivos ya que al igual que la tecnología avanza también crece la ciber delincuencia buscando nuevos oportunidades de robar nuestra información.

Dentro de los mayores ataques que han existido y que han avanzado ha sido la denegación de servicios distribuidos, que se ha estado aplicando a estos dispositivos IoT en la actualidad, por su tecnología que es actual y sigue avanzando aún queda conocer más sobre esta tecnología y su forma de cuidar la manera en que se maneja la información a través de los mismos.

Junto con sus protocolos LoRaWAN que se han implementado en estos dispositivos y los paquetes de información que maneja se ha tomado en cuenta el desarrollo de un sniffer y en futuro un Api Rest que permita tanto capturar estos paquetes para conocer su destino e información que contiene como almacenarlas en una base de datos que se pueda consultar esta información para que pueda ser analizada y conocer si es un ataque de denegación de servicio.

Basándose en los actuales principios del open source o código abierto e implementando la mejora continua mediante la tecnología LEAN para el desarrollo de este proyecto a la par del framework Scrum para un correcto uso de cada fase explicada dentro de este proyecto.

PALABRAS CLAVE: Denegación de servicio distribuido, IoT, LoRaWAN, open source, sniffer, api rest.

ABSTRACT

Every day technology continues to advance and with it the smart devices for our home, with this comes the birth of the IoT or the internet of things that have allowed thousands of people to connect, now that it has currently allowed the implementation of almost everything that there is in the home, however, for this reason it is the responsibility to seek solutions for the security of each of the devices since, as technology advances, cybercrime also grows, looking for new opportunities to steal our information.

Among the biggest attacks that have existed and that have advanced has been the distributed denial of services, which has been applied to these IoT devices today, due to its technology that is current and continues to advance, there is still more to know about this technology and their way of taking care of the way in which the information is handled through them.

Along with its LoRaWAN protocols that have been implemented in these devices and the information packets it handles, the development of a sniffer and in the future a Rest Api has been taken into account that allows both capturing these packets to know their destination and the information they contain. store them in a database so that this information can be consulted so that it can be analyzed and find out if it is a denial-of-service attack.

Based on the current principles of open source or open source and implementing continuous improvement through LEAN technology for the development of this project along with the Scrum framework for a correct use of each phase explained within this project.

KEYWORDS: denial-of-service attack, IoT, open source, rest api, sniffer.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Este proyecto se tratará sobre cómo podemos visualizar el estado de los ataques de las redes LoRaWAN, en especial los ataques de denegación de servicios y como pueden influir en estas redes que son mayormente utilizadas para los dispositivos inteligentes(IOT), con esto permitirá conocer más sobre la seguridad defensiva para estos dispositivos mediante la construcción de mecanismo de prevención como también de contención para los ataques de denegación de servicios, también servirá para conocer y comprender estos tipos de ataques para evaluar el impacto que puede tener en las redes LoRaWAN.

Estas redes son muy utilizadas al momento de implementar Smart parking o Smart Cities, ya que actualmente se tiene la iniciativa de automatizar las mayorías de acciones que permiten facilitar a los usuarios en su diario vivir, es por ello que este protocolo permite que se pueda llevar a cabo, ya que no tiene un costo por su uso y tiene su arquitectura como su infraestructura abierta el cual permite su uso y realizar diferentes tipos de modificaciones.

Es por este motivo que se debe mantener la seguridad dentro de estos dispositivos y para esto se nos permite ejecutar mejores técnicas para la protección y la adaptación a diferentes situaciones, al realizar una evaluación de seguridad de sus redes a través del desarrollo de un API REST para el análisis de tráfico basado en Wireshark permitirán detectar estos paquetes de información para conocer lo que está captando nuestro dispositivo IOT, que también se puede realizar con la evaluación de tramas de LoRaWAN que junto con lo mencionado anteriormente nos permitirá obtener un prototipo de inspección de estas redes.

1.1 Objetivo general

Implementar un prototipo que permita analizar el tráfico basado en Wireshark para la detección y evaluación de los ataques de denegación, inspeccionando las tramas de LoRaWAN a través de la integración de este módulo a un prototipo de API REST para una mejor visualización del análisis.

1.2 Objetivos específicos

1. Comprender como funciona el análisis de trafico de Wireshark para los ataques de denegación.
2. Desarrollar un prototipo de sniffer que permita visualizar el análisis de las tramas de las redes LoRaWAN.
3. Integrar el prototipo del análisis de trafico de datos con el API REST para una mejor comprensión del funcionamiento de las tramas LoRaWAN.

1.3 Alcance

Con el presente componente comprende en la construcción de un prototipo de un módulo de inspección de tramas LoRaWAN con una interfaz API REST que permitirá detectar los paquetes anómalos de las redes LoRaWAN y pueda leerlos de tal manera que pueda reconocer los tipos de paquetes que ingresan para así poder captar los datos necesarios de los paquetes que ingresan al dispositivo, se trabajar en la arista de seguridad ofensiva, se aplicará una adaptación de la metodología LEAN para el desarrollo del prototipo ya que esta metodología contiene los principios necesarios para un correcto desarrollo, que permitirá aplicar los principios de acuerdo a esta metodología que son: optimizar, eliminar algún desperdicio, calidad, aprendizaje, entregas rápidas para ser evaluadas y la mejora de cada fase del proyecto, al igual que en lo largo de la ejecución del proyecto se utilizara la metodología de desarrollo SCRUM, debido que el desarrollo del prototipo de software es pequeño y en un tiempo de ejecución corto, considerando que estas metodología se pueden ir adaptando en el transcurso del desarrollo del proyecto evitando así los riesgos de fracaso, a continuación se describe en a cada etapa lo que se realizará durante el desarrollo del proyecto.

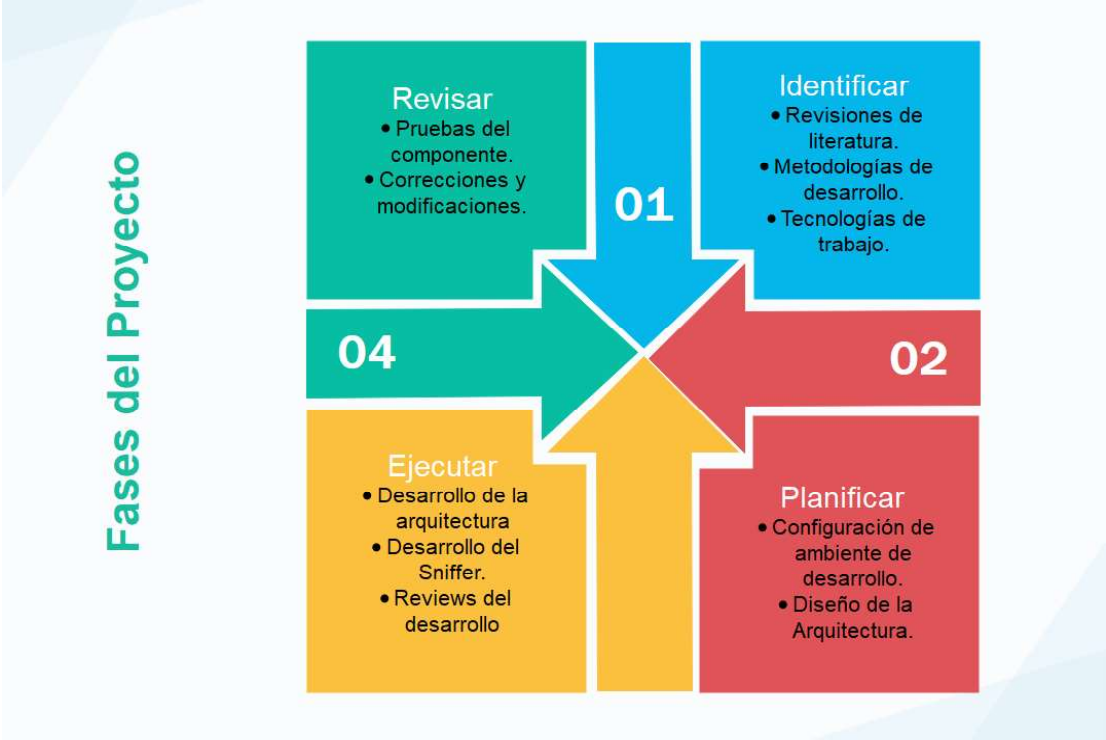


Figura 1 Fases del proyecto

1.3.1 Limitaciones

De acuerdo a las revisiones de literatura que se ha propuesto en la primera fase del proyecto, se tiene en cuenta ciertas limitaciones que pueden existir al momento de la ejecución del proyecto, ya que en las librerías a utilizar para el desarrollo del sniffer como la integración del API REST y su manejo en ambientes virtuales de los frameworks que se utilizaron para comprobación del mismo, se observó que la instalación de las librerías tanto Pyshark como Tshark en sus ambientes virtuales no se permitía o existía el error de que identificarlos o tomarlos como parte de las librerías manejadas del framework, además de la insuficiente información sobre la librería Pyshark y su uso por lo que aun esta en desarrollo, como también Tshark al ser una variante de Wireshark que es manejada por línea de comandos, han limitado la manera de integrar esta función directamente con el API REST, sin embargo, el análisis de las tramas LoRaWAN se realiza conforme a lo planeado, el ingresar la información directa al API REST se lo puede realizar de manera manual para una mejor interacción, pero es una posibilidad que después se pueda mejorar la implementación de acuerdo al avance del desarrollo de estas librerías y la aceptación de otras librerías dentro de los frameworks descritos también en este proyecto.

1.4 Marco teórico

En ocasiones y más en este análisis de las redes LoRaWAN que permiten la conexión de distintos dispositivos inteligentes(IOT), se debe tomar en cuenta la seguridad de los mismo porque al ser un dispositivo conectado inalámbricamente puede sufrir ataques de denegación de servicio que son comunes en estos aparatos, lo principal dentro de la seguridad de red como puede los firewalls y herramientas IDS/IPS se concentran en verificar todo el tráfico que se genera, existen soluciones en este análisis de tráfico de red se mantienen intercambiando independientemente de que tipo de paquete sea, puede ser TCP/IP convencionales.

Ahora toda la innovación de los sistemas de internet de las cosas(IOT) en muchas ocasiones pude ser un blanco para poder ser víctima de ataques, es por esto que junto el analizador de tráfico como Wireshark no puede ayudar a poder analizar estar tramas o tráfico que puede estar generando y de esta forma detectar los ataques que en especial se concentrará en los ataques DoS, como se conoce Wireshark es un poderoso analizador de red o protocolo porque nos permite analizar la distintas estructuras y que para las redes LoRaWAN nos serán de ayuda para detectar estos ataques, para ello Wireshark nos ayuda a poder analizar estos ataques en la redes TCP Y LPWAN que son las más utilizadas para los dispositivos IOT.

Como se conoce los ataques DoS es un tipo de ataque que el atacante busca interrumpir temporal o permanentemente todos los servicios para que no estén disponibles para los usuarios que en este caso seria los dispositivos de IOT, esto se realiza mediante peticiones o solicitudes al servidor con la intención de saturarlo de su capacidad permitida, esto hace que se inunde con todo el tráfico generado por distintas fuentes y es por eso que es imposible detener este tipo de ataque, aunque este ataque no tiene la cualidad de robar información tiene como objetivo es bloquearlo y perder de esta manera tiempo y dinero en muchas ocasiones cuando estos ataques son realizados a grandes escalas, junto con esta herramienta como es Wireshark nos permite analizar este tipo de ataque, es importante tener en cuenta el mantener actualizados todos los parches de seguridad para evitar esos ataques.[1]

Wireshark para la detección de ataques DoS en redes TCP

Como partimos desde principio, con la herramienta Wireshark va permitir capturar el tráfico de la red, ahora cuando estos paquetes se observan detenidamente podemos captar que hay gran cantidad de conexiones TCP con su indicador activado el cual se genera desde una solo dirección IP, todas estas solicitudes TCP no logra reconocer el servidor. Además,

que se puede observar un período corto de tiempo como también cuando dispositivo realiza una gran cantidad de intentos para conectarse la máquina y no logra ser reconocido por parte del servidor, y es por esto que no se logra completar el protocolo de enlace porque durante ese tiempo de espera el servidor trata de reconocer estas solicitudes, pero con todos esos intentos siguen llegando más paquetes que activan las nuevas conexiones y esto se lo reconoce como un ataque de DoS. [1]

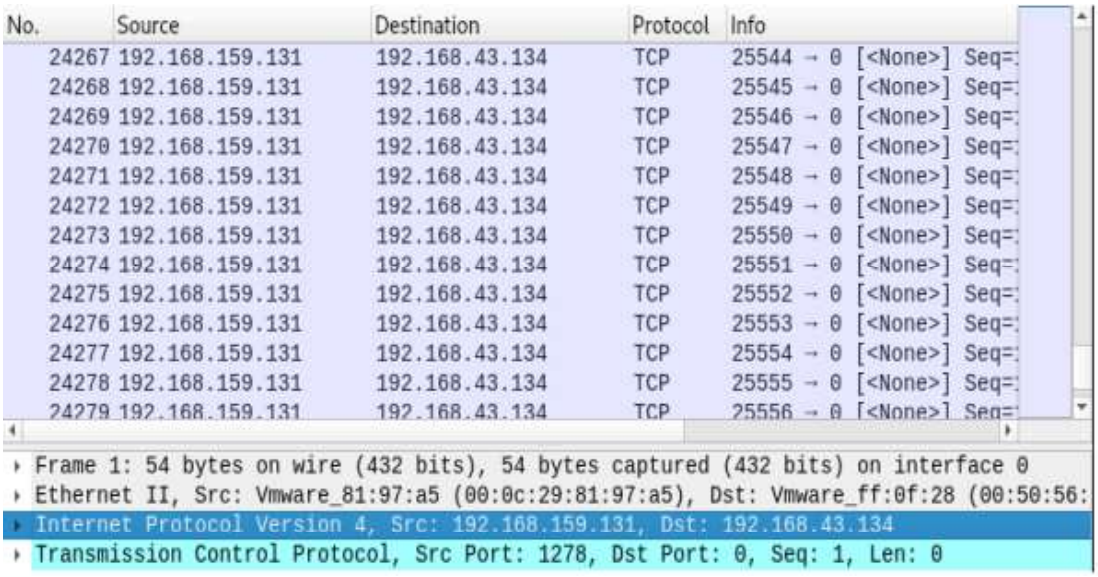


Figura 2 Captura de ataque DoS en redes TCP[1]

Wireshark para la detección de ataques DoS en redes LPWAN

Debemos tomar en cuenta que tanto las redes WAN como LPWAN trabajan dentro de las capas de la 1 a la 3 permitiendo de esta manera un esquema de direccionamiento uniforme que además llegan a abarcar grandes lugares a un bajo consumo energético, que envían paquetes inalámbricamente para su reconocimiento, que en este caso serían los dispositivos IOT, el cual permite la conexión de decenas o centenares de nodos.[3]

Entonces este análisis se realiza como el de TCP ya que el objetivo es el mismo el poder detener un servicio para el usuario y no tener la disponibilidad ya que se satura con todas

las solicitudes de conexión de varios nodos que hace que responda con mayor lentitud o no responda. [3]

Ahora Wireshark permite captar el envío de paquetes cuando se genere el tráfico de la red, los paquetes que se observan detenidamente y podemos captar que hay gran cantidad de conexiones de nodos con su indicador el cual se genera desde varios nodos que piden ser conectados al servidor, todas estas solicitudes no logran reconocer el servidor por el cual no ralentiza o deja de responder y se captan los paquetes para poder ser reconocidos como un ataque de DoS. [4]

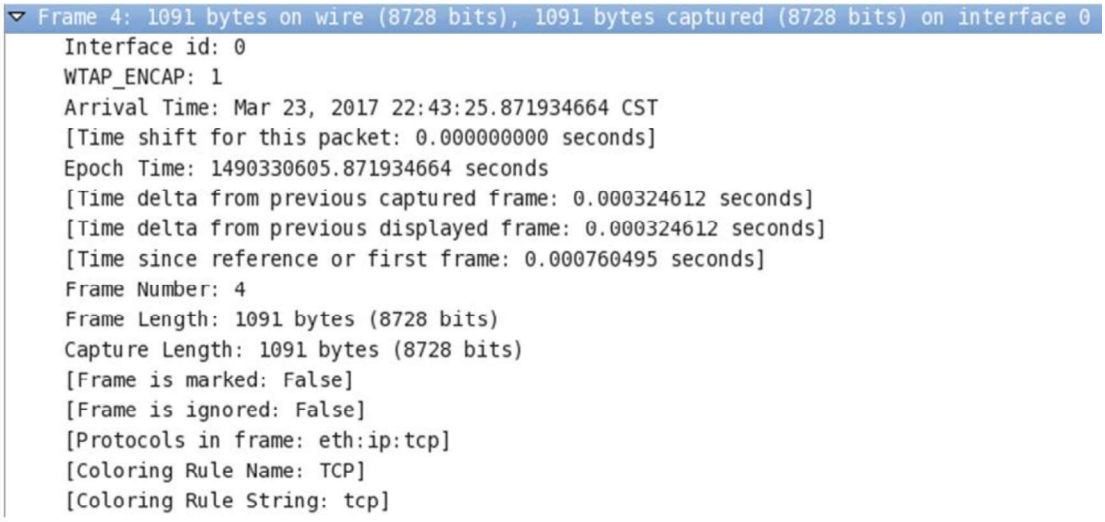


Figura 3 Información del paquete del nodo capturado.

Definiciones de API REST

1) Un API REST es conocida como una interfaz de programación de aplicaciones que al igual se ajusta a los límites que se pongan dentro de la arquitectura REST y que da acceso a la interacción con algunos servicios web, junto con las API son un grupo de definiciones y protocolos que se usan para diseñar e integrar aquellas funciones en el software, es por esto que le permite interactuar con un dispositivo o sistema para obtener datos o ejecutar un funcionamiento de la aplicación y así pueda comprender la solicitud pedida.[5]

   Es por esto que REST es un grupo de pautas que logran implementarse según lo pedido, por este motivo las API REST sin más rápidas y ligeras, que cuenta con una capacidad enorme de ajuste ya que por esto es ideal para el internet de las cosas conocidas como IOT y el desarrollo de aplicaciones de dispositivos móviles.[5]

2) Las API REST son con conjunto de reglas que definen la manera de conectarse a los dispositivos o aplicaciones y de cómo pueden comunicarse entre ellos, además, cumple con los principios de diseño del estilo de las arquitecturas REST o transferencia de estado representacional por el cual se conocen también con el nombre de API RESTful.[6]

   Además, que brinda una forma más flexible y ligera de poder integrar las aplicaciones de manera que han surgido como un método común para conectar componentes en las arquitecturas de microservicios ya que existe demanda de mejores experiencias del usuario y más aplicaciones dentro de las TI y IOT. [6]

2 METODOLOGÍA

Durante estas 4 fases que fueron descritas en la parte del alcance se tomará como framework de Scrum para el desarrollo de este proyecto, ya que el tamaño del desarrollo es pequeño y en tiene un tiempo de ejecución corto. También nos permitirá ejecutar pruebas de integración de los resultados que se puedan obtener, dando como resultados las detecciones o correcciones de problemas que puedan existir durante el desarrollo del mismo. Además, este framework nos ayudará a adaptarnos a los problemas que puedan surgir en el desarrollo del proyecto, permitiendo una reducción de los riegos que se puedan presentar, por último, ya que es una nueva tecnología en desarrollo no es posible estimar la complejidad de la librería utilizada, los distintos frameworks para la realización de Apis como también el uso de metodologías tradicionales como cascada fueron desechas ya que no se podía adaptar al desarrollo de este proyecto.

Además, durante estas etapas planificadas y las ejecuciones de las mismas se aplicará el framework de desarrollo scrum, como se muestra en la figura 4 se observa cada uno de los pasos con sus respectivas tareas a realizar en este proyecto.

Figura 4 Metodología aplicada al proyecto

2.1 Identificar

En esta primera etapa se ha dispuesto revisar la literatura con respecto a diferentes temas que ayudarán con el entendimiento sobre las redes LoRaWAN, ataques DDoS, Wireshark y API REST que se puede utilizar en el futuro para mejores implementaciones.

Se ha dispuesto una búsqueda a través de preguntas específicas para cada tema propuesto para que se pueda obtener distintas literaturas referentes al tema, conforme se realizó la búsqueda por palabras claves para que sean más específicas las búsquedas, en la figura 5 se puede apreciar la manera en la que las preguntas y palabras claves corregidas se utilizaron para las búsquedas.

	Palabras Claves
¿Cómo funciona Wireshark respecto a la detección de ataques de DoS?	Wireshark ; attacks; DoS detection
¿Cómo se puede utilizar API REST en la contruccion de herramientas y componentes de ciberseguridad?	API REST; cybersecurity tools API REST; components
¿Cómo afectan los ataques DoS a las redes LoRaWAN?	LoRaWAN; DoS attack
¿Cuál es el funcionamiento de los ataques DoS en la redes LoRaWAN?	dos attacks architecture; ddos anatomy;ddos schema

Figura 5 Formulaciones de preguntas y palabras claves.

Se obtuvo distintas literaturas de diferentes fuentes de búsqueda que puedan tener sustento científico de cada uno de los temas a entender para el desarrollo de este proyecto, las fuentes que se utilizaron fueron las siguientes.

2)	**Fuentes de información**
	IEEEXplore digital
	ScienceDirect.com
	SpringerLink

Figura 6 Fuentes de búsqueda

Además de todo esto se propuso distintos criterios de exclusión para obtener mejores resultados actuales y que sean referidos a los temas propuestos antes, ya que al transcurso de las búsquedas se optó por estos criterios para tener sustento científico de los temas, además, de la generalización del tema que la seguridad de los sistemas IOT. A continuación, se explicará la importancia de cada tema.

2.1.1 LoRaWAN y ataques de denegación de servicios (DDoS)

Los dispositivos inteligentes(IOT) han permitido ofrecer un potencial enorme para poder conectar miles de millones de dispositivos alrededor del mundo, con una infraestructura que permite la vinculación a distintos servicios como a sistemas empresariales. Por tal magnitud de dispositivos se ha pensado en la seguridad de los mismo conectando sensores para el monitoreo de distintos equipos relacionados con el IOT que da enormes ventajas para los usuarios hoy en día.

Una de las tecnologías que es muy importante para que estos dispositivos se puedan conectar es la red de área amplia baja potencia conocida como LPWAN, además hay frecuencias que permiten conexiones de largas distancias como por ejemplo el conectar nodos de sensores inteligentes a una puerta de enlace. Como se conoce esta puerta de enlace debe devolver estos datos de internet para un adecuado análisis de esta información que puede llegar.

Ahora dentro de la paca física se encuentra la arquitectura de la red LoRaWAN que es una topología de estrella de estrellas que ayudan a que las puertas de enlace puedan transmitir mensajes entre dispositivos finales y un servidor de red central.

Esta comunicación inalámbrica que existe entre los dispositivos se aprovecha de las características de largo alcance de la capa física LoRa, ya que el enlace da un solo salto entre el dispositivo final y una o varias puertas de enlace, todos estos modos son capaces de tener una comunicación bidireccional como también existe soporte para grupos de direccionamiento multidifusión para que todo tenga un uso eficiente durante las tareas.

Como se ha descrito anteriormente una área clave para todo este tipo de conectividad de baja potencia y de largo alcance son las denominadas ciudades inteligentes, por la utilización de estos dispositivos que ayudan para un sin número de situaciones que pueden servir dentro de este ambiente como la capacidad de insertar sensores inteligentes inalámbricos para la calidad del aire, la densidad de tráfico y el transporte, aunque también se puede utilizar en toda una infraestructura urbana que permita proporcionar información clave de la actividad de una ciudad.

Es por esto que LoRaWAN como implementación de LoRa se convirtió en la tecnología LPWAN más utilizada en todas las bandas sin licencia por debajo de 1GHz, permitiendo la instalación de sensores alimentados por paneles solares y así expandirse a través del internet de las cosas.

Hablando de los ataques de denegación de servicio distribuido, existen distintas formas de explicarlo, se pondrá un ejemplo, se imagina que hay un estadio donde se jugará la final del mundial de futbol, como se sabe este estadio cuenta con varias puertas para que los hinchas puedan ingresar, siempre se pone una hora antes de la hora oficial del partido para que la gente pueda ir ingresando lentamente pero ordenado de forma constante, existen personas en las puertas pendientes de revisar los boletos de entrada aunque es una labor muy cansada continúan haciendo su trabajo, ahora faltando 10 minutos para que el juego empiece, muchas personas llegan con grandes cantidades de entradas falsas y los encargados de revisar las entradas no pueden con tantas personas a la vez, entonces las personas que si contaban con su boleto original no podrán entrar al estadio.

Las personas encargadas de prohibir el paso al estadio se complican para diferenciar entre los boletos originales y los falsos, además, por el cansancio de todo el día se complica aún más. Conforme con esto se explica a los hinchas que están ingresando al estadio que se llevara más tiempo para ingresar, ya necesitan verificar cada boleto que sea original, entonces las personas con las entradas falsas intentan ingresar en medio del desorden y por la fuerza y es por esto que las entradas se ven saturadas por varios minutos, muy pocos hinchas pueden entrar y los demás con sus boletos originales se están perdiendo del espectáculo, se pregunta ¿Qué ocurrió? Pues sucedió una denegación de servicio de acceso al estadio de futbol.

Ahora en un sitio web es como una de las puertas de acceso del estadio y solo se puede dar servicio a limitados números de personas al mismo tiempo, es por esto que, si se recibe demasiadas solicitudes de lo que el sitio web puede manejar, se logrará un bloqueo en el sitio y por esto nada entra ni sale y posteriormente se cae el servicio.

Desde la perspectiva del IOT y su incremento de los dispositivos conectados, se analiza que en un corto y mediano plazo se incrementará la cantidad de cosas conectadas en estos ambientes por el cual los ataques también aumentarán es por esto que la seguridad de estos dispositivos se comienza a tomar una gran consideración, es por esto que la infraestructura para realizar estos ataques DDoS no necesariamente debe ser grande y que puede poner en serios aprietos a las organizaciones o ciudades que estén comenzando a utilizar esta tecnología, se han propuesto distintas soluciones como el análisis de tráfico para que pueda ser procesado mientras estos ataques de producen.

Pero una propuesta más clara seria adecuar el mercado de estos dispositivos y desarrollar políticas de seguridad como también estándares que ayuden a implementar medidas para poder detectar, controlar y mitigar estos ataques de los ambientes aplicados.

2.1.2 Pyshark y Tshark

Durante mucho tiempo han existido diferentes herramientas para el análisis de redes, ahora en el ambiente aplicado de este proyecto que será en Linux, existen varios como por ejemplo Wireshark, tcpdump, nload, iftop, iptraf, tcptrack y así entre otros, pero también existen otras herramientas que las podemos adaptar a lo que realmente se necesita, ya que muchas de las herramientas mencionadas anteriormente no brindan lo necesario como también la velocidad y carga de cada análisis se puede ver afectada por lo cual se necesita un buen desempeño por lo que esto motiva a utilizar plugins o librerías que permitan la solución a estos problemas.

En Python existen bastantes librerías para el procesamiento y análisis de redes, para esto la programación en bajo nivel, librerías de sockets es lo principal, además, para poder monitorear tanto los puertos de red como los flujos de paquetes que es lo que nos interesa en este proyecto, se va utilizar lo que es Pyshark como Tshark, estas librerías permitirán la supervisión de paquetes y su tráfico.

Para este proyecto se utilizará la librería Pyshark para poder revisar y supervisar los paquetes que llegan a una interfaz de red específica, sin embargo, como se conoce la librería podría realizar mucho más, pero la documentación existente hace que sea innecesariamente difícil por lo que podría existir algunos problemas al momento de revisar la obtención de estos paquetes con su respectivo análisis.

Pyshark es un wrapper(son embalajes para mejorar la compatibilidad de diferentes estructuras de software o presentar algo a nivel visual) de Python para Tshark, ya que tiene la capacidad de exportar datos XML para su análisis, esta librería permite realizar un análisis desde el archivo de captura o una captura en tiempo real porque utiliza los disectores de Wireshark.

Por su parte Tshark es una analizador de protocolos de red que permite capturar información de los paquetes que ingresan en una red activa o también puede leer paquetes guardados en archivos que se han capturado previamente, se puede imprimir de manera decodificada de los paquetes obtenidos tanto con una salida estándar como escribiendo lo necesario para observar del análisis, el formato de captura de esta librería es pcapng que es un formato que se utiliza en Wireshark y en otras herramientas.

Este analizador funciona similar a tcpdump ya que utiliza la librería pcap para la captura del tráfico de la primera interfaz de red que esté disponible y también demuestra el resumen de lo capturado en una salida estándar, esta forma de presentar la información contiene

campos específicos por el archivo de preferencias que son similares a campos que se muestran en la herramienta de análisis de paquetes de Wireshark, mientras está capturando los paquetes se escribe la información que recibe, es por esto que la librería asociada que es pcap permite que existan distintos filtros para la obtención de información necesaria de los paquetes que ingresan a la red activada.

Lo interesante de Tshark es que puede detectar, leer y escribir los mismos archivos de captura que son compatibles con Wireshark, sin embargo, como es una librería externa no tiene todo lo necesario para un análisis más avanzado al igual Pyshark.

2.1.3 Wireshark y Api Rest

Como se ha explicado con anterioridad acerca de la librerías de Python que se utilizarán para el desarrollo del sniffer para la captación de los paquetes, es importante entender de donde sale estas librerías es por esto que estas librerías son desarrolladas en base a Wireshark, esta herramienta nos permite analizar protocolos de red y es ampliamente utilizado en el mundo, también ayuda a observar lo que sucede a la red a niveles microscópicos, esta herramienta es utilizada en instituciones educativas para su enseñanza como también en empresas y agencias gubernamentales, que han podido ayudar en muchos de los análisis que se realizan para conocer lo que sucede en el ambiente de red en donde se encuentran.

Wireshark como se ha explicado antes nos va a permitir entender un poco más sobre como capta estos paquetes cuando realiza el análisis de los mismos, cuando existe el tráfico dentro de una red, esta herramienta es excelente para el estudio de las comunicaciones y las resoluciones que se pueden dar de manera didáctica sobre los problemas de la red que pueden existir, además que Wireshark implementa muchos filtros que nos ayudan a definir los criterios de búsqueda para los distintos protocolos que son soportados actualmente, por su parte cuenta con una interfaz intuitiva como sencilla que nos permite separa o segmentar por capas todos los paquetes que son capturados, siendo así posible revisar la información dentro de cada paquete que ingresa a nuestra red.

La importancia de esta herramienta es porque entiende cada estructura de cada protocolo y nos permite visualizar cada campo de las cabeceras y capaz de cada paquete que es captado, dando mayores ventajes como posibilidades para quienes administran las redes puedan abordar sus tareas acerca del análisis del tráfico en la red.

Explicando sobre las API REST se puede entender proporcionan una manera más flexible como también ligera para la integración de aplicaciones, también se le conoce como una interfaz de programación que tiene ciertas reglas que permiten definir la forma en que los dispositivos y las aplicaciones pueden comunicarse. Se debe cumplir con los principios de diseño de este estilo Rest o conocido como trasferencias de estados que son los métodos HTTP que dan la información del estado entra la comunicación del dispositivo con el servidor con el cual está conectado.

Existen principios para la creación de estas API REST, que están desde los más básico hasta lo más complejo dependiendo de lo que se necesite, en este caso será un diseño que permitirá a una aplicación acceder en manera de servicio, lo cual se denomina cliente y donde se va obtener la información o su contenido se lo conocerá como servidor.

Como recordamos la comunicación de las API REST se realiza por solicitudes HTTP que al ejecutar ayudan activar funciones de las bases de datos estándar, las más conocidas como el crear, leer, actualizar y borrar los registros o información ingresada que comúnmente se los conoce como CRUD por sus siglas en inglés, en las llamadas de API es permitido utilizar cada uno de los métodos HTTP, normalmente los estados que se reciben o la información que se presenta podrá ser presentada en cualquier formato, pero los más conocidos son JSON,HTLM XML, entre otros. El formato más popular por el cual es representado la información de las APIS es el JSON porque es más factible como legible para las máquinas como para los usuarios y es independiente de cualquier lenguaje a utilizar.

Tanto como los parámetros y las cabeceras de las solicitudes son muy importantes en las llamadas de cada API REST ya que tienen información de cada identificador, estos pueden ser metadatos, cookies, autorizaciones, etc. Estos siempre se va utilizar junto a todos los códigos de estados que tiene HTTP, estas operaciones y mantenimientos de datos que son realizadas mediante estos métodos que normalmente son utilizadas en gestores relacionales de bases de datos.

Método HTTP	Operación
GET	Leer todas las entidades dentro de la colección
PUT	Actualización mútiple y/o masiva
DELETE	Borrar la colección y todas sus entidades
POST	Crear una nueva entidad dentro de la colección

Figura 7 CRUD para una colección en una API REST

Método HTTP	Operación
GET	Leer los datos de una entidad en concreto
PUT	Actualizar una entidad existente o crearla si no existe
DELETE	Borrar una entidad en concreto
POST	Añadir información a una entidad ya existente

Figura 8 CRUD para entidades de una API REST

2.2 Planear

En esta fase se expondrá la manera en la que el ambiente de desarrollo se configurará para la ejecución de proyecto, como también la forma en que se van a utilizar las librerías o plugins para el análisis de tráfico de los paquetes LoRaWAN y la arquitectura que se utilizar para el sniffer como para el API REST, teniendo en cuenta que las limitaciones de las librerías pueden tener al momento del desarrollo, ya que como se ha explicado antes no existe mucha documentación respecto a las librerías a utilizar pero se evaluará la manera en la que se podrá utilizar y que en un futuro se pueda realizar el API REST de manera segura con el análisis de los paquetes dentro de la redes LoRaWAN.

2.2.1 Configuración de ambiente de desarrollo

Para la implementación de la infraestructura que se utilizará mediante un ambiente Linux ya que es más práctico para la instalación de las librerías requeridas para el análisis del tráfico de red como también las diferentes interfaces a analizar, sin embargo, se tomará en cuenta las distintas formas del accionar de las librerías, por parte del lenguaje de programación para el desarrollo tanto del sniffer como del prototipo de API REST será Python que es más factible para este tipo de desarrollos, para el framework se utilizara Django que nos permite la creación del prototipo API REST de una manera más fácil utilizando los estados y métodos HTTP que se necesitan para las recibir como enviar la información desde la app hacia el servidor que será una base de datos como PostgreSQL o MySQL Server donde se guardará la información de los paquetes captados a través del sniffer que es la idea de este API REST, mediante el avance del desarrollo del sniffer que es lo primordial se tomará en cuenta el funcionamiento de las librerías a utilizar.

2.2.2 Utilización de librerías y pruebas

Como se ha indicado anteriormente las librerías a utilizar serán Pyshark y Tshark que son derivadas de Wireshark, sin embargo, la falta de documentación hace que sea complejo el conocer en su totalidad el funcionamiento de las librerías, conforme se vaya avanzando en el desarrollo del sniffer para el análisis del tráfico de datos se podrá analizar el funcionamiento de los mismos.

Tanto en el Pyshark permite conocer o capturar los paquetes que ingresan dentro de la interfaz de red a elegir siendo capaz de filtrar para este proyecto por puertos para que sea mes específico en la búsqueda de estos paquetes LoRaWAN, conociendo que tiene un distinto tipo de parser, se debe conocer su realmente estas librerías serán capaces de analizar estos paquetes que ingresan para conocer su información y observar de donde están ingresando como de su respectiva dirección, al igual que el Tshark nos ayudará a que su visualización sea mejor al momento de imprimir los resultados, ya que Tshark es una librería que junto Pyshark permitirá que sea más fácil la manipulación de estos datos tanto para obtenerlos como para mostrarlos.

Las pruebas que se han querido plantear es el poder enviar ataques simulados o enviar paquetes LoRaWAN desde un puerto para que el sniffer pueda captar estos paquetes, de acuerdo a la interfaz escogida en donde se enviará los paquetes, se tomara en cuenta que la simulación de los ataques se enviara por medio de los puertos UDP porque la simulación del envió de los paquetes solo puede realizarlo a través de puertos UDP, luego también gracias a la misma librería se podrá obtener los datos o información relacionados a los paquetes enviados a través de la red hacia el puerto, que el sniffer deberá captar los paquetes enviados para luego revisar su contenido y observar los más importante de cada uno, y reconocer lo que está ingresando a través de la red.

2.2.3 Diseño de arquitectura

Para la arquitectura se utilizará mencionado antes el ambiente Linux por su manera más factible de instalar las librerías solicitadas para el análisis de redes, se optó para el desarrollo del sniffer el lenguaje Python por su manejo de los datos y más que todo porque la librería hasta donde está realizada se encuentra en el mismo lenguaje, también se optó por utilizar entro dos frameworks para el API REST como Flask y Django que son frameworks basados en Python para la realización de estos tipos de API REST por su rapidez al momento de su creación y muestra de datos para el usuario, sin olvidar que cada uno de ellos maneja los métodos HTTP que serán utilizados en el proyecto tanto como

GET/POST/DELETE que es que se necesita para poder guardar los datos después de análisis, los otros métodos como PUT no se utilizarán ya que no hay un sentido en actualizar los datos de los análisis si lo que se necesita es el realizar un análisis nuevo cada vez que se requiera y sea guardado como dato nuevo, para el almacenamiento de esto de estos datos se utilizará MySQL Server o PostgreSQL, tomando en cuenta que los datos tanto al guardar en la base de datos como la presentación de los mismos será en formato JSON para que tenga una mejor visibilidad hacia el usuario sobre el contenido de los paquetes guardados.

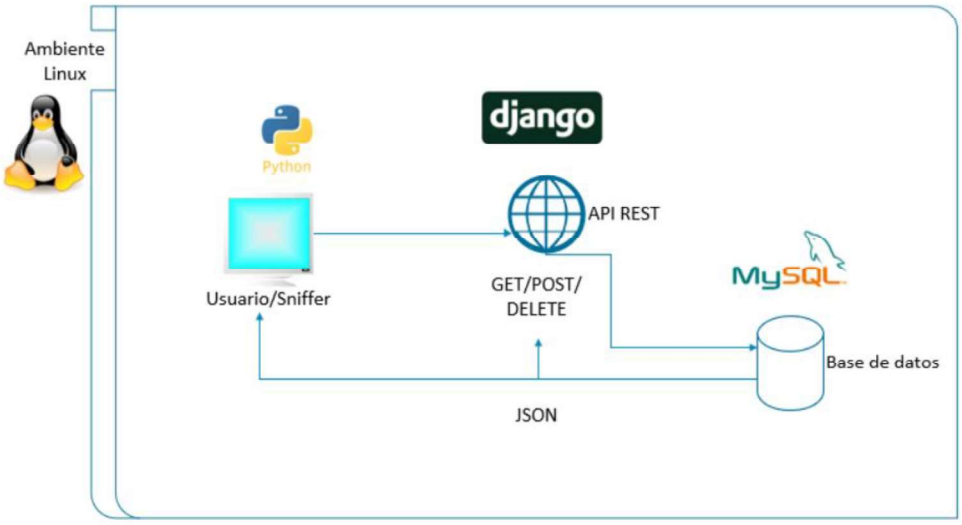


Figura 9 Diseño de Arquitectura

2.3 Ejecutar

El ambiente de desarrollo ya levantado durante toda la fase de planeación como se indicó antes y junto con su respectiva arquitectura levantada antes de su integración, ya que lo primero que se va a realizar es el sniffer para el análisis del tráfico de los paquetes LoRaWAN que van a ingresar a través de la red, conforme se va desarrollando se podrá observar la manera de que las librerías puedan dar opciones del análisis de esos paquetes, por otra parte de la misma manera se eligió en primer lugar realizar el API REST en Flask pero se observó que tuvo algunos problemas con la integración de las librerías y también con los formatos de visualización de los paquetes de ejemplo, se optó por cambiar el framework de desarrollo utilizando así Django, que permitió la manipulación tanto de los datos como de los métodos HTTP para el almacenamiento de los datos de los paquetes ingresados manualmente para conocer el funcionamiento de estos métodos, también se optó por escoger la base de datos Mysqlserver por su mejor adaptación a este framework de desarrollo y por su instalación directamente al API REST, se debe recalcar que estos ajustes se realiza a nivel de implementación tanto para el API REST como para el sniffer, recordando que la falta de documentación de la librerías utilizar se puede optar por más cambios en su implementación.

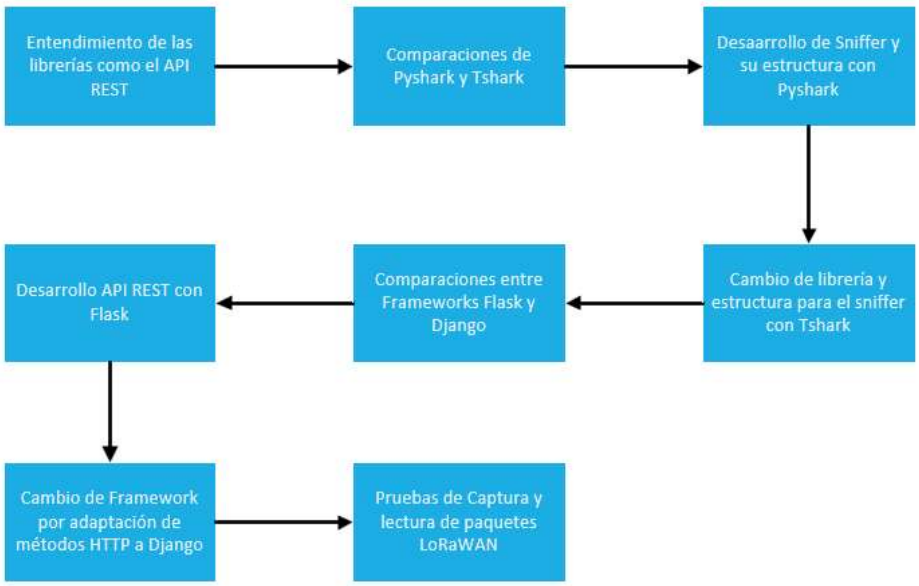


Figura 10 Adaptación del Sniffer y API REST

Mediante las adaptaciones que se fueron realizando por distintas limitaciones encontradas tanto en los frameworks como en las librerías del análisis de datos, se optó por estos cambios para encontrar soluciones, en el caso de la adaptación del sniffer para la captación de paquetes LoRaWAN existió un problema con la librería Pyshark ya que capturaba los paquetes en forma general, sin embargo, no capturaba los paquetes Lora porque se pudo comprar a través del Wireshark que mediante esta herramienta se capturaba sin problema pero en el sniffer desarrollado por la librería no capturaba, se realizó distintos cambios y filtros para esta captura de los paquetes como el escaneo de puertos específicos, interfaces y distintas capas, sin embargo no se pudo capturar los paquetes LoRaWAN, esta fue la razón de pasar a prueba la otra librería de Tshark para comprobar que se realice el análisis de trafico de paquetes y comprobar si puede leer estos paquetes para continuar, la manera en que trabaja Tshark es similar a Wireshark solo que es todo por consola, a diferencia de Pyshark no realiza la lectura de la data capturada, sino que es solo generada en formato hexadecimal y es necesario la transformación respectiva de esta parte de los datos de los paquetes que van entrando a través de la red para observar el contenido o la información de estos paquetes y conocer si está capturando de la manera correcta.

Se está llevando a cabo el desarrollo en Python de script que llame a este comando para el análisis respectivo por lo que la estructura cambio, igual se debe recordar que esto fue a nivel de código para las mejoras en el sniffer de los paquetes LoRaWAN, se implementará la manera que pueda pasar del formato hexadecimal a ASCII para comprobar el contenido de los paquetes y de esta manera en un futuro implementarlo junto al API REST.

2.4 Revisar

Para poder conocer si realmente la validación de los cambios de la librería Pyshark a Tshark se propone realizar las siguientes pruebas conformo a los que se venía trabajando con Pyshark que es la captura de los paquetes LoRaWAN y su contenido, luego de ello se implementara el cambio del formato hexadecimal a ASCII para la lectura de la data de cada uno de los paquetes que ingresan por la red.

Prueba	**Resultado esperado**
Captura de paquetes en general	Paquetes capturados en sus interfaces
Captura de paquetes filtrados por puertos	Paquetes capturados en general
Captura de paquetes LoRaWAN	Captura de la data del paquete
Data en formato hexadecimal	Archivo en formato csv
Transformación de archivo hexadecimal	Observación de los datos del paquete LoRaWAN
Visualización de datos LoRaWAN en API REST	Observación de los datos importantes del paquete LoRaWAN

Cada una de estas pruebas permitirán comprobar y desarrollar el sniffer que pueda capturar el tráfico de paquetes LoRaWAN y su respectivo contenido para que luego de ello al ser importado en un archivo con formato CSV permita separa la data que es la parte principal que queremos obtener de cada paquete que entra a la red, ya que tendrá información importante de los paquetes LoRaWAN y así poder reconocerlos de la mejor manera, sin embargo, serán capturados en especial el campo de data en formato hexadecimal por el cual se debe transformar a formato ASCII que nos ayuda a entender el contenido de estos paquetes y de este campo, para que en un futuro se pueda ir agregando estos resultados al API REST con los métodos HTTP y puedan obtener una base de datos con estos ejemplos de paquetes.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Durante esta sección del documento se mostrará los resultados, conclusiones y recomendaciones que se dará después de las pruebas realizadas y que fueron planteadas en la sección 2.4 (fase de revisión), manteniendo en mente el objetivo de validar el sniffer para que pueda capturar los paqueres LoRaWAN, respetando los requerimientos necesarios que se expresaron en sección 2.1 (fase de identificación), entendiendo la documentación necesaria para el desarrollo del sniffer.

Se realizo las pruebas propuestas en la sección 2.4 dando como resultado con éxito cada una de ellas sobre el prototipo funcional del sniffer LoRaWAN y sus paquetes, logrando simular los ataques y capturándolos, permitiendo descifrar la información requerida del paquete como también decodificando los paquetes LoRaWAN para conocer la información que se va transportando durante el tráfico de red hacia el dispositivo inteligente.

3.1 Resultados

3.1.1 Preparación y comprensión del ambiente de captura de los paquetes LoRaWAN

Revisado la documentación respectiva durante el desarrollo del sniffer, se conoce que primero la simulación del ataque de los paquetes LoRaWAN era necesario conocer la estructura del mismo para así saber lo importante que se debe tomar en cuenta de la información del paquete al ser capturado por el sniffer, es por ello que se toma en cuenta la estructura del paquete que es enviado a través de la red, como se muestra en la figura 11.

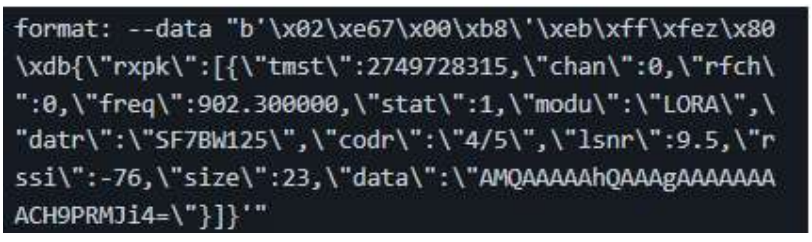


Figura 11 Formato o Estructura del paquete LoRaWAN

Conociendo el formato del paquete que se debe recibir a través de la red, procedemos a abrir un puerto específico para la simulación del envío de estos paquetes, para esto utilizamos la herramienta Netcat que nos ayuda abrir puertos tanto TCP como UDP, que para este caso necesitamos abrir un puerto UDP ya que la herramienta de simulación del envío de paquete lo realiza a través de los puertos UDP, y se realizó a través de línea de comandos por la facilidad de estar utilizando un ambiente Linux como se indica en la figura 12.

Figura 12 Herramienta Netcat para abrir el puerto UDP

Se comprobó que el puerto se haya abierto para que puedan llegar los paquetes en este caso el puerto 1700 a través de la misma herramienta Netcat como se indica en la figura 13.

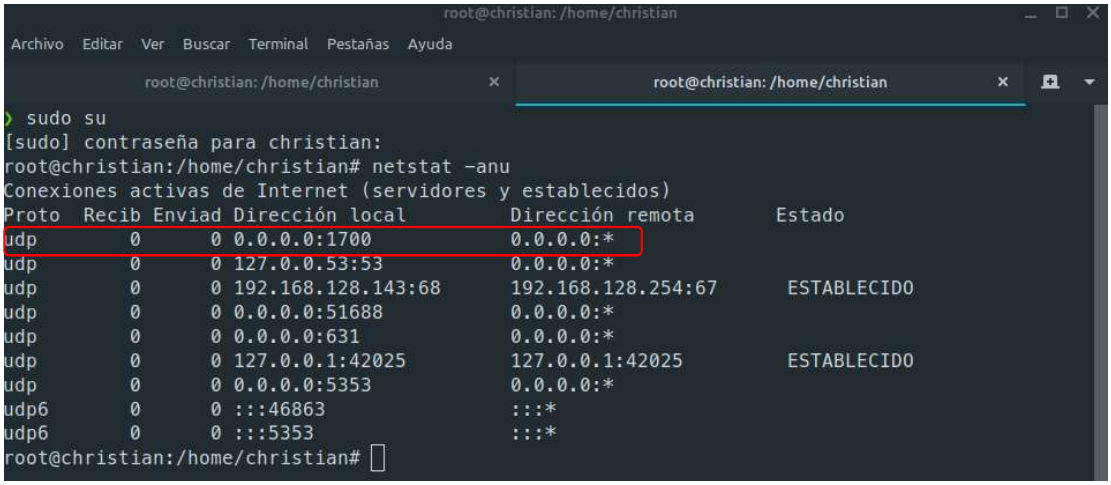


Figura 13 Comprobación del puerto abierto

Luego con la herramienta de simulación para el envío de paquetes ya configurado para su función estará listo para conocer el funcionamiento del sniffer que captura los

paquetes LoRaWAN, recordando que el envío del paquete debe mantener el formato que se explico en la figura 11, y de esta forma se envía los paquetes a través de la red hacia el puerto abierto como se indica en la figura 14.

Figura 14 Forma de envío del paquete LoRaWAN

3.1.2 Captura de los paquetes LoRaWAN

Durante el envío de los paquetes de prueba se debe mantener en ejecución el script del sniffer desarrollado con el nombre de tsharklorawan.py que en primer lugar se nos pedirá la interfaz a revisar, el tipo de puerto que se va a analizar y el numero de paquetes a capturar a través del sniffer como se muestra en la figura 15.

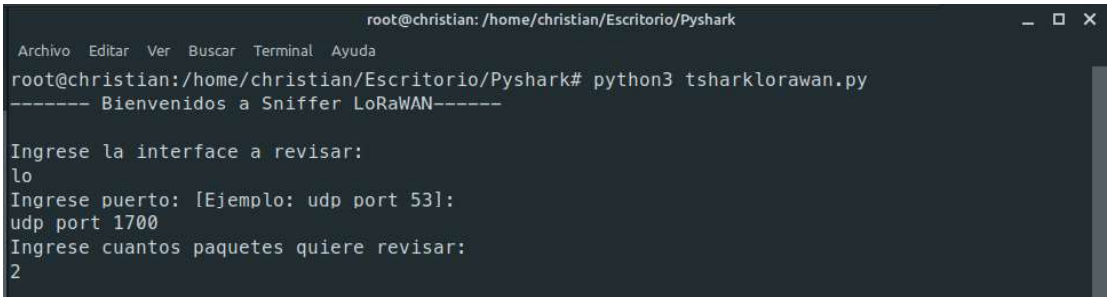


Figura 15 Ingreso de datos a revisar en el sniffer

A continuación, se procede a enviar los paquetes hacia el puerto 1700 que se abrió anteriormente y de la manera como se indica en la figura 14, para esta prueba se realizó el envío de dos paquetes distintos para poder conocer la diferencia de información capturada por el sniffer, como se indica en la figura 16.

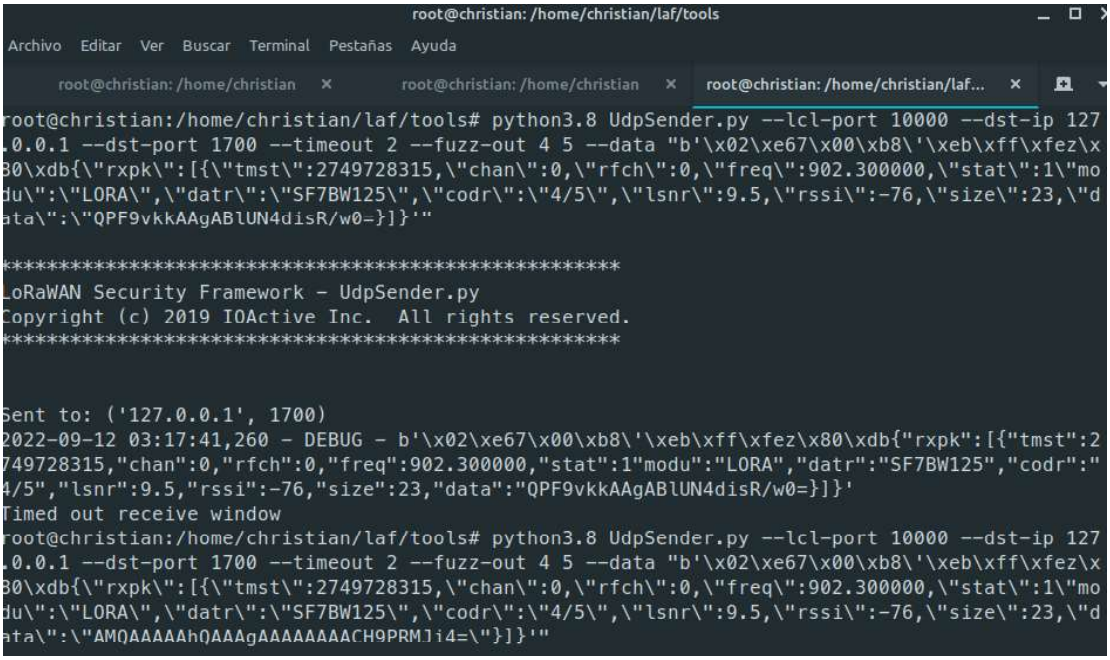


Figura 16 Confirmación de envío del paquete LoRaWAN

Luego del envío, el sniffer indicará nuevamente el número de paquetes que capturó en la red con su respectiva localización en donde se realizó el análisis, es decir la interfaz analizada como se indica en la figura 17.

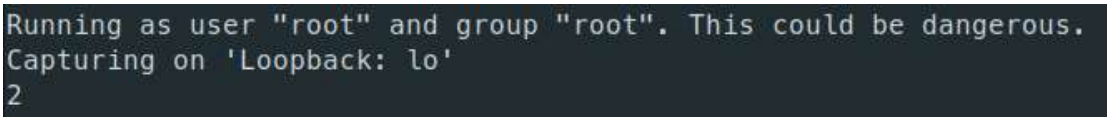


Figura 17 Número de paquetes capturados

3.1.3 Decodificador hexadecimal y de LoRaWAN

Al capturar los paquetes LoRaWAN a través del sniffer, recibimos la información en formato hexadecimal por el cual debemos decodificarlo al formato ASCII para recibir la información completa, dentro del desarrollo del sniffer se incluyó el decodificador hexadecimal para la transformación directa de los paquetes capturados, en la figura 18 se indicará la data hexadecimal que captura el sniffer como también la información decodificada.

```
-------- Data LoRaWAN dispositivo: 1 -------

Data Packet Hexadecimal: 02e63700b827ebfffe7a80db7b227278706b223a5b7b22746d7374223a32373439373
2383331352c226368616e223a302c2272666368223a302c2266726571223a3930322e3330303030302c22737461742
23a31226d6f6475223a224c4f5241222c2264617472223a225346374257313235222c22636f6472223a22342f35222
c226c736e72223a392e352c2272737369223a2d37362c2273697a65223a32332c2264617461223a2251504639766b6
b41416741426c554e34646973522f77303d7d5d7d

Data Translate Hexadecimal to ASCII: æ7¸'ëÿþzÛ{"rxpk":[{"tmst":2749728315,"chan":0,"rfch":0,"f
req":902.300000,"stat":1"modu":"LORA","datr":"SF7BW125","codr":"4/5","lsnr":9.5,"rssi":-76,"si
ze":23,"data":"QPF9vkkAAgABlUN4disR/w0=}]}
```

Figura 18 Decodificador Hexadecimal a ASCII

Ahora ya capturado nuestro paquete LoRaWAN necesitamos extraer lo más importante del paquete, en este caso es la “data” ya que ahí contine la información de las cabeceras, dirección del dispositivo(FHDR), entre otras. Al igual que el decodificador anterior, también se incluyo el decodificador de formato LoRaWAN que viene en base64 a directamente a la lectura de la trama que contiene esa información para conocer la información directa del paquete LoRaWAN que se envió, esto realiza el sniffer con cada paquete que llega a través de la red siendo de utilizada para poder conocer que tipo de paquetes ingresan y conocer si son seguros o no, facilitando que en un futuro se pueda analizar si a través de estos paquetes que ingresan son dañinos o puedan ocasionar una denegación de servicios distribuido y también se puedan ingresar a una API REST la información necesaria para futuras consultas, en el gráfico 19 se indicará la decodificación del paquete LoRaWAN.

```
decoding from Base64:  AMQAAAAAhckAAgAAAAAAAACH9PR5Ji4=
Decoded packet
--------------
          Message Type = Join Request
            PHYPayload = 00C40000000085C900020000000000000087F4F479262E

          ( PHYPayload = MHDR[1] | MACPayload[..] | MIC[4] )
                 MHDR = 00
            MACPayload = C40000000085C900020000000000000087F4
                   MIC = F479262E

          ( MACPayload = AppEUI[8] | DevEUI[8] | DevNonce[2] )
               AppEUI = 00C98500000000C4
               DevEUI = 0000000000000002
             DevNonce = F487
```

Figura 19 Decodificador de paquetes LoRaWAN

Se describe la manera en que el paquete capturado pasa por diferentes fases hasta ser decodificado en su totalidad para obtener la información necesaria e importante del paquete LoRaWAN, como se indica en la figura 20.

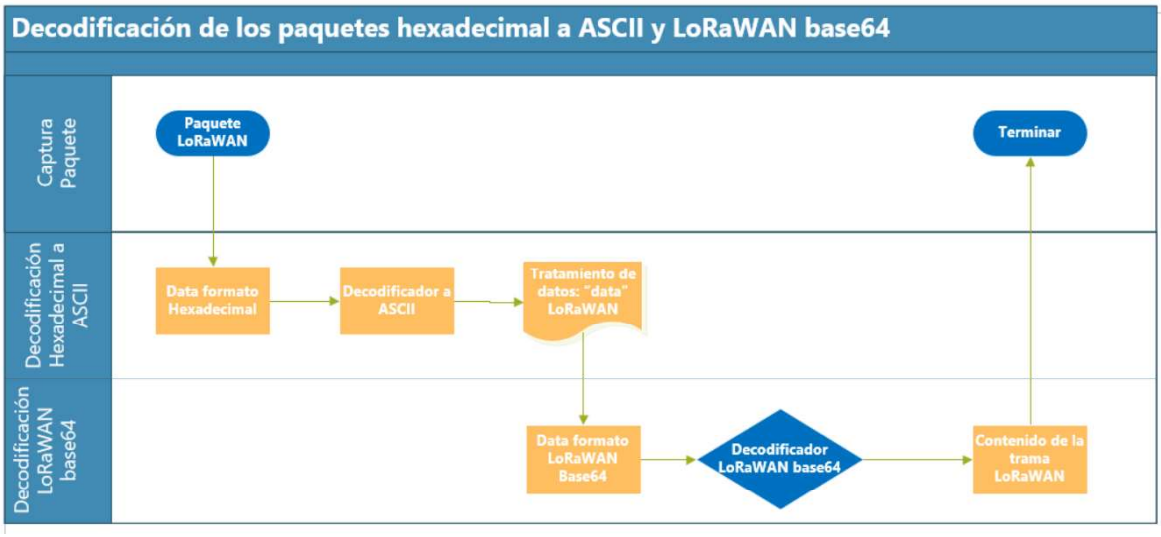


Figura 20 Fases de decodificación del paquete LoRaWAN

3.1.4 Api Rest y datos del paquete LoRaWAN

Al final se realizó el almacenamiento de los datos del paquete LoRaWAN en el API REST, con sus respectivos métodos HTTP para conocer los datos del paquete, tener cuenta que se debe mejorar en la presentación de los datos ya que dentro de mismo framework Django se debe configurar ciertos parámetros para la visualización de los datos, como se indica en la figura 21 y 22.

Figura 21 Muestra general del API REST

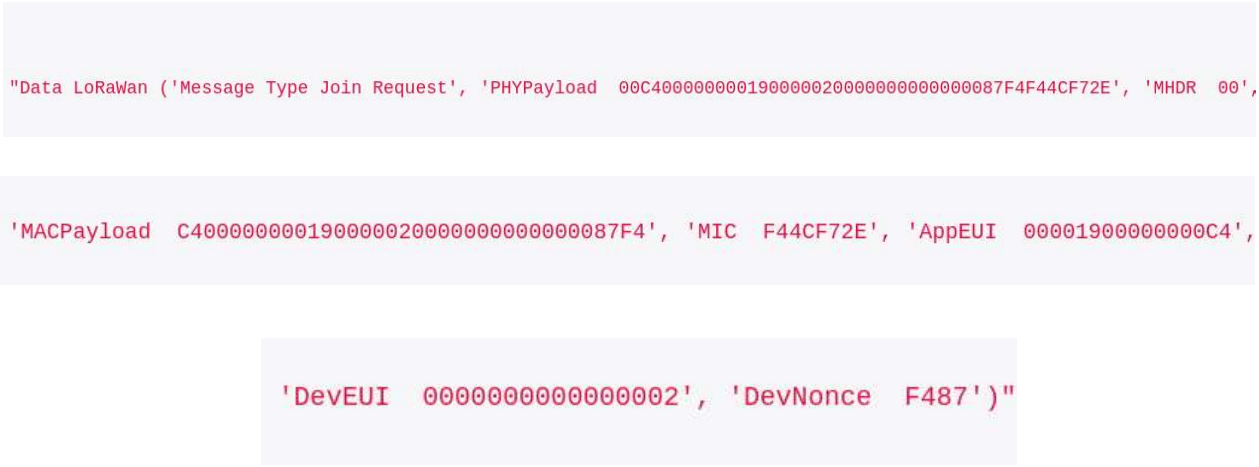


Figura 22 Muestra de Datos LoRaWAN en API REST

3.2 Conclusiones

- Se comprendió la importancia del análisis del tráfico en la red a través del Wireshark para la implementación del sniffer y sus ataques de denegación de servicios(DDoS) distribuidos, y su manejo a través de la herramienta Wireshark, como se indica en la sección 2.1.1.
- Se logró la construcción del sniffer basado en Wireshark que permite analizar las tramas de los paquetes LoRaWAN y decodificar su respectiva información para conocer el contenido de los paquetes que ingresan a través de la red, como se indica en la sección 3.3.3 del documento.
- A pesar de las limitaciones al momento de integrar la librería Tshark para el API REST, ya que el framework Flask no permitió la instalación de esta librería en el ambiente virtual que maneja, por el cual en el framework Django permitió su instalación para la ejecución del análisis y guardado de datos en el API REST, como se indica en la sección 2.4.
- Se ejecutaron con éxito las pruebas planteadas en la sección 2.4, sobre la forma de capturar los paquetes, la decodificación de hexadecimal a ASCII como también la decodificación de la forma base64 de LoRaWAN, además de la manera de exportación de los datos que se indicará en el anexo II.

- Se comprendió la decodificación y el contenido que tiene un paquete LoRaWAN, es decir, la información de sus cabeceras importantes para poder reconocer de donde se envían los paquetes como también información del dispositivo, como se indica en la sección 3.3.3.

3.3 Recomendaciones

- Realizar la implementación del sniffer para un ambiente distinto, ya que se manejo dentro del ambiente Linux que permitía el análisis directamente desde sus interfaces que uno podía elegir, para ello una nueva implementación para otro ambiente seria ideal ya que muchos usuarios utilizan distintos ambientes de sistemas operativos.
- Investigar más sobre la librería Pyshark, ya que contiene muchas aplicaciones interesantes e información que se puede obtener a través de su análisis de tramas. Para ello es necesario distintas pruebas de captura de paquetes y la manera en la que se puede editar, ya que su información es escasa hasta el momento por ser una librería nueva o en desarrollo.
- Investigar sobre la implementación o instalación de las librerías de análisis de tráfico de paquetes en la red para los frameworks en sus ambientes virtuales, y la utilización de los mismos para que se pueda facilitar el desarrollo de un API REST amigable para el usuario.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Iqbal, Haroon & Naaz, Sameena. (2019). Wireshark as a Tool for Detection of Various LAN Attacks. International Journal of Computer Sciences and Engineering. 7. 833-837. 10.26438/ijcse/v7i5.833837.

[2] J. E. Varghese y B. Muniyal, "A pilot study in software-defined networking using wireshark for analyzing network parameters to detect DDoS attacks", en Information and Communication Technology for Competitive Strategies (ICTCS 2020), Singapore: Springer Singapore, 2021, pp. 475–487.

[3] RAI, Mahendra Kumar; HALDKAR, Gyanendra. Mitigation of Intruders and TCP bad Connection Detection in WAN Environment using Wireshark. 2015.

[4] E. Aguirre Hernández, A. E. Guerrero Zenil, A. A. Hernández Medellín, C. Hernández Lara, y G. Hernández Hernánde, "Análisis de tráfico TCP utilizando la herramienta Wireshark", Cienc. Huasteca Bol. Cient. Esc. Super. Huejutla, vol. 5, núm. 10, 2017.

[5] "¿Qué es una API de REST?", Redhat.com. [En línea]. Disponible en: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Consultado: 08-ene-2022].

[6] "rest-apis", Ibm.com. [En línea]. Disponible en: https://www.ibm.com/es-es/cloud/learn/rest-apis.

[7] "What is LoRaWAN® specification", LoRa Alliance®, 21-oct-2020. [En línea]. Disponible en: https://lora-alliance.org/about-lorawan/. [Consultado: 12-sep-2022].

[8] M. Fraile Izquierdo y J. Vales Alonso (advisor), "Ataques DDoS en entornos IoT", 2019.

[9] "Tshark(1)", Wireshark.org. [En línea]. Disponible en: https://www.wireshark.org/docs/man-pages/tshark.html.

[10] "PyShark", Github.io. [En línea]. Disponible en: http://kiminewt.github.io/pyshark/.

[11] "rest-apis", Ibm.com, 06-abr-2021. [En línea]. Disponible en: https://www.ibm.com/es-es/cloud/learn/rest-apis.

[12] "Wireshark · Go Deep", Wireshark.org. [En línea]. Disponible en: https://www.wireshark.org/.

[13] R. Minerva, A. Biru, and D. Rotondi, “Define IoT - IEEE Internet of Things,” Ieee, 2015. https://iot.ieee.org/definition.html.

[14] N. Kong, P. Jung-Soo, N. Crespi, G. Lee, and I. Chong, “The Internet of Things - Concept and Problem Statement.”. [Online]. Available: https://tools.ietf.org/html/draft-lee-iot-problem-statement-00.

[15] S.-Y. Wang and T.-Y. Chen, “Increasing LoRaWAN Application-Layer Message Delivery Success Rates,” in 2018 {IEEE} {Symposium} on {Computers} and {Communications} ({ISCC}), Jun. 2018, pp. 148–153, doi: 10.1109/ISCC.2018.8538457.

[16] “The Things Network | We are building a global open free crowdsourced long range low power IoT data network.” https://www.thethingsnetwork.org/docs/

[17] LoRa Alliance, “Learn More about Smart Industries,” 2020. http://pages.loraalliance.org.pages.services/Industry-Vertical-Market-1/learnmore.html?ts=1601964052073 (accessed Mar. 30, 2021).

[18] LoRa Alliance, “Learn More about Smart Logistics,” 2020. http://pages.loraalliance.org.pages.services/Logistics-Vertical-Market/learnmore.html?ts=1601964418182

[19] LoRa Alliance, LoRaWAN® Specification v1.0 - LoRa Alliance®. 2015.

20] LoRa Alliance, “LoRaWAN® 1.0.4 Specification Package - LoRa Alliance®,” 2020. https://lora-alliance.org/resource_hub/lorawan-104-specification-package/

[21] Arduino, “What is Arduino? | Arduino,” 2018. https://www.arduino.cc/en/Guide/Introduction

[22] Red Hat, “What is an Arduino? | Opensource.com.”

[23] L. Alliance, “LoRaWAN® Specification v1.0.3 LoRa AllianceTM.” 2018, Accessed: Nov. 10, 2019. [Online]. Available: https://lora-alliance.org/resource-hub/lorawanrspecification-v103.

[24] I. Butun, N. Pereira, and M. Gidlund, “Security Risk Analysis of LoRaWAN and Future Directions,” Futur. Internet, vol. 11, no. 1, p. 3, Dec. 2018, doi: 10.3390/fi11010003.

5 ANEXOS

ANEXO I. Envío de dos paquetes distintos LoRaWAN

```
*****************************************************
LoRaWAN Security Framework - UdpSender.py
Copyright (c) 2019 IOActive Inc.  All rights reserved.
*****************************************************


Sent to: ('127.0.0.1', 1700)
2022-09-12 03:17:41,260 - DEBUG - b'\x02\xe67\x00\xb8\'\xeb\xff\xfez\x80\xdb{"rxpk":[{"tmst":2
749728315,"chan":0,"rfch":0,"freq":902.300000,"stat":1"modu":"LORA","datr":"SF7BW125","codr":"
4/5","lsnr":9.5,"rssi":-76,"size":23,"data":"QPF9vkkAAgABlUN4disR/w0=}]}'
Timed out receive window
root@christian:/home/christian/laf/tools# python3.8 UdpSender.py --lcl-port 10000 --dst-ip 127
.0.0.1 --dst-port 1700 --timeout 2 --fuzz-out 4 5 --data "b'\x02\xe67\x00\xb8\'\xeb\xff\xfez\x
80\xdb{\"rxpk\":[{\"tmst\":2749728315,\"chan\":0,\"rfch\":0,\"freq\":902.300000,\"stat\":1\"mo
du\":\"LORA\",\"datr\":\"SF7BW125\",\"codr\":\"4/5\",\"lsnr\":9.5,\"rssi\":-76,\"size\":23,\"d
ata\":\"AMQAAAAAhQAAAgAAAAAAAACH9PRMJi4=\"}]}'"
```

```
*****************************************************
LoRaWAN Security Framework - UdpSender.py
Copyright (c) 2019 IOActive Inc.  All rights reserved.
*****************************************************


Sent to: ('127.0.0.1', 1700)
Fuzzing MIC position 21 out of 23
2022-09-12 03:17:57.170479:
Replacing byte offset=20, old value=76, new=121
2022-09-12 03:17:57.170587:
Replacing byte offset=7, old value=0, new=201
Old FCnt 133, New FCnt 51589
2022-09-12 03:17:57,172 - DEBUG - b'\x02\xe67\x00\xb8\'\xeb\xff\xfez\x80\xdb{"rxpk":[{"tmst":2
749728315,"chan":0,"rfch":0,"freq":902.300000,"stat":1"modu":"LORA","datr":"SF7BW125","codr":"
4/5","lsnr":9.5,"rssi":-76,"size":23,"data":"AMQAAAAAhckAAgAAAAAAAACH9PR5Ji4="}]}'
Parsed data: {"mhdr":{"mType":"JoinRequest","major":"LoRaWANR1"},"macPayload":{"joinEUI":"00c9
8500000000c4","devEUI":"0000000000000002","devNonce":62599},"mic":"f479262e"}
Timed out receive window
root@christian:/home/christian/laf/tools#
```

ANEXO II. Formato de exportación CSV. De los paquetes LoRaWAN

	A	B	C	D	E	F	G	H
1	frame.number	frame.time	eth.src	eth.dst	ip.src	ip.dst	ip.proto	data
2	1	Sep 12, 2022 03:17:41.259965763 -0	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	17	02e63700b827ebfffe7a80db7b227278706b223a5b7b22746d7374223a3237343937323833
3	2	Sep 12, 2022 03:17:57.170708965 -0	00:00:00:00:00:00	00:00:00:00:00:00	127.0.0.1	127.0.0.1	17	02e63700b827ebfffe7a80db7b227278706b223a5b7b22746d7374223a3237343937323833

ANEXO III. Tipo de información que contiene los paquetes compuestos LoRaWAN

```
Message Type = Data
            PHYPayload = 40F17DBE4900020001954378762B11FF0D

          ( PHYPayload = MHDR[1] | MACPayload[..] | MIC[4] )
                 MHDR = 40
            MACPayload = F17DBE490002000195437876
                   MIC = 2B11FF0D

          ( MACPayload = FHDR | FPort | FRMPayload )
                 FHDR = F17DBE49000200
                FPort = 01
            FRMPayload = 95437876

              ( FHDR = DevAddr[4] | FCtrl[1] | FCnt[2] | FOpts[0..15] )
             DevAddr = 49BE7DF1 (Big Endian)
               FCtrl = 00
                FCnt = 0002 (Big Endian)
               FOpts =

          Message Type = Unconfirmed Data Up
             Direction = up
                  FCnt = 2
             FCtrl.ACK = false
             FCtrl.ADR = false
       FCtrl.ADRACKReq = false
```

ANEXO IV. Tipo de información que contiene los paquetes simples LoRaWAN

```
         Message Type = Join Request
           PHYPayload = 00C40000000085C900020000000000000087F4F479262E

         ( PHYPayload = MHDR[1] | MACPayload[..] | MIC[4] )
                MHDR = 00
           MACPayload = C40000000085C900020000000000000087F4
                  MIC = F479262E

         ( MACPayload = AppEUI[8] | DevEUI[8] | DevNonce[2] )
               AppEUI = 00C98500000000C4
               DevEUI = 0000000000000002
            DevNonce = F487
```