Implementation of a Wormhole Attack Against a RPL Network: Challenges and Effects

Pericle Perazzo*, Carlo Vallati*, Dario Varano*, Giuseppe Anastasi* and Gianluca Dini*
* Department of Information Engineering
University of Pisa
Pisa, 56122, Italy
Email: *[name.surname]*@iet.unipi.it

***Abstract*—In the near future, the Internet of Things will cheaply connect smart devices, in such a way to form large Wireless Sensor and Actuator Networks (WSANs). For its characteristics, the Routing Protocol for Low-Power and Lossy Networks (RPL) is considered the standard choice for WSANs. Since they often carry sensitive or safety-critical data, securing these networks from cyberattacks is paramount. One of the subtlest security attacks in RPL WSANs is the *wormhole attack*, in which a malicious actor establishes and controls an out-of-band channel between two distant nodes of the network. Due to its convenience, RPL is induced to use such a channel to forward the traffic. As a result, the malicious actor can control a potentially large amount of traffic and can eavesdrop or discard it. The wormhole attack cannot be avoided by traditional cryptographic countermeasures, for example by encrypting and authenticating all the traffic. Nevertheless its importance, the wormhole attack has been studied exclusively by theory. The practical techniques to realize it on a WSAN have not been studied until now.**

The contribution of this paper is two-fold. First, we present an implementation of a wormhole capable of attacking an IEEE 802.15.4-based WSAN, using also a technique to increase its impact (*proxy acker technique*). We test the realized wormhole against a real WSAN, measuring its impact with respect to various parameters. As a second contribution, we discuss the various countermeasures proposed by the literature, and we test the feasibility of one of them in practice. We conclude that the most convenient way to counteract a wormhole attack in a WSAN may be to avoid subsequent attacks, i.e., traffic eavesdropping and selective packet dropping.

I. Introduction

Recent advancements in computing and communication technologies are currently paving the road to make real the Internet of Things (IoT), a vision in which smart objects, i.e., common objects empowered with communication capabilities, are seamlessly integrated into information systems [1]. IoT is expected to trigger significant changes into many areas of our lives: health (e.g., remote patient monitoring), home (e.g., smart lighting and heating), and city (e.g., smart traffic and parking applications). In this context, Wireless Sensor and Actuator Networks (WSANs) will represent a key building block as they will guarantee rapid installation of smart objects to cover large areas, so keeping the deployment costs low. Data delivery through wireless links in a multi-hop fashion reduces the need for complex network infrastructure and guarantees the flexibility required for expansion and evolution.

In order to ease the integration of WSANs into existing information systems, IETF has standardized a set of protocols for *IoT WSANs*. The communication protocol stack is built on top of the IEEE 802.15.4, widely exploited in WSANs deployments [2], and adopts IPv6 as communication protocol. To this aim, the group has defined 6LoWPAN, an adaptation layer to allow the transmission of IPv6 packets on IEEE 802.15.4 networks, and the IPv6 Routing Protocol for Low-Power and Lossy Networks, RPL [3], considered the standard routing solution for IoT [4].

Although both the IEEE 802.15.4 and the RPL standards include a set of mechanisms to ensure the security of communication and the resiliency of network control operations against malicious actions [5], [6], the shared and open nature of the wireless medium makes WSANs intrinsically vulnerable to a wide range of security attacks [7]. Among them, the *wormhole attack* [8] is one of the subtlest, because it is hard to detect and to avoid. In a wormhole attack, a malicious actor establishes and controls an out-of-band channel between two distant nodes. Due to the convenience of this channel, the routing service is induced to use it to forward the traffic. As a result, the malicious actor controls a potentially large amount of traffic and can eavesdrop or discard it. The wormhole attack is feasible also if the routing messages are encrypted and authenticated, without the need of stealing any cryptographic secret from the nodes. Nevertheless its importance, the wormhole attack has been studied exclusively by theory. The practical techniques to realize it on a WSAN have not been studied until now.

The contribution of this paper is as follows.

- We present an implementation of a wormhole capable of attacking an IEEE 802.15.4 WSAN, using also a technique to increase the impact on RPL routing protocol (*proxy acker technique*). As a proof of concept, we attack a real WSAN with our wormhole, and we measure the attack impact with respect to various parameters.
- We discuss the various countermeasures proposed by the literature, and we test one of them with real experiments. We conclude that avoiding or detecting a wormhole attack may be too expensive for the typical IoT WSAN, which is resource-constrained. A most convenient way may be to avoid or detect subsequent attacks, namely traffic eavesdropping and selective packet dropping.

The rest of the paper is organized as follows. In Section

II we review related work about RPL attacks and impact assessment of wormholes. In Section III we give the technical background about IEEE 802.15.4 and RPL protocols. In Section IV we give a detailed description of our wormhole implementation and the proxy acker technique. In Section V we attack a real WSAN with our wormhole, and we measure the attack impact with respect to various parameters. In Section VI we discuss the various countermeasures proposed by the literature, we test one of them with real experiments, and we conclude that the most convenient way to counteract a wormhole could be to avoid or detect subsequent attacks. In Section VII we draw the conclusions.

II. Related Work

Since its proposal by the IETF ROLL group, the research community analyzed the security of RPL and assessed the impact of various attacks [9], [10]. Dvir et al. [11] studied the effects of the *sinkhole attack*, in which a malicious node advertises a falsely convenient path to the root, in order to attract traffic from surrounding nodes. The malicious node can then eavesdrop and/or discard the attracted traffic. Mayzaud et al. [12] studied the impact of the *DODAG Version attack*, which has a similar effect. The majority of research papers focused on sinkhole and DODAG Version attacks [11], [13], [14], [15], and proposed various countermeasures. Though very disruptive, these attacks are easy to avoid (at least with non-internal adversaries) by means of cryptographic authentication of the RPL messages. Perazzo et al. [16] analyzed the effects of the *DIO suppression attack*, by which a malicious node can partition a RPL network also if some cryptographic authentication is active. However, also this attack can be avoided by an effective replay protection mechanism, based again on cryptography. Standard-compliant implementations of such authentication and replay protection mechanisms are already available for IEEE 802.15.4 [5] and RPL [6]. On the contrary, the wormhole attack is not avoidable by means of mere cryptography at all and constitutes thus a subtler threat.

The wormhole attack has been introduced by Hu et al. [8], who made also a first qualitative impact assessment. The particular severity of the attack has attracted considerable research, some of which [17], [18], [19], [20] have quantitatively assessed the impact of a wormhole attack against simulated ad-hoc networks. The most important study on wormhole impact is perhaps Khabbazian et al. [20], which developed a complete theoretical framework to estimate the impact of a wormhole against a generic network that uses a hop-count routing metric. Khabbazian et al. and the other papers in the literature studied the wormhole attack exclusively by theoretic analysis or simulation. The practical techniques to perform it on a real WSAN have not been studied until now. In this paper, we present an implementation of a wormhole capable of attacking an IEEE 802.15.4 WSAN. As a proof of concept, we attack a real WSAN with our wormhole, and we measure the impact with respect to various parameters.

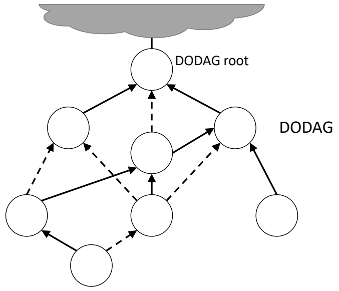

Fig. 1. Example of DODAG. Solid arrows point to preferred parents, dashed arrows point to other parents in the parent set.

III. Technical Background

A. IEEE 802.15.4 MAC Protocol

The IEEE 802.15.4 standard [21] is a wireless communication standard for low-power, low-rate and low-cost WSANs. The standard supports different network topologies and different channel access modes. In this paper we focus on networks with mesh topologies in which multi-hop communication is achieved by means of a routing protocol left unspecified by the standard. As far as the channel access mode is concerned, we consider the nonbeacon-enabled mode that is the one most widely adopted by practical deployments. With this mode, nodes communicate in a distributed manner using the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) algorithm. The transmission of both broadcast and unicast frames is supported. In order to ensure proper delivery, unicast transmissions are acknowledged by the receiver through acknowledgment frames (ACKs) that are sent in response.

B. RPL Routing Protocol

RPL [3], [4] is a distance-vector routing protocol specifically designed for constrained devices. Its design assumes that the majority of the application traffic is upward, i.e., generated by nodes and directed towards a single node acting as a border router. For this reason, RPL builds and maintains a logical topology for upstream data delivery, while downward routes are established only when required. The logical topology built by RPL is a *Destination Oriented Directed Acyclic Graph* (DODAG), an example of which is shown in Fig. 1. In a DODAG, every node selects a set of neighbors, called *parent set*, as candidates for upstream data delivery. One of the nodes within the parent set is selected as the *preferred parent*, which is exploited for the actual data forwarding. The DODAG is rooted in a single node, the *DODAG root*, to which all upstream data is directed. The DODAG root acts also as a border router for other networks. The DODAG root triggers the RPL topology formation by emitting *DODAG Information Object* (DIO) messages. Non-root nodes listen for DIOs and use the included information to join the DODAG. Upon joining the DODAG, a node also starts emitting DIOs to advertise its presence and its distance to the root. The emission of DIOs is regulated by the *Trickle algorithm* [22]. Trickle was originally

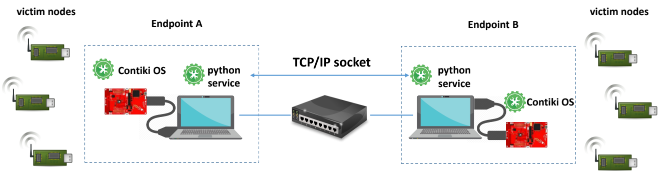

Fig. 2. Wormhole implementation schema.

designed for polite gossiping in wireless networks, to reduce the power consumption of the nodes by minimizing the redundant messages and by dynamically adapting the transmission rate. The Trickle algorithm divides time in periods of variable length. The node schedules the transmission of a DIO message at a random time t in the second half of each period. Until t, the node listens for messages and keeps track of the DIOs. At time t, the scheduled DIO message is broadcast unless a certain number of DIOs have been already received, above a given suppression threshold. At the end of the period, the length of the next period is doubled, until a maximum length I_{max} is reached (*max Trickle period*). At any time, if a DIO that advertises updated information is received, the Trickle algorithm is reset, meaning that the current Trickle period is interrupted and the algorithm starts over from a period of a minimum length I_{min} (*min Trickle period*). The asynchronous emission of DIOs can be requested by broadcasting *DODAG Information Solicitation* (DIS) messages, which cause the Trickle algorithm to reset on the receiving nodes.

Each DIO message specifies the *rank* of the sender, which is a scalar measure of its distance to the root node. According to RPL specifications, in order to avoid loops in the logical topology, the rank must monotonically increase along each path as the distance to the root increases. The rank is calculated by each node according to an *Objective Function* (OF). Each OF defines the policy for the selection of the preferred parent based on the rank of neighbors. Different OFs have been proposed, among them the Minimum Rank with Hysteresis Objective Function (MRHOF) [23] is often adopted as it aims at improving the stability of the routing protocol by reducing route flaps caused by small metric fluctuations. To this end, MRHOF adopts a hysteresis mechanism: a new preferred parent is selected only if it has rank that is more convenient than the current preferred parent by a *parent switch threshold* (Δ_{rank}).

Although the OF can take into account a wide range of communication costs for rank computation, the Expected Transmission Count (ETX) is widely adopted. ETX is defined as the inverse of the product of the Packet Reception Rate (PRR) computed using received packets and the PRR computed using received ACKs [24].

IV. Wormhole Implementation

In this section we present our wormhole implementation. Such an implementation has been made publicly available on Github[1]. In a nutshell, our wormhole acts by replaying the traffic sniffed from one portion of the network to another portion and vice versa. It operates at the MAC layer, meaning that it never interprets higher-layer information carried in the payload of the MAC frames. This makes the attack possible even if the whole traffic is secured through MAC-layer encryption and authentication. Indeed, MAC-layer encryption encrypts the MAC payload, which the wormhole never interprets, whereas the MAC-layer authentication protects the integrity of the MAC header and the MAC payload, which the wormhole never modifies.

The realized schema is pictured in Fig 2. The wormhole is realized through two endpoints deployed in different areas of the victim network. Each endpoint is composed of an IEEE 802.15.4 sensor board connected via serial link to a laptop on which a python service runs. The two laptops communicate using a local network, e.g., an Ethernet or WiFi connection, or an Internet connection, if the victim network is large. It is worth to note that, in a real attack scenario in a human-monitored environment, it may be convenient for the adversary to replace laptops with smaller devices, which are easier to hide. Any device with a serial connection (e.g., an USB port), an Internet connection, and the capability of running python scripts is suitable. For example a cheap Raspberry Pi board.

For the IEEE 802.15.4 sensor boards we adopted two CC2650 Launchpad boards by Texas Instruments[2]. It is a low-cost evaluation board designed for fast prototyping that is equipped with an IEEE 802.15.4 transceiver. The board runs the Contiki operating system[3], a popular operating system for sensor devices. Each endpoint implements two different functions: the *sniffer* and the *replayer*. Each function is implemented with two processes: one Contiki process running on the board and another python process running on the laptop.

The Contiki sniffer process continuously captures all the traffic transmitted in the network by programming the wireless transceiver in promiscuous mode at bootstrap. Every time a frame is eavesdropped, the process forwards it to the python sniffer process running on the laptop through the serial link between the board and the laptop. The python sniffer process is programmed to forward the frame to the other endpoint, specifically to the replayer python process running on the other laptop, using a TCP socket. The python replayer process, instead, is responsible to receive the frames from the other endpoint and to retransmit them. The retransmission is performed by forwarding the frame to the board. The Contiki replayer process is programmed to retransmit every frame that is received using the IEEE 802.15.4 interface.

A. Proxy Acker Technique

The sniffer and replayer functions allow frames to be transmitted from one endpoint to the other and vice versa. An unicast transmission that takes place between two nodes on two different sides of the wormhole, however, requires

[1] Wormhole implementation: https://github.com/darvarr/wormhole

[2] http://www.ti.com/tool/launchxl-cc2650

[3] http://contiki-os.org/

proper delivery of ACKs. Even though they can be transmitted through the wormhole as regular frames, the wormhole introduces an additional latency that might cause the transmitter to ignore the acknowledgment. Based on the IEEE 802.15.4 specifications an ACK must be received before a certain timeout, whose value might be shorter than the latency introduced by the wormhole, since the two endpoints could communicate even through an Internet connection. In this case, the ACK may be ignored by the transmitter, which could consequently increase its ETX estimate, so that the wormhole link would appear less convenient for routing.

In order to avoid this, each endpoint implements in the board an additional process, the *proxy acker*, which is responsible for generating ACKs for the unicast frames that are transmitted from one endpoint to the other, to mimic the proper transmission of ACKs through a real (fast) wireless link. In this way, the ACKs are received timely and the victim nodes do not increase their ETX estimate. Also, the proxy acker technique increases the probability that a victim node receives ACKs without errors. This is because ACKs are transmitted directly by the wormhole endpoint and do not have to be received by the other endpoint and pass through the wormhole, which is a lossier procedure. This additionally decreases the ETX estimate that the victim node makes of the wormhole link, thus increasing its apparent convenience for routing.

The proxy acker is programmed to generate an ACK for each unicast frame that has as destination a node located on the other side of the wormhole. The MAC addresses of such nodes are stored on a MAC table populated through backward learning. Every time a frame is received and replayed by the Contiki replayer process, the latter adds the source MAC address of the frame in the MAC table, if not already available. In order to avoid message storms, the MAC table is also exploited by the sniffer process to filter the unicast messages to be forwarded on the other side.

In Fig. 3 we illustrate the flow diagram of our wormhole implementation with the proxy acker technique, as it typically works on a RPL network. The diagram assumes that one endpoint (Endpoint A) is located closer to the RPL root node, while the other (Endpoint B) is farther. At first, the MAC tables of both the endpoints are empty, therefore, the first message transmitted through the wormhole is a broadcast message, i.e., a DIO message. Let us assume that the first DIO message transmitted through the wormhole is from Endpoint A to Endpoint B (a DIO message transmitted from Endpoint B to Endpoint A would be likely ignored as it conveys a higher rank making no changes in the network behavior). The DIO message is sniffed by the board of Endpoint A (1), then it is forwarded to the python sniffer process via serial (2). Since the message is broadcast, the process forwards the message to the python replayer process on Endpoint B through the TCP socket (3), which adds the source MAC address to the MAC table and then sends the message to the Contiki replayer process on the board via serial connection (4). The message that is broadcast is received by the nodes in reception range of Endpoint B, which likely select the sender of the DIO message as new preferred parent, considering that it has a lower rank. When a unicast message is forwarded to the preferred parent by one of those nodes, the message is received by the Contiki sniffer process of Endpoint B (5). Since the MAC address is known, the message is forwarded to the python sniffer process (6), while the proxy acker generates an ACK (7). The frame is eventually transmitted in proximity of Endpoint A, following the same reverse path (8-11).

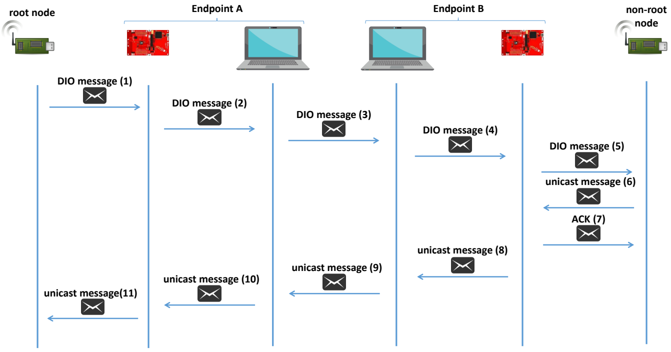

Fig. 3. Typical attack information flow with proxy acker technique.

Note that the proxy acker technique is applicable also if all network traffic is authenticated at the MAC layer. This is because the IEEE 802.15.4 standard does not allow to authenticate ACKs. Thus an adversary is always able to transmit forged ACKs without being detected [25].

V. Experiments

A. Victim Testbed and Wormhole Placement

Our victim testbed is a network of 22 6LoWPAN wireless sensor nodes deployed at our department in University of Pisa (Fig. 4) [26]. Each sensor is a TMoteSky[4]. The sensors run the Contiki OS which executes the RPL routing protocol. Each sensor sends an UDP packet to the root once a minute. All the traffic generated by the testbed is encrypted and authenticated at the MAC layer, by means of an AES-128 key preinstalled in each node. This is to prove the feasibility of the attack and of the proxy acker technique even in this case.

B. Best Wormhole Placement

In order to study the maximum impact of a wormhole in a wireless network, Khabbazian et al. [20] suggest to place the wormhole endpoints at the intersection points of the transmission ranges of the nodes. Unfortunately, in our testbed the transmission ranges are not easily measurable, and they may also change over time due to environment variability. In this case, the same authors [20] suggest to place the wormhole endpoints in the close proximity of the victim nodes. In this way, what we get is a lower bound of the real maximal impact of a wormhole. We chose to proceed in this latter way. We performed some preliminary experiments to identify the best wormhole placement. Fig. 5 shows a typical example of RPL

[4] https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html

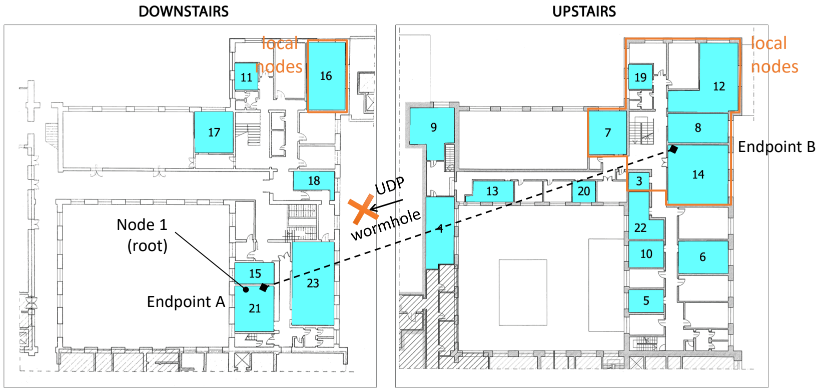

Fig. 4. Testbed map with the victim 6LoWPAN wireless sensor network and the wormhole. The numbers indicate nodes, each of which is deployed in a different room. Node 1 (the root node) and Node 21 are in the same room.

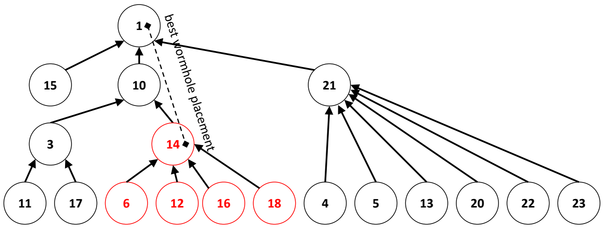

Fig. 5. Example of RPL topology built by our victim testbed after 1 hour of operation. The dashed line reports the best wormhole placement in terms of number of affected nodes. In red color the affected nodes.

topology that the nodes build after 1 hour of network activity.

From this topology, we can see that the best wormhole configuration is to place one endpoint close to Node 1 (the root) and another endpoint close to Node 14. Indeed, Node 14 is selected by multiple legitimate upward routes: the routes from Nodes 6, 12, 16, 18, and 14. It is worth to notice that, instead of Node 14, which is two-hop distant from the root, one of the nodes one-hop distant from the root (e.g., Node 21) could have been selected to capture more routes. Such a configuration, however, would have been ineffective: since the legitimate nodes are in direct communication, the adversary would not have controlled any route. Our preliminary experiments confirmed that in most of the cases the best placement of the wormhole endpoints is close to Nodes 1 and 14, hence this configuration has been adopted in all the following experiments as shown in Fig. 5.

C. Experiments Set Up

We performed experiments which are divided in four phases: (i) a *warm-up phase* (2 hours) in which the wormhole is off and the victim testbed stabilizes and converges towards legitimate routes; (ii) an *attack phase* (1 hour) in which the wormhole operates; and a (iii) *post-attack phase* (1 hour) in which the wormhole is turned off again and the victim testbed recovers from the attack. To make measurable the effect of the wormhole, we drop every unicast frame flowing from Endpoint B to Endpoint A, thus realizing a denial-of-service attack. Unicast frames are easily identifiable by the wormhole, since MAC header is not encrypted. The effect of this wormhole is that the DIOs sent by the root are replayed towards the far away nodes, so that these nodes are likely to choose the root as their preferred parent. When the nodes try to send UDP data packets to the root, these packets get lost, because the wormhole drops them. The victim nodes does not detect such data loss, because Endpoint B sends ACKs to them and illudes them that the lost packets have been correctly received. This attack could be mounted, for example, against a cryptographically protected wireless sensor network for environment monitoring and event reporting, to obtain a denial of service without stealing secret keys or performing energy-expensive and easy-to-detect jamming attacks.

D. Impact of the Attack

We measure the impact of our wormhole attack with the following metrics.

- *Wormholed nodes*, defined as the number of nodes whose path to the root passes through the wormhole.
- *Global packet loss*, defined as the percentage of undelivered UDP data packets directed towards the root within a given interval time. The packet loss can be caused by both regular network events (e.g., wireless transmission errors), or by the wormhole.
- *Local packet loss*, defined in the same way as global packet loss but measured only on the nodes geographically close to Endpoint B, namely Nodes 3, 7, 8, 12, 14, 16, 19. Such nodes are labelled as “local nodes” and circled in red in Fig. 4.

For each combination of parameters, 10 independent experiment replicas have been run to obtain more statistically sound results.

Fig. 6 shows the average wormholed nodes over time, with a max Trickle period set to 17min 29sec ($= 2^{20}$ milliseconds) and $\Delta_{rank} = 1.0$, which represent the default Contiki settings. It can be seen that the wormhole affects the upward routes of 4-5 nodes in average at its maximum effect. Note that the

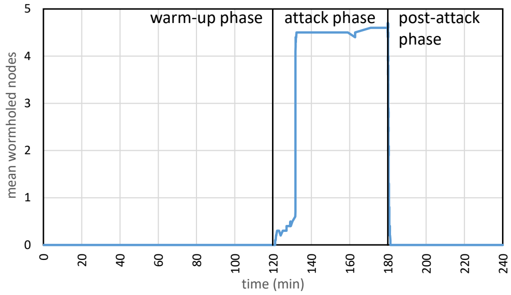

Fig. 6. Average wormholed nodes wrt time, with $I_{max} = 17\text{m}29\text{s}$ and $\Delta_{rank} = 1.0$.

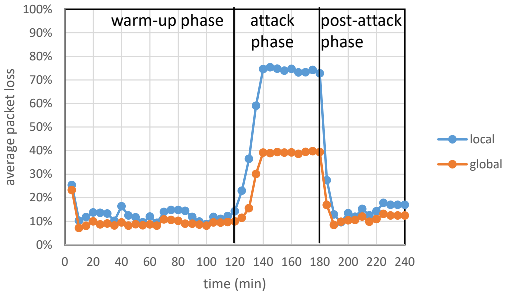

Fig. 7. Average packet loss wrt time.

attack does not have an immediate effect, but it needs some time. This is because after the warm-up phase, the root sends a DIO message each max Trickle period, which is non-negligible (17min 29sec). In order to speed up the attack effect, the adversary should cause a reset of the Trickle period of the root node, for example by sending a malicious DIS message. This is possible only if she can forge a malicious DIS message or replay a DIS message sent before by a legitimate node. This is not the case of our experiments, in which all the legitimate traffic is authenticated at the MAC layer. On the other hand, the recovery of the network after the attack is quick. This is because RPL nodes notice that they do not receive anymore ACKs from the preferred parent, and they quickly react by selecting a failover parent.

Fig. 7 shows the average global and local packet loss with time. As expected, when the wormhole is disabled, the network experiences the packet loss caused by normal network events, such as packet transmission errors. When the wormhole is activated, instead, the global and local packet losses increase noticeably, roughly following the trend of the number of wormholed nodes reported in Fig. 6. As can be seen considering the local packet loss rate, the increase in the global packet loss is due to the increase of the packet loss experienced by the nodes in proximity of the Endpoint B, which report an average packet loss of approximately 80%. Although such results are omitted here for the sake of brevity, if we look at the single values of the packet loss experienced by such nodes, we can see that the majority of the nodes report a packet loss of 100% during the attack, while a minority a value of approximately 10%. This confirms that not all the nodes in proximity of Endpoint B are affected and continue to send their packets towards the legitimate path.

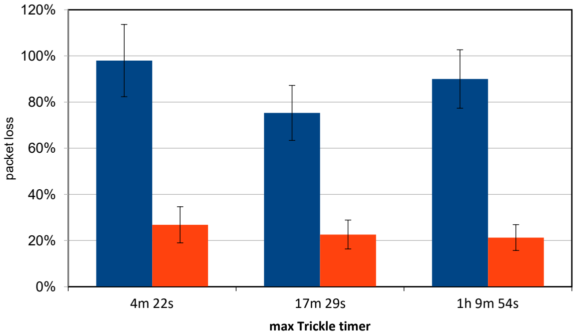

Fig. 8. Average packet loss during attack wrt max Trickle period. The blue dark bars represent the local packet loss, the light red bars represents the global packet loss. 95%-confidence intervals are displayed in error bars.

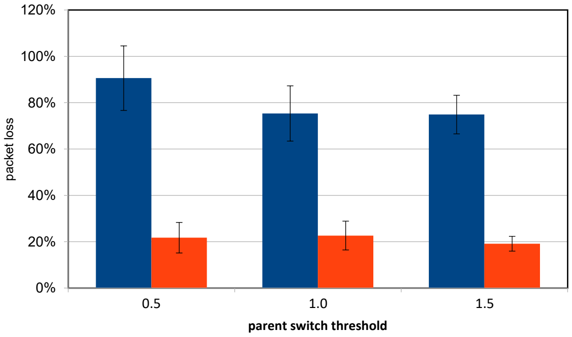

Fig. 9. Average packet loss during attack wrt parent switch threshold.

Figs. 8 and 9 show the average packet loss over the whole attack time (1h) with respect to the max Trickle period, and with respect to the parent switch threshold of MRHOF. These two system parameters are considered to evaluate a possible countermeasure consisting in modifying the behavior of RPL in order to make the network more “static”. Specifically, by increasing the max Trickle timer or the parent switch threshold each node might be less prone to select a new convenient (possibly wormholed) preferred parent. The results, however, do not show clear trends, since the various confidence intervals overlap. This suggests us that a simple configuration-based defence is not feasible in this case.

VI. Countermeasures Analysis for IoT WSANs

Researchers have proposed many countermeasures against the wormhole attack, none of which seems definitive. In this section, we study the applicability of such countermeasures in IoT WSANs. We show that none of these countermeasures seems to be practical, because they are generally too expensive or energy-consumptive for constrained IoT devices. We then argue that the best way to avoid wormhole attacks in IoT WSANs is probably to avoid or detect subsequent attacks, in such a way to nullify the incentive for mounting a wormhole. We discuss how to do this in IEEE 802.15.4 networks.

The wormhole countermeasures in the literature can be roughly divided in those based on specialized hardware [17], [27], network monitoring [18], [28], [29], [30], reciprocity of the wireless medium [31], [32], secure knowledge of the nodes' positions [27], [33], and timeouts between authenticated frames [34]. The countermeasures based on specialized hardware require for example directional antennas (in [17]) or on-chip atomic clocks (in temporal leashes proposed by [27]). Equipping cheap IoT devices with such advanced hardware resources may be unpractical or too expensive. The countermeasures based on network monitoring are more feasible, but they may have a significant cost in terms of resource consumption and traffic overhead. For example, the solutions in [28] and in [29] require nodes to continuously overhear all the neighbors' transmissions. This impedes the nodes to follow effective radio duty-cycle policies, which are often required to obtain acceptable battery lifetimes. Other network monitoring solutions like those in [18] and [30] require nodes to execute probing tests, compute routing statistics, and send periodic reports to a central node. This may constitute a burdensome activity for many resource-constrained nodes. On the other hand, countermeasures based on secure knowledge of the nodes' positions (e.g., geographical leashes in [27]) are feasible and cheap for networks with fixed nodes, whose positions can be statically configured. This is incidentally the case of the victim testbed we used for our experiments. However, applying such solutions in case of mobile nodes may require other hardware, for example GPS receivers, which may be too expensive or too energy-consumptive for many applications. Also, GPS receivers are susceptible to GPS spoofing attacks [35], so positions measured by GPS cannot be considered secure in general. The countermeasures in [31], [32] are based on the physical reciprocity of the wireless medium, that is the property by which the physical characteristics of a wireless channel (e.g., the signal attenuation) are roughly the same in one direction and in the opposite one [31], [32]. Krentz et al. [32] has shown that, in order to be really effective, these solutions require nodes to perform expensive channel tests, by sending and receiving multiple probing frames on separate channels. This again may constitute a burdensome activity for many constrained nodes.

Finally, the countermeasures based on timeouts between authenticated frames appear to be cheap and feasible. Intuitively, these countermeasures aim at precisely measuring the round-trip time between two authenticated frames, in order to detect the additional delay introduced by a wormhole in between the communicating nodes. These countermeasures are partial, since they cannot defend against signal-level wormholes, which introduces very short delays. Even though in theory they could be effective to defend against frame-level wormholes, we experimentally show in the following that such countermeasures can be hardly implemented in practice. Typical IoT devices are generally resource-constrained devices that usually introduce a significant delay in their operations. For this reason, it is unfeasible to enforce a strict timeout between the transmission of two authenticated frames lower than the delay introduced by a wormhole.

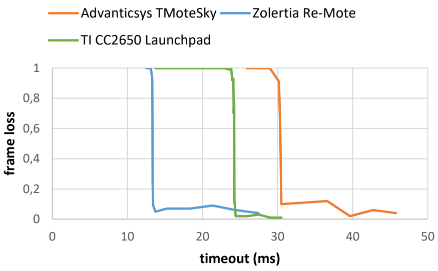

Fig. 10. Frame loss vs sender timeout.

In order to demonstrate this issue, we carried out several experiments considering different IoT-oriented commercially available platforms: Advanticsys TMoteSky[5], Zolertia RE-Mote[6], and Texas Instruments Launchpad[7]. In each experiment a pair of devices of the same platform is programmed as follows using the low-level operating system Contiki OS: one device is programmed to send an authenticated 802.15.4 data frame every second and wait for the relative acknowledgment until a timeout; the other is programmed to receive the data frame and send back an authenticated acknowledgment frame. The data frame is authenticated by means of AES-128 CBC-MAC, which is realized via hardware by the 802.15.4-enabled radio chips. Since in the IEEE 802.15.4 standard acknowledgment frames cannot be authenticated [21], we employed the *enhanced acknowledgment frames* (ENH-ACKs), introduced by the IEEE 802.15.4e amendment [2].

Fig. 10 shows the frame loss, defined as the percentage of acknowledged frames over a total of 100 sent frames, with respect to the timeout adopted by the sender device. It can be seen that, for all considered platforms, the minimum timeout in order to obtain an acceptable frame loss (< 0.1) is in the order of tens of milliseconds. This significant delay is attributable mainly to the software mechanisms necessary to receive and process the challenge frame and prepare and transmit the ENH-ACK. Such a delay is larger than the one that an efficient frame-based wormhole can introduce. With a cheap COTS-based wormhole like the one we implemented, the introduced delay is lower-bounded only by the frame transmission time and the ENH-ACK transmission time between the two endpoints, which are both in the order of hundreds of microseconds, plus the delay introduced by the wormhole link, which is hard to lower bound.

Using the radio chip to produce the authenticated ENH-ACKs, instead of sending them by software, would help to obtain stricter timeouts. Unfortunately, no current 802.15.4-enabled chip is able to send in hardware an authenticated ENH-ACK after a frame reception. We conclude that counter-

[5] https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html

[6] https://github.com/Zolertia/Resources/wiki/RE-Mote

[7] http://www.ti.com/tool/launchxl-cc2650

measures based on strict timeouts are generally hard to apply to IoT-oriented devices currently available in the market.

To summarize what emerges from the research of the past years, there is not a cheap and energy-affordable method to avoid wormhole attacks for typical IoT WSANs. However, it is worth to note that a wormhole attack is not detrimental *per se*, but it is only a way to control a lot of traffic in order to mount other attacks. Typical subsequent attacks are packet eavesdropping and (selective) packet dropping. For these attacks, there are cheap and energy-affordable countermeasures. A simple encryption at the MAC layer avoids packet eavesdropping, and using authenticated acknowledgments makes packet dropping detectable by the victim network. IEEE 802.15.4 standard allows for AES-128 CCM authenticated encryption, which allows for both the above functionalities. AES-128 CCM is also implemented in Contiki [5] and realized via hardware by the 802.15.4-enabled radio chips. These countermeasures avoid or detect subsequent attacks and promise to be cheaper and more effective than employing complex countermeasures against wormhole.

VII. Conclusions

In this paper we presented a wormhole implementation to mount an attack against an IEEE 802.15.4 WSAN. The implementation, composed of a Contiki program and a python code, has been made available opensource on Github. The implementation has been exploited to evaluate the impact of an attack on a real RPL network. The experiments highlighted that the attack can be effective to mount other attacks, e.g., a denial of service. Eventually, we concluded analyzing the feasibility of possible countermeasures proposed in literature. Our analysis highlighted that the proposed methods are difficult to implement in practice, and the most convenient way to avoid a wormhole could be to avoid or detect subsequent attacks, namely traffic eavesdropping and selective packet dropping.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Computer Communications*, vol. 88, no. Supplement C, pp. 1 – 24, 2016.

[3] T. Winter, "RPL: IPv6 routing protocol for low-power and lossy networks," Internet Requests for Comments, RFC 6550, 2012.

[4] O. Gaddour and A. Koubâa, "RPL in a nutshell: A survey," *Computer Networks*, vol. 56, no. 14, pp. 3163–3178, 2012.

[5] K.-F. Krentz, H. Rafiee, and C. Meinel, "6LoWPAN security: Adding compromise resilience to the 802.15.4 security sublayer," in *ASPI'13*. ACM, 2013, pp. 1:1–1:10.

[6] P. Perazzo, C. Vallati, A. Arena, G. Anastasi, and G. Dini, "An implementation and evaluation of the security features of RPL," in *ADHOC-NOW'17*. Springer, 2017, pp. 63–76.

[7] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1294–1312, thirdquarter 2015.

[8] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," in *INFOCOM'03*, vol. 3. IEEE, 2003, pp. 1976–1986 vol.3.

[9] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson, "A security threat analysis for the routing protocol for low-power and lossy networks," Internet Requests for Comments, RFC 7416, 2015.

[10] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in RPL-based Internet of Things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459–473, 2016.

[11] A. Dvir, T. Holczer, and L. Buttyan, "VeRA - version number and rank authentication in RPL," in *MASS'11*. IEEE, 2011, pp. 709–714.

[12] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A study of RPL DODAG version attacks," in *AIMS'14*. Springer, 2014, pp. 92–104.

[13] K. Weekly and K. Pister, "Evaluating sinkhole defense techniques in RPL networks," in *ICNP'12*. IEEE, 2012, pp. 1–6.

[14] K. Iuchi, T. Matsunaga, K. Toyoda, and I. Sasase, "Secure parent node selection scheme in route construction to exclude attacking nodes from RPL network," in *APCC'15*, Oct 2015, pp. 299–303.

[15] H. Perrey, M. Landsmann, O. Ugus, M. Wählisch, and T. C. Schmidt, "TRAIL: Topology authentication in RPL," in *EWSN'16*, 2016, pp. 59–64.

[16] P. Perazzo, C. Vallati, G. Anastasi, and G. Dini, "DIO suppression attack against routing in the Internet of Things," *IEEE Communications Letters*, vol. PP, no. 99, pp. 1–1, 2017.

[17] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks." in *NDSS'04*, 2004, pp. 241–245.

[18] N. Song, L. Qian, and X. Li, "Wormhole attacks detection in wireless ad hoc networks: a statistical analysis approach," in *IPDPS'05*, 2005, pp. 8 pp.–.

[19] B. Awerbuch, R. Curtmola, D. Holmer, H. Rubens, and C. Nita-Rotaru, "On the survivability of routing protocols in ad hoc wireless networks," in *SECURECOMM'05*, 2005, pp. 327–338.

[20] M. Khabbazian, H. Mercier, and V. K. Bhargava, "Severity analysis and countermeasure for the wormhole attack in wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 2, pp. 736–745, Feb 2009.

[21] "IEEE standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[22] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle algorithm," Internet Requests for Comments, RFC 6206, 2011.

[23] P. L. O. Gnawali, "The Minimum Rank with Hysteresis Objective Function," Internet Requests for Comments, RFC 6719, 2011.

[24] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, Jul. 2005.

[25] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," in *WiSe'04*, 2004, pp. 32–42.

[26] C. Vallati, E. Ancillotti, R. Bruno, E. Mingozzi, and G. Anastasi, "Interplay of link quality estimation and RPL performance: An experimental study," in *PE-WASUN'16*. New York, NY, USA: ACM, 2016, pp. 83–90.

[27] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, Feb 2006.

[28] I. Khalil, S. Bagchi, and N. B. Shroff, "LITEWORP: a lightweight countermeasure for the wormhole attack in multihop wireless networks," in *DSN'05*, 2005, pp. 612–621.

[29] S. Choi, D. y. Kim, D. h. Lee, and J. i. Jung, "WAP: Wormhole attack prevention algorithm in mobile ad hoc networks," in *SUTC'08*. IEEE, 2008, pp. 343–348.

[30] I. Khalil, S. Bagchi, and N. B. Shroff, "MobiWorp: Mitigation of the wormhole attack in mobile multihop wireless networks," *Ad Hoc Networks*, vol. 6, no. 3, pp. 344 – 362, 2008.

[31] S. Jain, T. Ta, and J. S. Baras, "Wormhole detection using channel characteristics," in *ICC'12*, 2012, pp. 6699–6704.

[32] K.-F. Krentz and G. Wunder, "6LoWPAN security: Avoiding hidden wormholes using channel reciprocity," in *TrustED'14*. ACM, 2014, pp. 13–22.

[33] R. Poovendran and L. Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *Wireless Networks*, vol. 13, no. 1, pp. 27–59, Feb 2007.

[34] J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos, "TrueLink: A practical countermeasure to the wormhole attack in wireless networks," in *ICMP'06*. IEEE, 2006, pp. 75–84.

[35] P. Perazzo, L. Taponecco, A. A. D'Amico, and G. Dini, "Secure positioning in wireless sensor networks through enlargement miscontrol detection," *ACM Transactions on Sensor Networks*, vol. 12, no. 4, pp. 27:1–27:32, Sep. 2016.