Revista Colombiana de Computación, Vol. 20, No. 1, enero-junio 2019, pp. 41-58.
http://dx.doi.org/10.29375/25392115.3606

Implementing a Wormhole Attack on Wireless Sensor Networks with XBee S2C Devices

Implementación del ataque Wormhole en redes de sensores inalámbricos con dispositivos XBee S2C.

Julián Ramírez Gómez[1], Héctor Fernando Vargas Montoya[1], Álvaro León Henao[1]

[1]Metropolitan Institute of Technology, Fraternidad campus, 050012 Medellín, Colombia.
julianbit.ramirez@gmail.com, hvargasm@gmail.com, alvarohenao7@gmail.com

(Received: 18 January 2019; accepted: 4 February 2019)

Abstract
One of the most dangerous threats to Wireless Sensor Networks (WSN) are wormhole attacks, due to their capacity to manipulate routing and application data in real time and cause important damages to the integrity, availability, and confidentiality of network data. An empirical method to launch a successful attack on IEEE 802.15.4/Zigbee devices with source routing enabled is adopted in this work to find signatures for detecting wormhole attacks in real environments. It uses the KillerBee framework with algorithms for packet manipulation through a malicious node to capture and inject malicious packets in victim nodes. Besides, a reverse variant of wormhole attack is presented and executed. To evidence the realization of this threat by the attacking software, the experimental framework includes XBee S2C nodes. The results include recommendations, detection signatures and future work to face wormhole attacks involving source routing protocols like DSR.

Keywords: Wormhole attack, ZigBee, IoT, cybersecurity, DSR.

Resumen
Una de las amenazas más peligrosas para las redes de sensores inalámbricos (WSN) son los ataques *Wormhole* debido a su capacidad de manipular datos de enrutamiento y aplicaciones en tiempo real y causar daños importantes a la integridad, disponibilidad y confidencialidad de los datos de una red. En este trabajo, se adopta un método empírico para lanzar un ataque de este tipo (que tiene éxito) en dispositivos IEEE 802.15.4/Zigbee con enrutamiento de origen habilitado, y con ello encontrar formas para detectar ataques de tipo *Wormhole* en entornos reales. Se utiliza el *framework KillerBee* con algoritmos para la manipulación de paquetes en un nodo malicioso, para capturar e inyectar paquetes maliciosos en los nodos víctimas. Además, se presenta y ejecuta una variante inversa del ataque *Wormhole.* Para evidenciar la realización de esta amenaza por parte del software atacante, el marco experimental incluye nodos XBee S2C. Los resultados incluyen recomendaciones, firmas de detección y trabajo futuro para enfrentar los ataques *Wormhole* que involucran protocolos de enrutamiento de fuentes como DSR.

Palabras claves: Wormhole attack, ZigBee, IoT, ciberseguridad, DSR.

1. Introduction

The Internet of Things (IoT) is a growing technology aimed at connecting all kinds of electronic devices to the Internet. The purpose of IoT devices is to interact and share information to ease end users' lives. Because of it it, nearly 37 billion devices will be connected to the cyberspace by 2020 (Sahmim & Gharsellaoui, 2017). Nevertheless, IoT is a new challenge in the field of information security because a wide range of devices with different security features can be integrated, leading to a wider security gap. Furthermore, implementing security measures such as strong cipher protocols on devices with reduced processing power and memory, such as environmental sensors, is a difficult task (Rani & Kumar, 2017). One of the most important IoT technologies

Cite this work as Ramírez Gómez J., Vargas Montoya H., León Henao A. (2019). Implementing a Wormhole Attack on Wireless Sensor Networks with XBee S2C Devices Revista Colombiana de Computación, 20(1), 41-58. http://dx.doi.org/10.29375/25392115.3606

are Wireless Sensors Networks (WSN), which can be deployed in many places (e.g. homes, buildings, cities, factories and hospitals) to monitor environmental variables: temperature, humidity, movement, lighting, and also to improve processes in the industrial field (Zhu, Leung, Shu, & Ngai, 2015).

On the other hand, a considerable number of vulnerabilities and security threats related to WSNs have been presented in various research studies (Anwar, Bakhtiari, Zainal, Abdullah, & Qureshi, 2014; Goyal, Bhatia, & Verma, 2015; Patle & Gupta, 2016), which introduce potential damages to the integrity, availability, and confidentiality of information in a WSN. Some of these threats are related to the network layer in the protocols stack. They include attacks, selective forwarding, sinkholes, and wormholes, and are intended to induce an unwanted behavior in specific elements of WSNs through malicious nodes and traffic manipulation. These attacks are successful because they give an attacker the ability to intercept and modify data in real time, execute denials of service and selective forwarding attacks, store packets, inject false information into legitimate nodes and disrupt routing processes (Jao et al., 2015). The risks of wormhole attacks represent new security gaps that must be addressed and reduced to protect end users' data and privacy.

1.1 Background

Wireless Sensor Networks (WSN)

Wireless sensor networks are a group of sensors that autonomously control and monitor different physical variables in a distributed (in collection) and controlled (central control in processing) (Yang, 2014). Wireless sensor networks contemplate a wide range of applications and are implemented in different environments, according to the needs of end users, as shown in Figure 1. By collecting information and creating a communications network, a data transmission is sent to a central node Sink type. These, at the same time, have the ability to forward the information to a local or remote application for final processing.

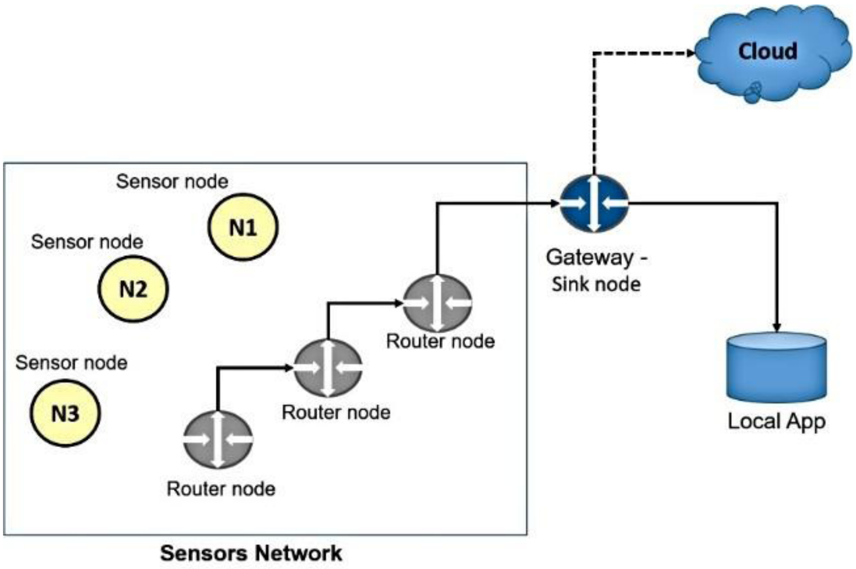


Figure 1. Typical structure of a sensor network

Each node of the network must have the capacity to send and receive information in the form of radio waves through transmitters/receivers that adapt to external antennas or are already included in the nodes. To operate properly, the nodes must have a microcontroller for data processing and storage, since it is natural (in its construction) for the nodes to convert analog signals to digital signals.

The WSN communication model comprises fewer layers than those considered in the traditional OSI model. This model includes application, routing, link management, access control, and physical layer (Forster, 2016).

Wormhole Attack

Wormhole attacks exploit the mechanisms to discover routes of on-demand routing protocols. The most remarkable cases are Ad-Hoc On-Demand Distance Vector (AODV) and Dynamic Source Routing (DSR) protocols, which use route request (RREQ) and route replay (RREP) packets as a way to discover routes by nodes in a WSN (Yih-Chun Hu, Perrig, & Johnson, 2006). A RREQ packet is a broadcast message sent by a source node ("S") to request a route to a destination node ("D"), while an RREP is a unicast message sent by the destination node in response to an RREQ. Besides, when the RREP that contains the route to reach "D" arrives at "S", the source node stores the route collected by the RREP in the route cache and then sends the application data to "D" through that route. Accordingly, the main goal of wormhole attacks is to build a tunnel between two remote nodes through a third node ("M") placed within a transmission range of "S" and "D". This occurs when "S" needs to send application data to "D" and broadcasts an RREQ message to discover a route to "D". "M," which is listening to network traffic, forwards the message directly to "D" because the RREQ sent by "M" reaches "D" before the original RREQ through the direct link. "M" can listen to the RREP from "D" first and then forward it to "S" with better metrics (zero hops), creating a false direct link between "S" and "D" through "M" in the process (Figure 2).

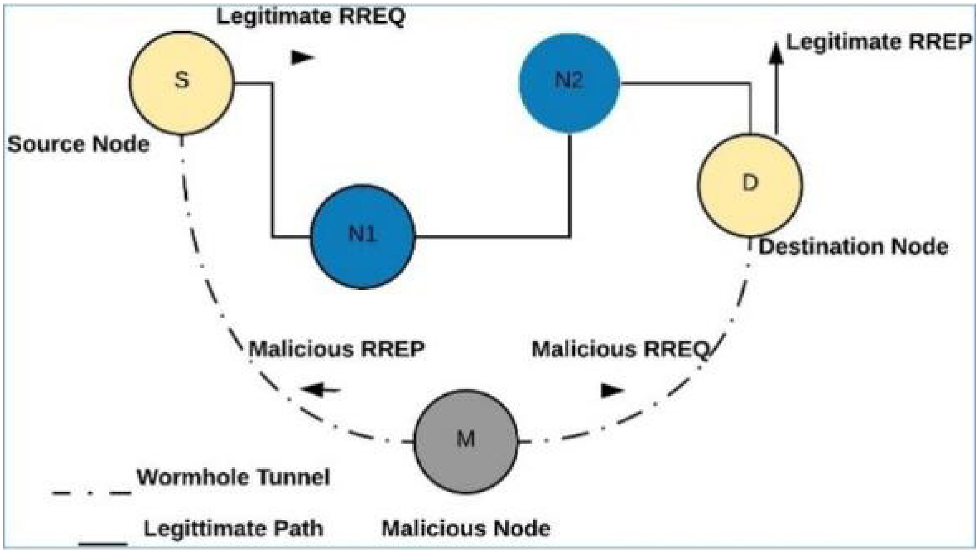


Figure 2. Wormhole attack with malicious node

At this point, the attacker can control the data that flows through the malicious tunnel and launch other attacks. Finally, if victim nodes are too far from each other, the attacker can use two malicious nodes sharing a link to build the wormhole tunnel (Jabeur, Sahli, & Khan, 2013).

1.2 Related Work

In Amish & Vaghela, wormhole attack detection is based on hop count and delay changes between source and destination nodes (Amish & Vaghela, 2016). If there is a wormhole tunnel between given source and destination nodes, the delay increases due to the longest path created by the wormhole tunnel, while hop count decreases for the same reason. In that sense, the detection scheme compares delay and hop count at a given moment with previous values to detect the attack.

Qazi, Raad, Mu & Susilo (2013), propose applying modifications to the DSR routing protocol to automatically calculate a Round Trip Time (RTT) delay value between source and destination nodes at a given moment (Qazi, Raad, Mu, & Susilo, 2013). Thus, initial RTT values are stored and compared with subsequent values of the same kind. If RTT changes, a wormhole attack is detected.

Additionally, the network nodes are set in a promiscuous mode to monitor neighboring nodes. Bhagat (2016), introduces a modified version of the AODV routing protocol to calculate the transmission force from source nodes (Bhagat & Panse, 2016). The method aims to detect wormhole attacks with high transmission power by establishing a transmission power threshold for network nodes. If a node exceeds this threshold, it could be a compromised node and a wormhole attack is detected. In another modification of the AODV protocol (Patel, Patel, & Patel, 2015), network nodes introduce the hash of the hop addresses and hop count into the RREQ packet while it follows a path from source to destination. When the RREQ packet reaches the destination node, the expected hash of the RREP is calculated and compared with the received hash. If the hashes do not match, the packet is discarded, assuming a wormhole attack in progress. Amish & Vaghela (2016) stated that to detect a wormhole attack, source nodes of the RREQ calculate the delay between a sent RREQ and every received RREP to establish an average RTT value for all received routes (Amish & Vaghela, 2016). If the RTT of one or more routes is less than the average RTT, a wormhole attack is detected, malicious routes are discarded, and the detection is replied to neighboring nodes to delete the malicious routes from their routing table. In Patel et al. (2015), every node calculates changes in the number of neighboring nodes by counting neighbors at different times.

As a result, a wormhole attack is detected if a predefined threshold of the number of neighboring nodes is exceeded by one or more nodes. BesZheng, Qian, & Wang (2015) presents a wormhole detection algorithm with node connectivity and statistical calculation (Zheng, Qian, & Wang, 2015). Such method defines two terms, node connectivity and network connectivity, to determine the probability of a wormhole attack in progress in the network. The probability of said attack depends on the network's density, which is based on the number of nodes and connections between nodes.

The research studies above conducted tests in simulation environments to measure the impact of wormhole attacks and the effectiveness of different detection/prevention algorithms in WSNs. Nevertheless, they are based on simulations of routing protocol attacks and are difficult to implement in real environments because of the lack of devices with the features required by the proposed methods. Due to existing and potential cybersecurity threats to WSNs, intrusion detection systems need to be developed for real sensor nodes. At last, since most WSN security research studies are based on simulation results, future characterization of WSN threats should focus on real devices to build actual security solutions and prevent security disasters in WSN technologies.

Marian & Mircea (2015) manage to empirically prove that the RSSI (Received Signal Strength Indicator) parameter to measure the power level of node signals can be used to detect Sybil type nodes, and is effective (Marian & Mircea, 2015). Additionally, an algorithm is proposed for detecting Sybil nodes, which consists of establishing three monitor nodes in the network, where two of the nodes collect the RSSI of the identities generated by a hypothetical Sybil node and send the values obtained to the third monitor node, which performs calculations of RSSI relationships and determines if the node is Sybil or not.

To expose the flexibility of a wormhole attack and its impact on real cybersecurity environments, this paper proposes an algorithm to execute classic and "reverse" wormhole attacks on XBee S2C devices with source routing enabled. The main goal is to modify the route record field in routing packet headers to manipulate the routing cache in victim nodes. The algorithm is implemented in Python language using the KillerBee framework and an RZUSBSTICK dongle with preinstalled KillerBee firmware. The results include recommendations to prevent wormhole attacks, attack patterns and fingerprints to develop an Intrusion Detection System (IDS) for WSNs as future work.

1.3 Attacks and Vulnerabilities of WSN Security

Due to the characteristics of the wireless sensor networks' nodes and the lack of robust security mechanisms in the protocols of the different layers of the communications model, there are a considerable amount of attacks and threats that seriously affect end users due to the risk of losing the fundamental characteristics of the information: integrity, availability and confidentiality.

WSN attacks can be classified according to the communication layer (Gaware & Dhonde, 2016; Ioannou & Vassiliou, 2016; Purohit & Sidhu, 2015; Tomic & McCann, 2017), where attacks such as denial of service, impersonation, man in the middle (MiTM) or information theft are the most common. Some of the attacks and vulnerabilities by layers can be seen in Figure 3, as well as the type of attack and possible impact.

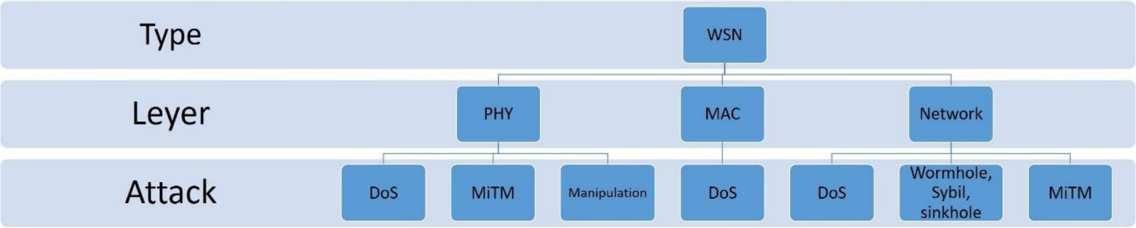


Figure 3. Classification of attacks on WSNs

Denial of service (DoS) attacks are more frequent between the layers, which constitutes a high risk for the sensors for the purposes of this article. The attacks in the layer of network are reviewed, which take advantage of point to point connection vulnerabilities, low monitoring components, a lack of configuration of cryptographic functions or having the shared means.

Denial of services – DoS: the attack occurs when the normal flow of data is interrupted, denying service from (or to) the source or destination. When this attack occurs, the WSN may be partially or totally affected, depending on where the attack occurs.

Man in the Middle – MiTM: This attack occurs when a node is maliciously entered into the network and positioned in the middle of a data stream of two (2) sensors or a sensor and a router node, capturing as much traffic circulating on the intercepted network as possible.

Sybil attack: Occurs when a malicious or compromised node assumes different identities within the network or replaces the identity of one or more nodes (Patle & Gupta, 2016).

Sinkhole attack: In this attack, a malicious or compromised node tries to attract all or at least a large part of network traffic, replacing the identity of the sink node, causing the other nodes to voluntarily send data to the node (Ioannou & Vassiliou, 2016).

Wormhole attack: In a wormhole attack, the attacker captures the packets from a specific point on the network (node) and forwards them to another previously selected point, making the two legitimate nodes involved in the attack believe that they are neighbors (Ioannou & Vassiliou, 2016).

2. Proposed Wormhole Attack Algorithm

The route record field in source routing packets contains the whole route from source to destination when the routing packet reaches the source of data transmission (Johnson, 2003; Zigbee Alliance, 2014). This feature allows the intermediary hops between source and destination nodes to introduce their network address into the routing packets (RREP) while the packet follows the path from destination to source. A route is thus created and can be used by source nodes to send data packets to the corresponding destination of the source route, as shown in Figure 4. When the route record field is void in received RREP packets, it means that both nodes source and destination are neighboring nodes.

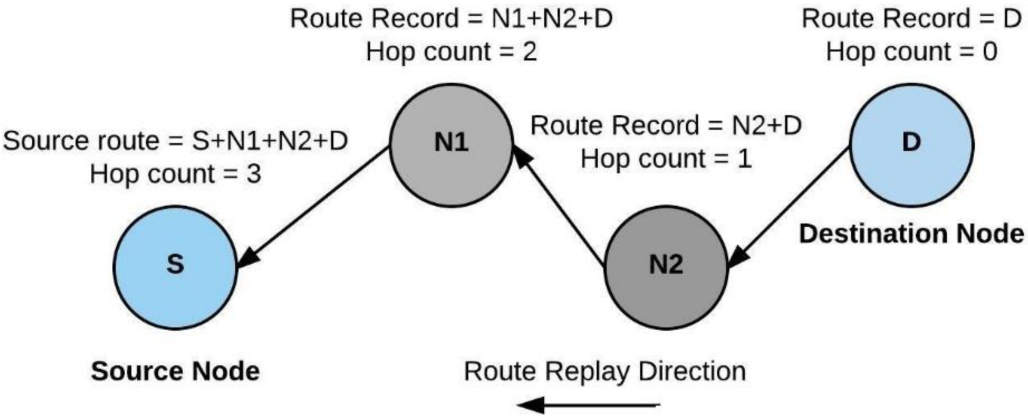


Figure 4. Route record parameter process

In a classic wormhole attack, the main goal is to create a false neighborhood between two remote nodes through a third malicious node, causing the route record field of RREP packets sent through the malicious links to be unmodifiable by intermediary nodes; as a result, they arrive at the destination with zero hops. This approach encompasses capturing packets, modifying the route record in RREP packets and injecting them into the source node to override its routing table with zero hop routes, which eventually builds a false neighborhood between source and destination nodes, as shown in Figure 5. Consequently, the route record parameter needs to be modified because RREP packets could come from an intermediary node.

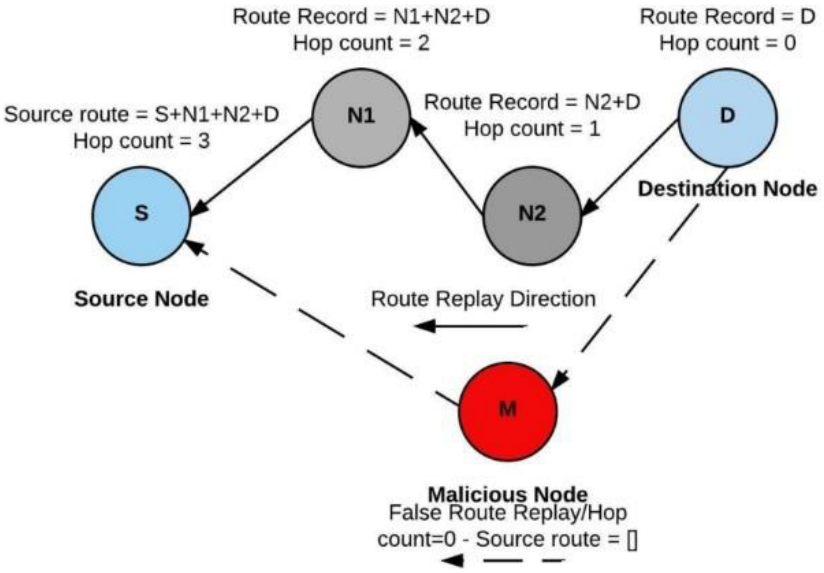


Figure 5. False route record injection

A wormhole attack begins with an attacker introducing a malicious node into a WSN to gather critical information about network attributes related to node types, network ID, frequency, and operational channel. During this step, the target nodes are selected. Subsequently, the malicious node starts a packet capture process to find routing packets involving target nodes (interesting traffic). Once the interesting traffic is captured, the malicious node sets the hop count and relay list to zero in the route record header of the routing packets. In addition, source and destination MAC addresses are changed to match the network addressing of victim nodes, since packets from an intermediary node can be captured.

Finally, the malicious node forwards the modified routing packet to the destination node, overriding its routing table with the false route and creating a false neighborhood between target nodes in the process. The next step is to continue capturing packets to find application data to be modified and injected into destination nodes. When malicious nodes are not able to capture interesting traffic, the packets are stored and the capture process is restarted. Figure 6 shows the workflow of the proposed algorithm.

In addition, two conditions must be satisfied to carry out a successful wormhole attack: (1) Source and destination addresses must match between layer 2 (802.15.4) and layer 3 (Zigbee); otherwise, the destination node of the RREP discards the packet, and (2) The packet sequence number has to be different from the original routing packet. Otherwise, the modified packet is discarded (Patel et al., 2015). The proposed attack works by overriding the destination node of the RREP's routing cache by injecting a modified version of the original routing packet, which prevents ZigBee devices from using the original route.

2.1 Impacts of Attack

The Wormhole attack impacts the confidentiality, availability and integrity of the data in the WSN, which could affect part or all of the network. Some of the impacts are described below:

Impact on confidentiality: confidentiality is the characteristic of the Information that guarantees that it can only be accessed by authorized elements or personnel, for which any unauthorized access is considered a violation (International Organization for Standardization, 2013). Under the assumption that all data collection and routing nodes are known in the WSN, including a malicious node without the respective authorization makes

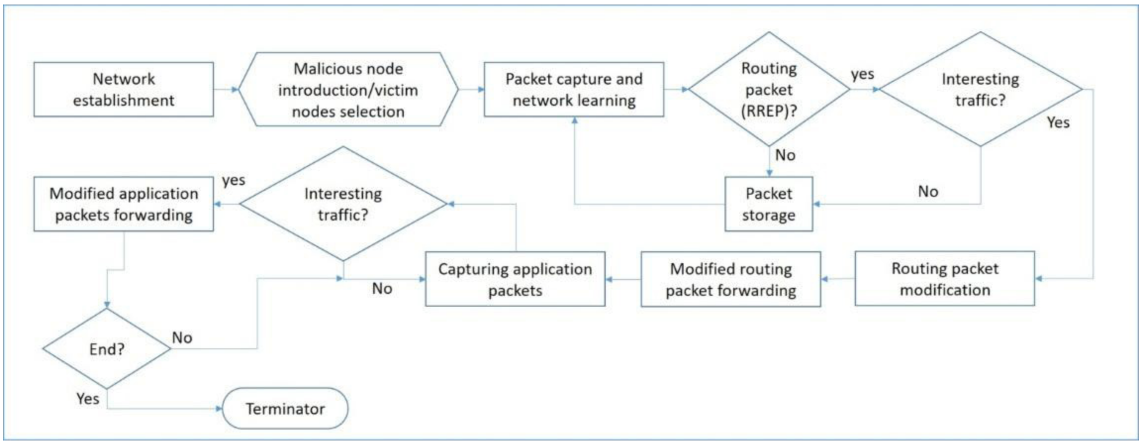

Network establishment
Malicious node introduction/victim nodes selection
Packet capture and network learning
Routing packet (RREP)?
yes
Interesting traffic?
Yes
No
Packet storage
No
Routing packet modification
Modified routing packet forwarding
Capturing application packets
Interesting traffic?
yes
Modified application packets forwarding
No
End?
No
Yes
Terminator

Figure 6. Proposed wormhole attack algorithm

the confidentiality of the network and data such that the principle of confidentiality is the first to be affected, since unregistered nodes enter the WSN, taking advantage of the lack of monitoring and prior registration.

Impact on availability: availability is the characteristic of the information that guarantees that the flow of data should not be interrupted and that the information is always available before any consultation or use (International Organization for Standardization, 2013). When a malicious node intercepts the information, it would have the possibility of blocking the origin or the destiny, since when re-doing an identity impersonation, it remains with the capacity to collect the information through the attack of man in the middle (MiTM) and the possibility to interrupt the flow of data, making the nodes and routers unavailable.

Impact on integrity: integrity is the characteristic that guarantees that information is free of modifications or alterations by third parties, keeping the data intact from the time it is created until its final disposal (International Organization for Standardization, 2013). Considering the inclusion of a malicious node in the WSN through a man in the middle (MiTM) attack, the attacker has the possibility to modify the original data received from the sensors, giving other information to the router nodes. With it, it would alter the normal flow of data.

Another relevant impact generated by a Wormhole attack would be associated with the same sensor network's credibility and stability, since it would not be possible to monitor the normal flow of data at any given time, creating an unreliable flow of data within the WSN.

2.2 Attacking Software Design

The proposed algorithm was used to develop an attacking software for real devices as a tool to probe security levels in wireless sensor networks, since most research studies describe wormhole attacks by means of simulation environments. On the other hand, the purpose of the attacking software is to expose attributes of ZigBee devices and wormhole attacks that could be used to effectively detect the latter. This section presents a short description of every phase of the attack.

2.3 Software Requirements

Scapy and KillerBee frameworks are required to dissect, capture and store packets, and also to inject malicious traffic into victim nodes. These features are combined in a Python script to execute the wormhole attack and build the malicious tunnel.

- Malicious node introduction: During this phase, an attacker sends "beacon-frame"[1] requests channel by channel to discover routing and coordinator nodes in the network, as well as device addressing and network IDs using the *zbstumbler* command of the KillerBee framework.
- Attacking software design: The attacking software presents the following attributes and functions.

Packet capture and network learning: it occurs when the attacker has selected victim nodes in the network. Then, using relevant networking data like PANID, frequency channel, and node addressing, it captures the

[1] A "beacon-frame" is a message sent by the coordinator node to synchronize the clocks with network nodes.

packets transmitted over the air through a malicious node. The following pseudocode algorithm describes the packet capture phase (see Algorithm 1):

Algorithm 1 Packet capture algorithm

```
Require: victim addr <- target_node_addressing
Require: channel <- panid_operational_channel
Require: filter <- routing_header
function main ( )
while true do
        pkt<-KillerBee sniffer (channel, filter)
        if  pkt is source_routing_packet then
        if compare (pkt, victim_addr)  then
                new_pkt <-pkt_mod (pkt, victim_addr)
                pkt_injection (new_pkt)
        else
                continue
        else
                continue
```

The attack begins by using the sniffer object of the KillerBee framework to capture packets with a ZigBee source routing header. Once a source routing packet has been captured, the next step determines if the packet belongs to a target device. If not, the while loop continues until KillerBee's sniffer captures a source routing packet that involves victim nodes.

Interesting traffic: A packet is interesting traffic when it is originated or sent from/to an attacker-defined victim device. In that sense, the attacker must dissect the captured packet, extract the addressing data and compare it with the victims' node addressing. As shown in Algorithm 1, the compare function compares addresses. Since the KillerBee sniffer generates an object from the captured packet, packet dissection becomes a simple task. It consists of retrieving the addressing data from the packet object attributes (see Algorithm 2).

Algorithm 2 Interesting traffic algorithm

```
Ensure: coincidence
function compare (pkt, victim_addr)
        src_addr<-pkt.source_address
        dst_addr<-pkt.destination_address
        if src_addr = victim_addr[0] then
        src_eval <- 1
        else
        src eval <-          0
        if dst_addr = victim_addr[1] then
        dst_eval <-    1
        else
        dst_eval <-    0
        coincidence <-          src_eval          dst_eval
        return coincidence
```

The previous algorithm determines if a captured packet involves a victim's node addressing in a data transaction. Once the addressing data is compared, the result can be true if both destination and source addresses of the captured packet and victim nodes are the same. It can also be false if one or more addressing data are not equal. In that case, the packet capture algorithm is executed again.

Routing packet modification: after finding an RREP packet with the right addressing, the attacking software changes some attributes of the routing information in the captured packet to build the malicious tunnel. It specifically modifies route record information related to hop count, relay list, and sequence number. Algorithm 3 executes the routing packet modification.

Algorithm 3 Routing packet modification

```
Ensure: new packet
function pkt_mod(pkt, victim_addr)
    new_packet pkt
    new_pkt.sequence number    Random(1, 255)
    if new_pkt.hop count1 then
    new_pkt.hop count    0
    new_pkt.relay list  []
    new_pkt.src address = victim addr0
    new_pkt.dst address = victim addr[1]
    else if pkt.hop count = 0 then
    new_pkt.hop count ← 1
    new_pkt.relay list ← [abcd]
    return new_pkt
```

Packet modification begins by rewriting the *sequence_number* of the captured packet with a random number between 1 and 255 to prevent the destination node from discarding the modified packet sent by the malicious node. At that point, the wormhole attack can present two scenarios: (1) victim nodes are further apart than one hop of distance, or (2) the victim nodes are neighbors.

The first case describes a classic wormhole attack, and the modifications of *hop_count* and *relay_list* are made to "eliminate" the distance between victim nodes. Such changes also make nodes "think" they are neighbors because of the wormhole tunnel. Because victim nodes are distant from each other, layer 2 addressing must be altered to match layer 3 addressing. The second scenario is a "reverse" wormhole attack, where victim nodes are neighbors and a malicious node tries to add distance in-between. In such a case, packet modifications are performed by increasing the *hop_count* number and adding intermediary nodes to the *relay_list*.

Routing packet forwarding: after the routing packet has been modified, the next step is to send it to its real destination with the send method of the KillerBee framework. Additionally, a new packet capture process is conducted to search for application data. The latter is used to make further modifications that may cause unwanted behavior in the WSN's application. Algorithm 4 shows the packet injection process.

Packet injection causes two possible effects in victim nodes because, once the modified packet is processed by the destination node, whether to forward the next application packets or not depends on the malicious node. If they are not forwarded, the attack may cause a Denial-of-Service (DoS) state.

Data packets modification and forwarding: as shown in Algorithm 4, this wormhole attack tries to modify application as well as routing data. In this case, the destination node of the application data would receive the attacker's data. The main differences with a replication attack are that the proposed wormhole prevents direct communication between involved victim nodes and it works over real-time traffic.

Finally, the entire process is repeated indefinitely, injecting false routes with every modified data packet sent to the destination node to maintain the wormhole tunnel until the script is stopped or moved to another network point.

Algorithm 4 Packet injection algorithm

```
Require: filter<- application_packet_header
Require: channel<-panid_operational_channel
Require: new_data<-attacker_defined_data
function pkt_injection(new_pkt)
    killerbee_send(new_pkt, channel, count <- 1
    while true do
    pkt <-killerbee_sniffer(channel, filter)
    if compare(pkt, victim_addr) then
        pkt.data = new_data
        killerbee_send(pkt, channel, count <- 1)
    else
        continue
```

3. Implementation and Results

In this section, the implementation of the proposed wormhole attack on a testing network takes place without encryption protocols applied in the packets to measure its impact on unsecured devices.

3.1 Network Requirements and Characteristics

Table 1 lists legitimate features of nodes and the parameters used to build the prototype network. The malicious node specifications are shown in Table 2. Atmel RZUSBSTICK with KillerBee firmware is used in conjunction with Raspberry Pi 3 to capture packets and inject modified data and routing packets into victim nodes.

Table 1. Legitimate node features

Type	XBee S2C (XB24C)
Firmware	405E
Functions set	ZIGBEE TH Reg
Medium Access Control	IEEE 802.15.4
Network Layer	ZigBee (Source Routing)
Frequency	2.4GHz
Router nodes	2
Coordinator nodes	1
Network ID (PANID)	10
Microcontroller	Arduino UNO - ATMEGA 328p

Table 2. Malicious node features

Node type	Raspberry Pi 3 Model B
Network interface	ATAVRRZUSBSTICK
Firmware	Killerbee
Scripting language	Python 2.7.14
Frameworks	Scapy - Killerbee
Operating system	Raspbian

In order to execute reverse and classic wormhole attacks, two testing networks were built with a coordinator node and two router nodes. Figure 7 presents a reverse wormhole scenario with router nodes sharing a direct link, which is common between neighboring nodes. On the other hand, Figure 8 shows router nodes without a direct link and the coordinator node as an intermediary node (adding one hop of distance between router nodes) to test a classic wormhole attack.

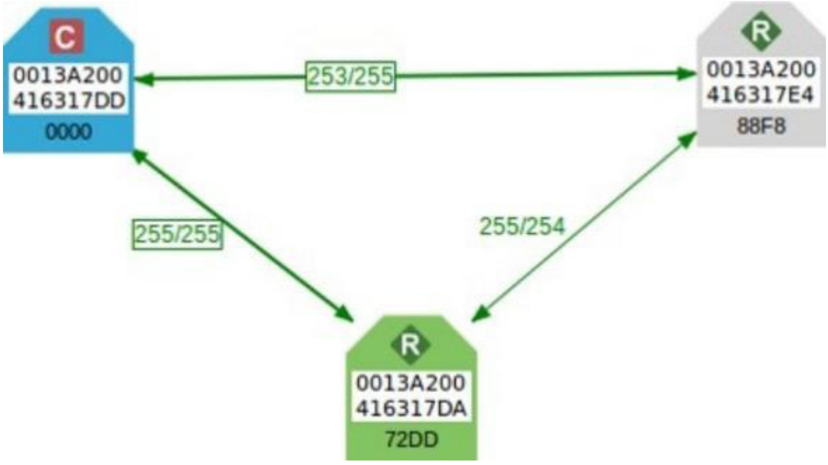


Figure 7. Prototype network for reverse wormhole attack

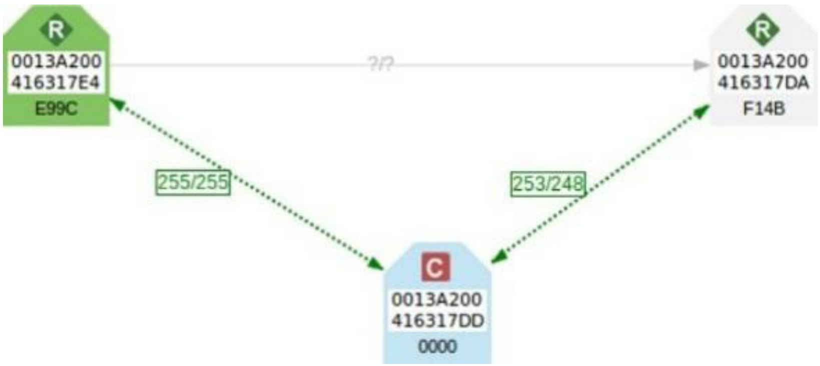


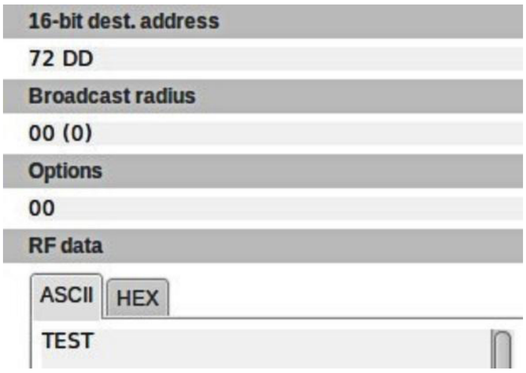
Figure 8. Prototype network for classic wormhole attack

In a reverse wormhole attack, victim nodes are identified by network addresses *0x72DD* (source of route record) and *0x88F8* (source of application data). In a classic wormhole attack, the source node has the network address *0xE99C*, while the destination node has 0xF14B. At last, the coordinator node has the default address 0x0000 in both cases.

3.2 Wormhole Attack Execution

1) Reverse wormhole attack: The main goal is to add distance between victim nodes by modifying the hop count and relay list in the routing packet, thus avoiding using the direct link shared by nodes *0x72DD* and *0x88F8*. The following command line output shows the execution of the wormhole attack script.

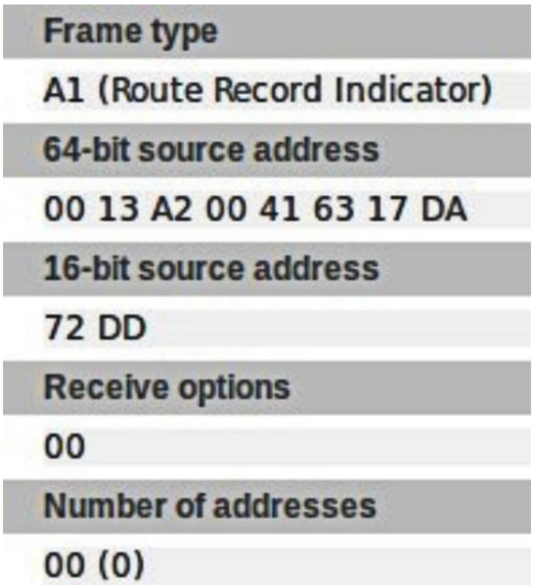
```
$ sudo python wormhole.py 0x72DD 0x88F8 14-e –f “reverse wormhole”
[**]Enter number of hops:1 [**]Enter comma separated hops:abcd [**]Sniffing and
searching
for sourcerouting...
[OK]Route record found for src’0x88f8’: [OK]Sequence Number:199
[OK]Route record parameters:
source_addr:’0x72DD’ addresses:[] hop_count:0 options:0
           id: route record [**]Inyecting route record to 88f8 Sent1packets.
[->]Fake route injected! [**]Sniffing for application data... Sent1packets.
```



16-bit dest. address
72 DD
Broadcast radius
00 (0)
Options
00
RF data

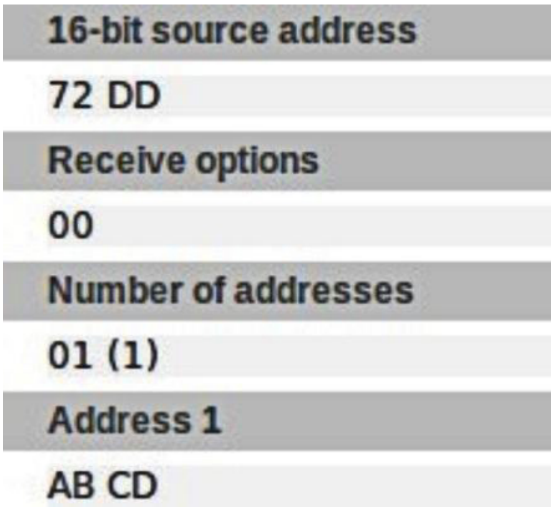
TEST

Figure 9. Original application packet payload



Frame type
A1 (Route Record Indicator)
64-bit source address
00 13 A2 00 41 63 17 DA
16-bit source address
72 DD
Receive options
00
Number of addresses
00 (0)

Figure 10. Original source routing packet for neighboring nodes



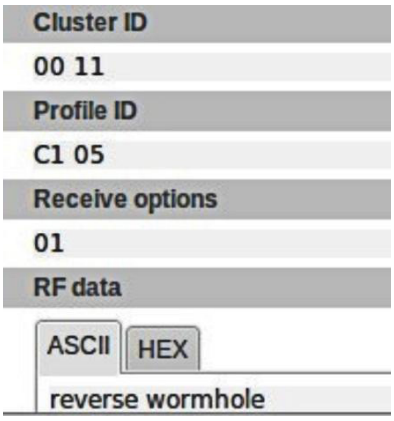
16-bit source address
72 DD
Receive options
00
Number of addresses
01 (1)
Address 1
AB CD

Figure 11. Modified source routing packet

The attack starts by capturing packets until a source routing packet involving victim nodes is found. Then, a hop counts equal to 1 and an intermediary node *(0xABCD)* is injected into the relay list parameter of the routing packet. Finally, when the script captures an application packet, the application data is replaced with the sentence “reverse wormhole”. Figure 9 shows the original application frame sent by node *0x88F8*. In this case, the original application packet has the word “TEST”. When the packet arrives at the destination node, an update of the source route is sent to *0x88F8* from *0x72DD*, as shown in Figure 10.

Figures 10 and 11 show the difference between both routes. The first one contains the attributes of the original route with 0 relays as “Number of addresses”. The second one contains the false intermediary nodes with a relay that has the address, *0xABCD*. Due to this, the malicious node is the only one that can listen to the next application packets sent by $0x88F8$, which are changed by the attacker’s malicious data (Figure 12).

Figure 11 shows the malicious route injected into $0x88F8$ when the reverse wormhole attack captures the first source’s routing packet.



Cluster ID

00 11

Profile ID

C1 05

Receive options

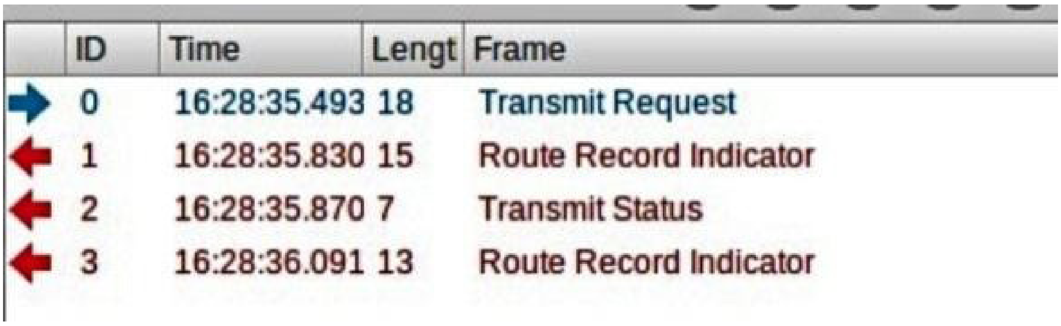
01

RF data

reverse wormhole

Figure 12. Malicious data received at destination node

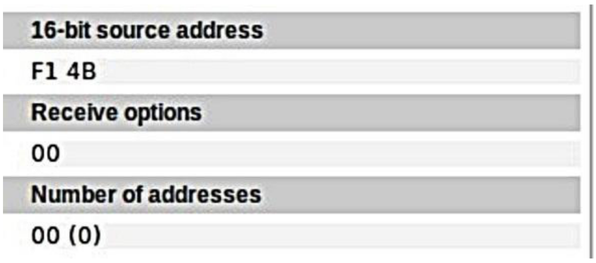
2) Classic wormhole attack: Similar to a reverse wormhole attack, this variant capture routing packets to create the malicious tunnel and application packets to inject malicious data. Figure 13 shows the legitimate and malicious routing packet received at the source node. The first route record indicator entry belongs to the original source of RREP and the second entry is the RREP modified by the malicious node to override the routing table of a victim node.



	ID	Time	Lengt	Frame
➡	0	16:28:35.493	18	Transmit Request
⬅	1	16:28:35.830	15	Route Record Indicator
⬅	2	16:28:35.870	7	Transmit Status
⬅	3	16:28:36.091	13	Route Record Indicator

Figure 13. Received routing

Once again, a source route is updated when the source node attempts to send the word “TEST” and the packet is captured and modified by the wormhole attack. Figures 14 and 15 present the changes in the route received by the source node.



16-bit source address

F1 4B

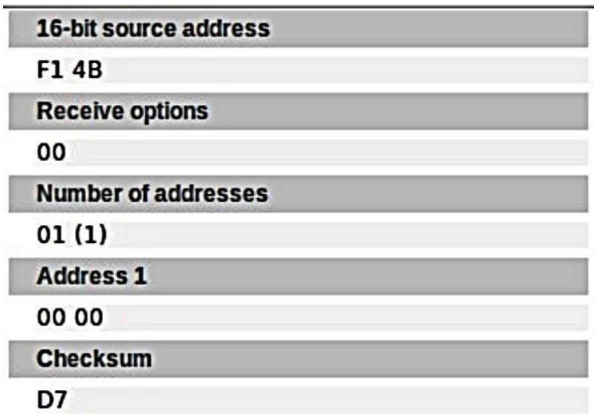
Receive options

00

Number of addresses

00 (0)

Figure 14. False source route fields



16-bit source address

F1 4B

Receive options

00

Number of addresses

01 (1)

Address 1

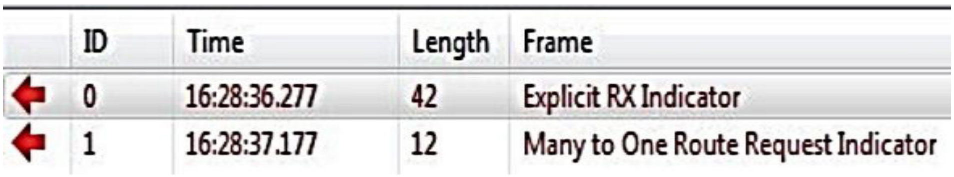
00 00

Checksum

D7

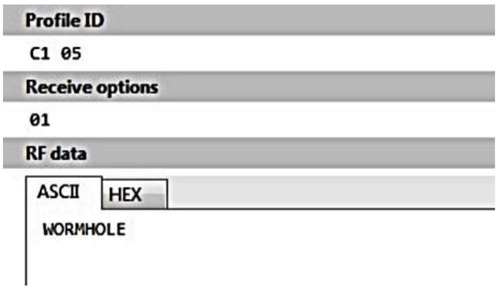
Figure 15. Original source route fields

An evident change can be observed in the field, *number of addresses* (hop count), of the source routes: the value goes from 1 hop in the first packet to 0 hops in the second. After false route injection, the source node attempts to send the word "TEST" and the task of the wormhole attack script is to replace these data with the word "WORMHOLE". Figure 16 shows the malicious data packet received by the destination node, and Figure 17 shows the content of the packet.



	ID	Time	Length	Frame
⬅	0	16:28:36.277	42	Explicit RX Indicator
⬅	1	16:28:37.177	12	Many to One Route Request Indicator

Figure 16. Modified application data received at destination node



Profile ID

C1 05

Receive options

01

RF data

WORMHOLE

Figure 17. Application data content after attack

3.3 Signatures for Wormhole Attack Detection

General detection procedure

To detect and identify attacks, an algorithm was designed that divides the tasks of the detection method into several procedures in order to allow the method to be flexible and modifiable according to the network's needs. Figure 18 shows a representation of the flows. In phase 1, the network interface is used for packet capture, then the type of packet that travels through the network is filtered and the relevant transmission and reception data is extracted. Phase 2 follows, in which a comparison is made with filtering rules (black lists and white lists) for known nodes and those that may be malicious. A validation is made of signatures that can identify if traffic is malicious or not and, depending on this, an alert is generated that allows the administrator to act quickly on the node that is threatening the WSN Security.

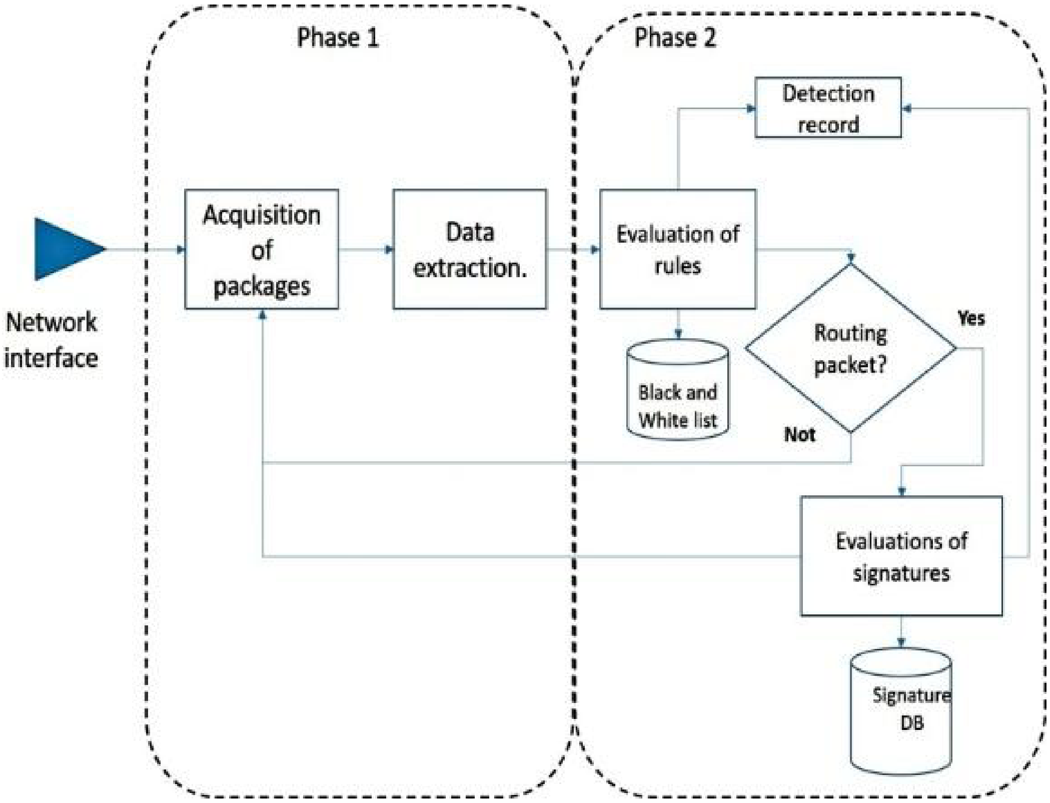


Figure 18: Attack detection flow

The signatures for detecting possible malicious nodes or attacks on the WSN are:

1) Routing packet duplication: in ZigBee devices, source routes can be requested by sending the Network Discovery (ND) command or updated when destination nodes receive a packet. In that sense, a wormhole attack must inject false routes for every modified packet that is sent, forcing the sensors/devices to receive two source routing packets per data packet transmitted to destination nodes. The abrupt changes in route record fields of the routing packets and the increase in transmitted routing packets could be used to detect the presence of an attacker in the network.

2) Multiple "beacon-frame" requests without a joined device: the first step to attack WSN is launching a discovering process to identify possible targets in the network. In 802.15.4/Zigbee networks, "beacon-frame" requests are responded by router and coordinator nodes to have new nodes join the network. However, after malicious nodes send a "beacon-frame" request, no new devices join the network. To monitor this behavior, pairing beacon request frames with newly joined devices in the WSN would help detect active scans before the wormhole attack occurs.

3) Neighborhood table and link status packets: ZigBee devices regularly send link status packets to maintain a first hop neighborhood table. Due to the fact that remote nodes cannot share link status packets, wormholes are detected by examining previous link status messages of nodes in a routing packet with a route record of zero hops. If previous link status messages are not found, a wormhole threat is detected. On the other hand, a reverse wormhole is detected by checking routing packets with route records containing more than one hop. If the nodes involved in the transmitted packet have shared link status messages before, a reverse wormhole is detected. This approach could be used with neighborhood tables instead of link status messages.

3.4 Recommendations

Due to the harmful behavior of a wormhole attack, the cryptographical features of the ZigBee specification should avoid modifying data and routing packets during wireless transmission. Furthermore, encryption keys must be regularly changed to prevent brute force attacks and reduce the functionality of possible key extraction from a stolen node. Additionally, the ZigBee specification must implement a better randomization method for the sequence number in every packet to make predicting this number difficult and prevent packet injection attacks, which causes packets with a wrong sequence number to be discarded by legitimate nodes.

4. Conclusions and Future Work

Implementing a wormhole attack in real devices was successful in using the algorithm proposed to manipulate packets with the KillerBee framework and Scapy decoders. Besides, a new variant of the wormhole attack was introduced and tested to show the flexibility and risk of malicious nodes in a network. This variant takes advantage of the vulnerability of ZigBee devices for wormhole attacks and packet injection. On the other hand, the lack of effective security measures for WSNs must be explored from an empirical point of view to close the security gap of the IoT with the available technology. This would also enable end users to implement security tools for real devices. As future work, an Intrusion Detection System (IDS) for wormhole attacks is going to be designed and implemented using signatures and patterns presented in this paper as result of the wormhole attack execution on real devices. Additionally, other experimental attacks, such as sinkhole and Sybil, will be explored to improve the detection system.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

References

Amish, P., & Vaghela, V. B. (2016). Detection and Prevention of Wormhole Attack in Wireless Sensor Network using AOMDV Protocol. *Procedia Computer Science*, *79*, 700–707. https://doi.org/10.1016/j.procs.2016.03.092

Anwar, R. W., Bakhtiari, M., Zainal, A., Abdullah, A. H., & Qureshi, K. N. (2014). Security Issues and Attacks in Wireless Sensor Network. *World Applied Sciences Journal*, *30*(10), 1224–1227. https://doi.org/10.5829/idosi.wasj.2014.30.10.334

Bhagat, S., & Panse, T. (2016). A detection and prevention of wormhole attack in homogeneous Wireless sensor Network. In *2016 International Conference on ICT in Business Industry & Government (ICTBIG)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICTBIG.2016.7892696

Forster, A. (2016). *Introduction to Wireless Sensor Networks*. John Wiley & Sons.

Gaware, A., & Dhonde, S. B. (2016). A Survey on Security Attacks in Wireless Sensor Networks. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 536–539).

Goyal, S., Bhatia, T., & Verma, A. K. (2015). Wormhole and Sybil attack in WSN: A review. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp.1463–1468).

International Organization for Standardization. (2013). ISO/IEC 27001:2013. Retrieved January 17, 2019, from https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en

Ioannou, C., & Vassiliou, V. (2016). The Impact of Network Layer Attacks in Wireless Sensor Networks. In *2016 International Workshop on Secure Internet of Things (SIoT)* (pp. 20–28). IEEE. https://doi.org/10.1109/SIoT.2016.009

Jabeur, N., Sahli, N., & Khan, I. M. (2013). Survey on Sensor Holes: A Cause-Effect-Solution Perspective. *Procedia Computer Science*, *19*, 1074–1080. https://doi.org/10.1016/j.procs.2013.06.151

Jao, M.-H., Hsieh, M.-H., He, K.-H., Liu, D.-H., Kuo, S.-Y., Chu, T.-H., & Chou, Y.-H. (2015). A Wormhole Attacks Detection Using a QTS Algorithm with MA in WSN. In *2015 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 20–25). IEEE. https://doi.org/10.1109/SMC.2015.17

Johnson, D. B. (2003). *The dynamic source routing protocol for mobile ad hoc networks (DSR)*.

Marian, S., & Mircea, P. (2015). Sybil attack type detection in Wireless Sensor networks based on received signal strength indicator detection scheme. In *2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics* (pp. 121–124). IEEE. https://doi.org/10.1109/SACI.2015.7208183

Patel, A., Patel, N., & Patel, R. (2015). Defending against Wormhole Attack in MANET. In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 674–678). IEEE. https://doi.org/10.1109/CSNT.2015.253

Patle, A., & Gupta, N. (2016). Vulnerabilities, attack effect and different security scheme in WSN: A survey. In *2016 International Conference on ICT in Business Industry & Government (ICTBIG)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICTBIG.2016.7892724

Purohit, R., & Sidhu, N. (2015). Wireless sensor network: Routing protocols and attacks- a survey. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 2130–2135).

Qazi, S., Raad, R., Mu, Y., & Susilo, W. (2013). Securing DSR against wormhole attacks in multirate ad hoc networks. *Journal of Network and Computer Applications*, *36*(2), 582–592. https://doi.org/10.1016/j.jnca.2012.12.019

Rani, A., & Kumar, S. (2017). A survey of security in wireless sensor networks. In *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)* (pp. 1–5). IEEE. https://doi.org/10.1109/CIACT.2017.7977334

Sahmim, S., & Gharsellaoui, H. (2017). Privacy and Security in Internet-based Computing: Cloud Computing, Internet of Things, Cloud of Things: a review. *Procedia Computer Science*, *112*, 1516–1522. https://doi.org/10.1016/j.procs.2017.08.050

Tomic, I., & McCann, J. A. (2017). A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols. *IEEE Internet of Things Journal*, *4*(6), 1910–1923. https://doi.org/10.1109/JIOT.2017.2749883

Yang, S.-H. (2014). *Wireless Sensor Networks. Principles, Design and Applications*. London: Springer London. https://doi.org/10.1007/978-1-4471-5505-8

Yih-Chun Hu, Perrig, A., & Johnson, D. B. (2006). Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, *24*(2), 370–380. https://doi.org/10.1109/JSAC.2005.861394

Zheng, J., Qian, H., & Wang, L. (2015). Defense Technology of Wormhole Attacks Based on Node Connectivity. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)* (pp. 421–425). IEEE. https://doi.org/10.1109/SmartCity.2015.107

Zhu, C., Leung, V. C. M., Shu, L., & Ngai, E. C.-H. (2015). Green Internet of Things for Smart World. *IEEE Access*, *3*, 2151–2162. https://doi.org/10.1109/ACCESS.2015.2497312

Zigbee Alliance. (2014). Standards: ZigBee Specification. Retrieved January 17, 2019, from https://www.zigbee.org/download/standards-zigbee-specification/

About the authors

Julian Ramirez Gómez

Julian Ramirez is a security analyst, telecommunications engineer and holds a master's degree in computer security from the Metropolitan Institute of Technology in the city of Medellín, Colombia. Julian has been working on cybersecurity issues at the level of networks and operating systems.

Héctor Fernando Vargas Montoya

Héctor Vargas is a System Engineer with a master's degree in ICT security. He is the coordinator and teacher of the master's degree in computer security at the Metropolitan Institute of Technology in the city of Medellín, Colombia. He is directing a master's thesis on cybersecurity and information security. He has worked for more than 17 years on security issues in telecommunications companies and as an independent consultant.

Álvaro León Henao

Alvaro Henao is a telecommunication engineer and has a master's degree in computer security from the Metropolitan Institute of Technology in the city of Medellín, Colombia. He has been working on IT, telecommunication and security issues in the banking sector.