Journal of ELECTRICAL ENGINEERING, VOL 73(2022), NO4, 267–275

sciendo

PAPERS

A potential flooding version number attack against RPL based IOT networks

Mehdi Rouissat[1,2] **Mohammed Belkheir**[1,3] **Hichem Sid Ahmed Belkhira**[1,4]

Routing protocol for low power and lossy networks (RPL) has been proposed for power, memory, and processing constrained devices. Owing to their constrained, RPL-based networks are exposed to a wide range of security attacks that mainly include control message tampering. In this paper we propose and study a modified version number attack, based on flooding the network by falsified incremented version numbers. The obtained results show that the modified attack led to an immense increase in the overhead, 1426%, compared with the attack-free case, and an increase of 182 % in the total energy consumption. When it comes to PDR a degradation to 4.7% has been recorded, affecting the reliability of the network. On the other hand, the latency also showed an increase from 0.24 s in the attack-free case to 0.89 s, which is mainly due to the high congestion created by the attack.

K e y w o r d s: RPL, security, version number, flooding, Cooja

1 Introduction

Due to the constraints of internet of things (IoT) devices, routing is one of the most researched fields nowadays. Routing protocol for low power and lossy networks (RPL), introduced by internet engineering task force (IETF) [1], is adequate and widely used with the power, memory, and processing constrained devices, where it became a routing standard of IoT networks, due its routing service proficiency. However, security mechanism is integrated in RPL as an optional part [1,2].

RPL is a distance-vector proactive routing protocol based on IPV6, dedicated for low-power and lossy wireless networks, [1]. It arranges and organizes the nodes in the network into an inverted treelike topology known as destination oriented directed acyclic graphs (DODAGs) [3]: it means a hierarchical arborescence of the connected nodes without loops, where the data traffic is rooted toward a single destination named the sink or the root node, [3]. The DODAG construction is based on the used objective function (OF), which defines rules and main configurations, such as the used routing metrics, how the rank of the nodes is calculated and how to select the preferred parent in the DODAG. The rank value determines the position of a node in the DODAG compared to the roots. It is calculated based on the routing metrics defined by the OF, [2].

Several academic and industrial research workers whiteness that RPL offers an effective routing solution for a wide range of IoT networks, [4]. Nevertheless, security remains a particularly challenging concern in RPL based networks, not only because of the node limitation in terms of resources but because of their open, mobile and untended deployment.

A set of security tools has been already considered in basic RPL. However, in most cases they can be useful and effective only against external attacks, in other words they are generally unable to protect the network from internal attacks as anomalous comportment of nodes that leads to an egoistic conduct and control message tampering attacks. Therefore, RPL can be threatened by a wide range of security attacks [5].

Some of the existing attacks aim to waste the resources of the motes, by compelling legitimate nodes performing excessive processing, to exhaust their resources. [6]. Version number attack is a well-known attack against the resources, it leads to a repetitive recreation of the network by provoking global repairs, this is done by illegitimately increasing the version number by the malicious node.

In this paper we propose and study a modified version number attack (VNA), in which the attacker node floods the network with incremented falsified version numbers, in a continuous way. In the next sections the impact of this attack in terms of control overhead, energy, latency and PDR is introduced. We introduce the RPL protocol, present the well-known VNA, and the simulation environment together with a proposed attack, and we highlight its impact.

2 RPL protocol overview

RPL is a distance-vector routing protocol based on IPv6. The routing scheme in RPL is based on a combined

[1] Centre Universitaire Nour Bachir El-Bayadh, 32000, El-Bayadh, Algeria, [2] STIC Laboratory, University Aboubekr Belkaid, Tlemcen, mehdi.m.rouissat@gmail.com [3] LIMA Laboratory, Univeristy Center Nour Bachir, El-Bayadh, Algeria, belkheir_m@yahoo.fr, [4] LTTNS Labortory Djillali Liabes University, Sidi Bel Abbes, Algeria, belkhira.hichem@gmail.com

https://doi.org/10.2478/jee-2022-0035, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

mesh-tree topology, called DODAG [7]. The DODAG graph is built in a step-by-step manner, using control messages.

2.1 RPL control messages

Contiki RPL relies mainly on three types of ICMPv6 control messages to create and manage the network topology and routing information:

a) DODAG information object (DIO) message: This message is advertised to construct, maintain a DODAG, and build upward routes from other nodes to the root, [8]. It carries important updated configuration parameters as: RPLs instance ID, DODAGs ID, version number, RPLs mode of operation, the rank of sender node, and other necessary maintenance parameters, [1.8]. Furthermore, DIO messages are sent as response to a received DODAG information solicitation (DIS) message, [9].

b) DODAG information solicitation (DIS) message: it is an upward ICMP control message used by a new node willing to join an existing network to solicit a DIO message from neighbors. It is sent by a node when no DIO message is received within a time interval (5s is the default value in RPL Contiki).

c) Destination advertisement object (DAO) message: while DIO and DIS messages are used to create and maintain the upward routes, DAO messages allow establishing downward routes, where each node advertises to its parent the set of nodes in its own sub-DODAG that selected it as preferred parent [9]. The DAO receiver node may optionally send a DAO acknowledgement message, if required, which is not the case of RPL Contiki.

The convergence is achieved upon each node has a path to the root. After the convergence, DIO messages are periodically exchanged between nodes for the network maintenance purpose, [10].

2.2 Trickle timer algorithm

In the normal operation of RPL, DIO messages are periodically exchanged to maintain the network topology. This organized periodic behavior allows nodes to avoid unnecessary control traffic and allows an energy efficient way that preserves the nodes limited resources.

RPL implements a trickle algorithm to dynamically regulates the sending of DIO messages, [8],[11] which strikes a tradeoff between reactivity to topology changes and energy management. Thus, the trickle timer ensures that DIO messages are sent in an aggressive manner if the network is in unstable state, and instead rebroadcast the DIOs at a progressive slow frequency when the network remains stable. Each node maintains a DIO counter with a pre-defined threshold and a timer with trickle. DIO message is sent upon the period of the trickle timer is reached or when a received DIO message carries updated RPL configuration parameters [12]. The two configuration parameters related to the trickle timer are carried in DODAG Configuration option of DIO messages: including a value for defining the minimum trickle time (minimum sending time interval), and another value for defining the maximum trickle time (maximum sending interval). In Contiki, the initial and the minimum trickle time is

$$T_{\min} = 2^{n_1}, \tag{1}$$

where n_1 is the RPL DIO INTERVAL MIN, the default value in Contiki is 12, thus $T_{\min} = 4096\,\text{ms}$ what is approximately 4 s.

Each time the received DIO message does not contain new configuration parameters, the counter increases. Thus, the trickle timer will be doubled until reaching a maximum value defined by

$$T_{\max} = T_{\min} 2^{n_2}, \tag{2}$$

where n_2 is RPL DIO INTERVAL DOUBLINGS, the default value in Contiki is 8, thus, $T_{\max} \approx 17.5$ minutes.

The DIO counter and the trickle timer return back to their initial settings whenever a node receives a DIO with an updated configuration or detects an event indicating an inconsistency in the topology, such events need active maintenance and DIO messages should carry the fresh corresponding information. RPL considers the following cases as inconsistencies:

- Detection of a routing loop;
- A node joins a DODAG;
- When a node changes its rank, allowing it to move toward the DODAG root;
- Inconsistency when forwarding a data packet;
- Leaving the current DODAG;
- Triggering topology repair.

The trickle timer algorithm brings an advantage to regulate the DIO messages broadcasting, since the transmission period increases in case of a stable topology, and promptly decreases to allow the fast dissemination of the updated configuration parameters overall the network [13].

In general, the trickle timer setting is directly related to the two parameters $T_{\min}$ and $T_{\max}$, that allow choosing values that meet the topology evolution need. A higher period values leads to energy efficiency, while the reactiveness becomes low. In contrary to a lower period, where the reactiveness is improved to the detriment of the node's lifetime.

2.3 Self-healing mechanisms of RPL

RPL Contiki includes two mechanisms of self-healing:

- The global repair

This mechanism is only triggered by the root upon perceiving several inconsistencies, [14]. Thus, the root increments the DODAG version number and broadcasts it to all the nodes. Nodes reply to this message by ignoring their current version number (VN), resetting their trickle timers and start the process to join the DODAG again.

- The local repair

It is triggered by a node and affects only its sub-DODAG. A node losing its connection with its parent

may initiate a local repair mechanism to rapidly allow the network convergence, by resetting its trickle timer and sending an updated DIO message, [9]. It can also happen when a node would poison its sub-DODAG by sending an INFINITE_RANK value [15], that consequently induces the detaching of the childs nodes from the present node and triggering a new parent selection process.

3 Traditional version number attack

The version number is one among the main parameters exchanged via DIO control messages, since it talks about the current state of the DODAG. As already mentioned, it is incremented by the root upon a change in the DODAG occurs or must occur to tell other nodes that their routing tables remain obsolete, [16]. Each receiving node compares its current version number to the received one, in the case of receiving higher value; the node must participate in the global repair by resetting its trickle timer and initiating at its level the joining process to the DODAG. Despite the advantage brought to reconstruct a new DODAG, the global repair mechanism could also be quite costly in terms of resources. The RPL protocol specifies that only the root is responsible for initiating the global repair by increasing the value of the VN whenever it is needed, to guarantee the topology steadiness, to ensure an optimized, an updated and a healthy DODAG, and to deal with some inconsistency situations, [17]. Therefore, a stable RPL topology stipulates that all the nodes must hold the same version number, where they quickly propagate the new advertised V_{N} by resetting their Trickle timer. This mechanism is repeated until all the network's nodes update their routing state and have the same V_{N}, [18].

However, a malicious node can take advantage of this mechanism to attack the network, by intentionally broadcasts illusive information regarding the VN associated with the topology, to often triggering an illicit global repair and force the other nodes to reset their timers and rebuild the DODAG from scratch. As result, the propagation of extra control messages will spend more energy and lead to an instability in the network as well as data loss, [19].

Version number attack (VNA) has been a topic of research for several studies. In [20] the authors analyzed and studied the effect of multiple VNAs in RPL networks, their results showed that increasing the number of attackers affects only the PDR results. In [21-23] the VNA has been studied in its traditional scheme.

Based on our research questions, the contributions of this work can be listed as follows:

- We study the traditional VNA in a particular and detailed way, where the main reasons of trickle timer resets are listed, and more parameters are taken into consideration, such as: convergence of the network
- We analyze the effect of flooding VNA, which has not been done previously,
- We investigate the performance of the network under attack based on different DIO messages periods,
- We analyze the effect of the flooding VNA attack in terms of generated DAO, forwarded DAO messages and no-path DAO, considering the related work, none of the studies focused on this detail.

4 Simulation environment

We performed our simulations using Contiki 3.0, which is an open source, multi-task operating system for Low Power and Lossy Networks-based IoT systems, [24]. This O.S has been chosen because it provides the implementations of the standardized protocols stack (*ie* , IEEE 802.15.4 physique and MAC layers, 6LoWPAN adaptation layer, RPL, IPv6, UDP transport layer and CoAP) which were proposed by the IEEE and the IETF for LLNs [25,26]. On the other hand, Contiki simulator *ie* Cooja, is a widely used and reliable network simulator, it provides emulations of well known WSN motes (*eg* , Z1 motes, Tmote Sky, MicaZ, MSP430-based motes) [27], it also provides several tools to evaluate the RPL protocol, such as power trace. The different simulation parameters are summarized in Tab. 1.

Table 1. Simulation parameters

Parameter	Values
Network layer Protocol	RPL
Operating System	Contiki 3.0.
Simulator	Contiki Cooja
Emulated nodes	Z1
MAC layer Protocol	IEEE 802.15.4
Radio model	UDGM
Simulation area	200 m × 200 m
Simulation time	15 minutes
Data transmission	1 Packet / 60s
Objective function	MRHOF

In our simulations, nodes are Z1 motes running RPL-UDP application. UDP clients running in nodes send UDP packets containing a simple Hello message to the root, every minute. A Perl script is used to extract and calculate data regarding latency and PDR metrics. Table 2 summarizes the main features of the used Z1 nodes. In this study, we consider a single malicious node, the studied topology is shown in Fig. 1, it consists of 1 sink node (node "1"), 13 fair nodes and 1 malicious node (node "15"). The simulations are timed out to end in 15 minutes for all the conducted simulations.

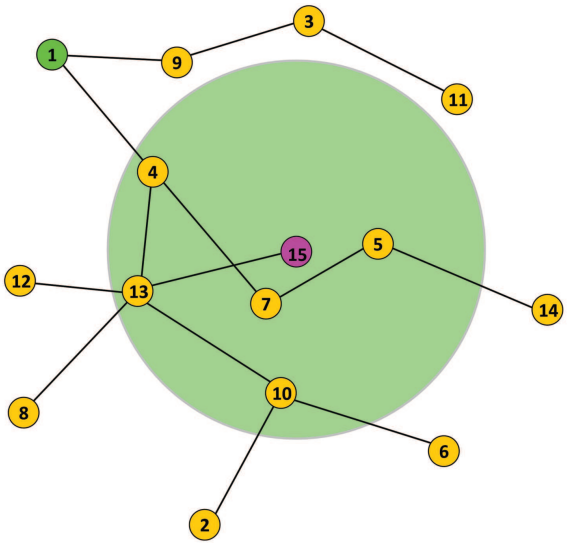

Fig. 1. Studied topology

Table 2. Hardware specification of the used Z1 nodes, [28]

Parameter	Value
CPU idle current	0.426 mA
Current consumption in TX mode	17.4 mA
Current consumption in RX mode	18.8 mA
CPU power down current	0.020 mA
Maximum RAM size	8 kB
Maximum ROM size	92 kB
RTIMER_SECOND	32768 ticks/s

To highpoint the effect of the version number attack, simulations are firstly run without any attack in the studied topology (attack-free topology), where obtained results are taken as reference.

The impact of the attack is evaluated basing on :

- Network Performance Metrics: control message overhead, latency, and data packets delivery ratio,
- Resource Requirement Statistics: total consumed energy.

5 Simulation results and discussion

In this section, we show the effect of the VNA on the performance of the network in terms of several indicators. Then, we present and study the potential flooding VNA and its effects on the network performances.

5.1 Traditional version number attack

To emulate the version number attack, we modified the RPL "udp-client.c" file of the malicious node, where it launches an illegitimate global repair by incrementing the advertised version number whenever it sends a scheduled DIO message, with respect to the default RPL Contiki trickle timer, discussed in section 3. The root on the other hand calls, for another time, for a global repair each time it receives the inconsistent version number, consequently the entire DODAG is recreated for a second time. By continuing to instigate the incrementing of version attack, the malicious behavior keeps the nodes busy in reorganizing the DODAG again and again.

In RPL Contiki, the default version number value starts from 240, to reach 255, and then starts over from 1 to reach again 255 and so on. The total number of participations in the global repairs is

$$R_T = V_{\mathrm{N}} + 15 + 255(n-1), \tag{3}$$

where V_{N} - is the version number advertised by a node after 15 minutes of simulation, value 15 comes from (255 -240), n - presents how many times the VN reached the pick value (255). Table 3 depicts the values of the version numbers declared by each node as well as the total participation in the global repair by each node at the end of the simulation. The malicious node is always ahead of the others, declaring a fake falsified incremented version value (87), due to the malicious behaviour. On the

Table 3. V_{N} changes in traditional VNA

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V_{N}	87	85	85	86	86	85	86	86	85	86	85	86	86	85	87
Total	102	100	100	101	101	100	101	101	100	101	100	101	101	100	102

Table 4. Control overhead for VNA

	Sent messages			
	Generated		Forwarded	
	DIO	DAO	DAO	Total
Attack-free	206	109	176	491
VNA	1945	926	244	3115

other hand, the sink node, is also declaring a legitim incremented value to lunch the global repairs, based on the detected inconsistency. For this scenario, the VN value reached the pick just one time, which gives an average of 101 global repairs lunched in the network.

Impact of VNA on the control overhead

The results related to the RPL control message overhead are shown in Tab. 4. It can be noticed that VNA generates a superfluous amount of exchanged control messages in comparison with the attack-free case, where an increase in the total overhead of 534% is recorded.

The number of sent DIO messages is directly related to how many times the nodes reset their trickle timers. In Contiki, the factors that lead a node to reset its own timer are:

- After joining a new DAG
- If the root receives an increased V_N
- If an old V_N is received
- After changing the preferred parent
- After launching a local repair

During the simulated VNA, a total of 3236 resets has been launched by the different nodes. Table. 5 lists the different factors and their contributions to the total recorded resets.

Table 5. Trickle timer resets during the VNA

Factor	Number of resets
joining New dag	14
inconsistent DIO version number	65 (by root only)
old version received	1060
infinite rank received	1572
Changed preferred parent	514
local repair	11
Total	3236

It can be noticed in Tab. 5. that the major factors lead the nodes to resetting their trickle timers are:

- The case where an infinite rank value is received, 49% of the total. Advertising an "Infinite_Rank" is the way a node poisons its routes, which is a step in the global repair process.
- The case where a node receives an old version number from its neighbors, 33% of the total. Since the nodes are incrementing continuingly their advertised VN, the received VNs from their children are more likely lower than their owns.

This recurrent reset of the trickle timer by each node elicits an upsurge of control messages, thus, this abrupt increase led to an unstable topology, by making communication channels unavailable and increasing the global energy consumption of the network, whereby the network lifetime is reduced.

The number of resets recorded by the malicious node is 484, (480 are due to the reception of an old VN, 3 are due to parent change process, and 1 after joining the DAG). The VNA is based on the number of DIO sent by the malicious node, which is related to how many times it resets its trickle timer. Thus, more neighbors the malicious node has, more old version values will be received, consequently, more trickle timer resets and more frequent DIO with falsified version values is advertised.

Impact of VNA on the consumed energy

When targeting LLNs, where nodes have battery constraints, the energy consumption of the nodes is an important performance indicator. The total amount of the consumed energy is calculated by adding up the energy consumed in four modes, which are: energy consumed in the CPU mode, energy consumed in the LPM mode, energy consumed in the TX mode and energy consumed in the RX mode.

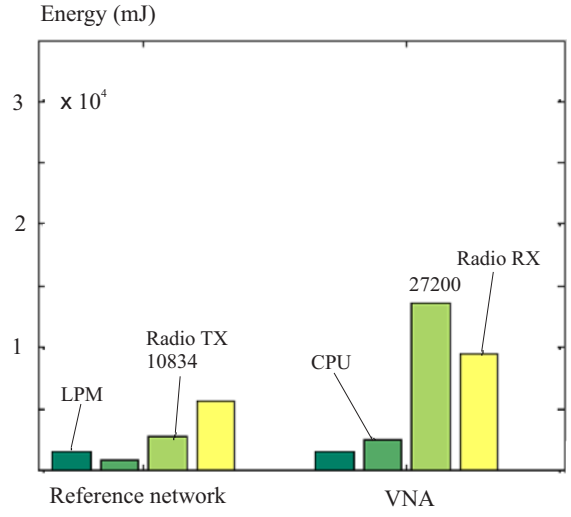

Fig. 2. Total consummed energy

In this regard, it can be observed in Fig. 2. that the version number attack affected considerably the total energy consumption of the nodes, where it jumped from 10.8 J in the reference network to 27.2 J in the case of the attack, which presents an increase of 252%. The consumed energy in the CPU mode is an important parameter to evaluate and judge how busy the network was, in the reference network it shows a consumption of 824.98 mJ, whereas, under attack the CPU shows a higher consumption of 2522.06 mJ. The consumed energy in the TX

Table 6. VN changes in flooding VNA

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V_N	102	101	96	103	103	101	101	101	102	103	97	101	103	101	103
Total	372	371	366	373	373	371	371	371	372	373	367	371	373	371	373

Table 7. Control overhead for the three scenarios

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V_N	102	101	96	103	103	101	101	101	102	103	97	101	103	101	103
Total	372	371	366	373	373	371	371	371	372	373	367	371	373	371	373

modes also shows an important increase from 3j to 16 J. This difference can be explained by the high overhead in terms of DIOs and DAOs that has been translated into an increase in the total consumed energy, in particular the TX and RX modes. These obtained results demonstrate the damaging impact of the attack on the energy, which considerably affects the nodes and the network lifetime.

5.2 Flooding VNA

The damaging effect of VNA is directly related to the number of sent DIO messages. Earlier, we studied VNA with respect to the default RPL Contiki DIO trickle timer parameters, which are defined by the sink and shared to all nodes through DIO messages. In this section, we propose a flooding VNA based on modified trickle timer parameters, in order to make the attack more damaging. The malicious node sent its falsified DIO messages with respect to the modified trickle parameters. In turn, legitimate nodes will be forced to flood the network by broadcasting DIO messages holding falsified incremented version numbers. The aim of this combined attack is to extremely affect the main critical network characteristics as: the control overhead, the latency; the energy consumption, and the reliability. Such situation produces congestions of the communication channel, not just of a specific part, but of all the network since every victim node is forwarding the falsified configuration parameters.

A malicious node may use any rates for flooding VNA. In our work, the default values defining the minimum and the maximum DIO intervals had been tempered, to be 10 and 1 for "RPL_DIO_INTERVAL_MIN" and "RPL_DIO_INTERVAL_DOUBLINGS", respectively.

Thus, the initial and the minimum interval of the trickle timer, is $T_{\min} = 2^{10} = 1024\text{ms} \approx 1\,\text{s}$.

On the other hand, the modified maximum interval of the trickle timer, is $T_{\max} = 1024 \times 2^1 = 2048\,\text{ms} \approx 2\,\text{s}$. q

The malicious node not only sets these modified values for itself, but also advertises them to its neighbors, in its DIO messages.

Table 6 depicts the values of the version numbers declared by different nodes and their total participation in the global repairs, after 15 minutes of simulation. In this scenario, the version number value reached the pick (255) twice, which gave an important number of global repairs lunched in the network, an average of 370 instead of 101 global repairs recorded in the traditional VNA.

Impact of Flooding VNA on the control overhead

Table 7. shows the exchanged control messages in the attack-free case, in the VNA and in the flooding VNA. The total overhead shows an incredible increase of 1426% compared with the attack-free case. The DIO messages present the big part of the increase with 3314%, giving an average of 47 DIO messages sent by node per minute. The underlying reason for such immense increase is the frequent reset of the trickle timer of all the nodes, in particular the malicious node, which presents the source of the incremented V_N and the falsified trickle timer parameters.

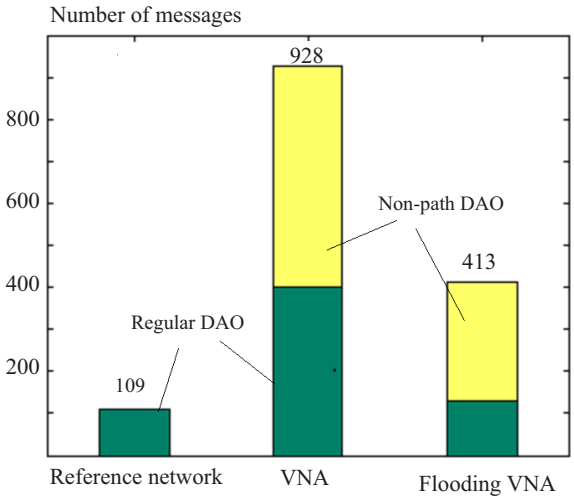

Fig. 3. Regular DAO and no-path DAO for the three case

A DAO message can be either a regular DAO used to build downward routes, or a no-path DAO used to poison routes. Also, it can be either generated by the node or generated by children nodes and being forwarded

Table 8. Trickle timer resets for the VNA and flooding VNA

Reasons	Number of resets	
	VNA	Flooding VNA
Joining new dag	14	14
Inconsistent DIO VN	65	202 (+ 137)
Old version received	1060	3064 (+ 2004)
Infinite rank received	1572	4594 (+ 3022)
Changed preferred parent	514	268 (- 246)
Local repair	11	8 (- 3)
All	3236	8150 (+ 4914)

Table 9. Network convergence time for the three cases

	Network setup time (s)
Reference	19.92
VNA	22.29
Flooding VNA	26.62

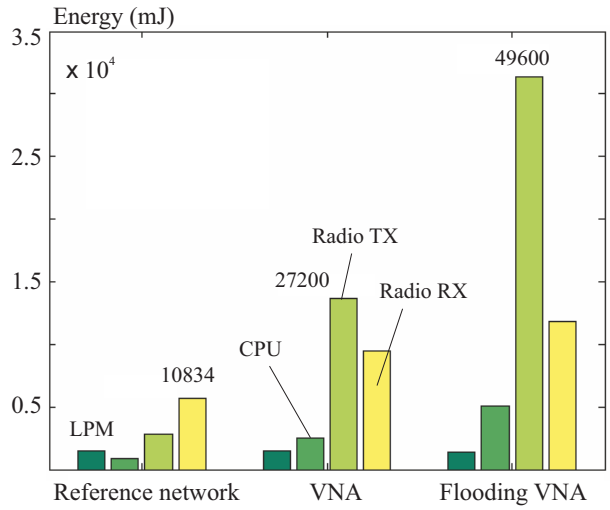

Fig. 4. Total consumed energy for the three case

upwards. Referring to Tab. 7, interestingly, we see that the DAO overhead decreased, from 926 in the case of VNA to 413 in the case of flooding VNA. Figure 3 shows the type and the number of exchanged DAO messages in the three cases.

No-Path DAO messages are a reliable parameter to judge the stability of a network topology. Referring to Fig. 3, we notice that the number of both regular and no-path DAO decreased. The underlying reason for such behavior is the flooding effects, which partially kept the nodes over busy sending DIO messages, which lead to losing control messages.

Table 8 details the different factors that lead the nodes to reset their trickle timers. As it can be seen, the total resets shows an increase of 151.8%, where the case of receiving infinite rank is the major factor, with 77% of the total. This increase in the number of resets forced the nodes to send DIO messages frequently, which drastically increases control packet overhead in the network. Regarding the malicious node, it records 1084 resets, where only one is due to joining the DAG, the remaining are all due to the case of receiving a lower V_N since it is always ahead compared to its neighborhood.

A noticeable increase in the overhead affects also the convergence of the network. Table 9 shows that the network setup time increases from 19.92 s in the attack-free network to 26.62 s in the case of flooding VNA, an increase of 33.6%. This is due to the high traffic caused by the attack, the collisions and the congestion in the topology during the flooding attack.

Impact of Flooding VNA on the consumed energy

The total consumed energy by a node depends on various parameters related with different layers of the protocol stack. Beside the control overhead, it also depends on the data traffic rate that IoT applications generate. In this study, one packet per 60 seconds was considered for all simulations.

Figure 4 depicts the consumed energy in terms of CPU, LPM, TR and RX for the three cases. The results show that the total consumed energy has noticeably increased from 27.2j in the case of VNA to 49.6 j in the case of flooding VNA, which presents an immense increase of 182%. These obtained results prove how damaging the VNA can be when combined to Flooding behavior.

Packet delivery ratio

The PDR (packet delivery ratio) is the ratio of the total received data packets by the root node to the total number of sent data packets by the network's nodes, we take the average PDR of all the packets received successfully at the root node [29]. It is an important parameter used to evaluate the reliability of the network.

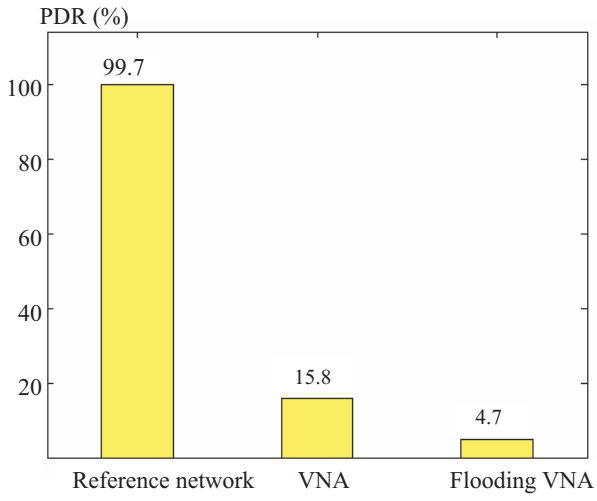

Fig. 5. The obtained PDR for the three case

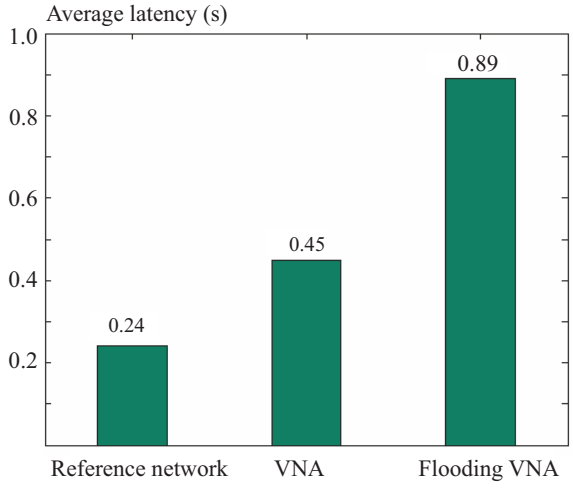

Fig. 6. The obtained latency for the three case

The obtained results regarding the PDR are shown in Fig. 5. The VNA causes detrimental effects on the PDR, which drops down to 15.8% for VNA, and dramatically drops to 4.7% for Flooding VNA. These results show the serious effect of the VNA when combined to flooding, where 95.3% of application packets cannot be delivered successfully. The attack could be more damaging when running the flooding VNA under a lower flooding interval.

The significant decrease in the data packet delivery ratio can be mainly justified by:

- Packets collision, due to the high control overhead,
- The congestion experienced by the forwarder nodes, due to the high overhead at the forwarder nodes under the effect of the attack,
- The unavailability of routes, packets are dropped since no routes exist before going down to MAC layer, due to the continues rebuilding of the DAG.

Latency

The latency presents the average time taken for a given number of data packet transmission by the network's nodes to be received by the root node [30]. In this study, only successful transmission are considered in the calculation, dropped and lost packets are not. The latency is also an important parameter to evaluate the quality of the routes and the reliability of the network.

The performance of the network in terms of latency is shown in Fig.6. In comparison to the baseline results, the latency has been adversely affected by VNA, where it increased to 0.45 s, a ratio of 87.5%. On the other hand, flooding VNA immensely increased the latency to 0.89 s, an increase of 271%. This again can be attributed to the collisions and the high congestion induced by the high overhead.

6 CONCLUSIONS

In this paper, a new flooding version number attack against RPL-based networks has been proposed, and analyzed regarding the network performance in terms of power consumption, overhead, packet delivery ratio and latency, under different scenarios and operating conditions. In the future, we plan to study the effect of the proposed attack under various time intervals, across different topologies. Future work will also address the countermeasure against this attack and testing it to evaluate its performance in real deployment.

References

[1] T. Winter *et al*, "RPL: IPv6 Routing Protocol for Low Power and Lossy Networks", *Internet Requests for Comments*, RFC Editor, RFC 6550, March, https: //ietf.org/doc/rfc6550/, 2012.

[2] A. Raoof, A. Matrawy, and C. Lung, "Secure Routing in IoT: Evaluation of RPL's Secure Mode under Attacks", *IEEE Global Communications Conference*, (GLOBECOM), pp. 1-6, 2019.

[3] J. Granjal, E. Monteiro, and J. Silva, "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues", *IEEE Comm. Surveys and Tutorials*, vol. 17, no. 3, pp. 1294-1312, doi :10.1109/COMST..2388550, 2015.

[4] S. Y. Hashemi and F. S. Aliee, "Dynamic and comprehensive trust model for IoT and its integration into RPL", *The Journal of Supercomputing*, vol. 75, no. 7, pp 3555–3584, doi: 10.1007/s11227-018-2700-3, 2019.

[5] F. Ahmed and Y. Ko, "Mitigation of black hole attacks in Routing Protocol for Low Power and Lossy Networks", *Security and Communication Networks*, vol. 9, pp. 5143–5154, doi:10.1002/sec.1684, 2016.

[6] S. M. H. Mirshahjafari and B. S. Ghahfarokhi, "Sinkhole+CloneID: A hybrid attack on RPL performance and detection method", *Information Security Journal: A Global Perspective*, vol. 28, pp. 107-119, doi: 10.1080/19393555.1658829, 2019.

[7] A. Musaddiq, Y. B. Zikria *et al*, "Routing protocol for Low-Power and Lossy Networks for heterogeneous traffic network", *J Wireless Com Network*, vol. 21, doi: 10.1186/s13638-020-1645-4, 2020.

[8] U. Shafique, A. Khan, A. Rehman, *et al*, "Detection of rank attack in routing protocol for Low Power and Lossy Networks", *Ann.Telecommun.*, vol. 73, pp. 429–438, doi: 10.1007/s12243-018 -0645-4, 2018.

[9] A. Raoof, A. Matrawy, and C.-H. Lung, "Routing Attacks and Mitigation Methods for RPL-Based Internet of Things", *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1582-1606, Secondquarter, doi:10.1109/COMST.2018.288 5894, 2019.

[10] H. Pereira, G. L. Moritz, R. D. Souza, A. Munaretto, and M. Fonseca, "Increased Network Lifetime and Load Balancing Based on Network Interface Average Power Metric for RPL", *IEEE Access*, vol. 8, pp. 48686-48696, doi:10.1109/ACCESS.29 79834, 2020.

[11] B. V. P. Levis, T. H. Clausen, O. Gnawali, J. Hui, and J. Ko, The Trickle Algorithm, *RFC 6206*, March, https://www.rfc -editor.org/info/rfc6206, 2011.

[12] H. Lamaazi and N. Benamar, "RPL Enhancement Based FL-Trickle: A Novel Flexible Trickle Algorithm for Low Power and Lossy Networks", *Wireless Pers. Commun.*, 110, 1403–1428, doi.org/10. 1007/s11277-019-06792-2, 2020.

[13] H. Kharrufa, H. A. A. Al-Kashoash, and A. H. Kemp, "RPL-Based Routing Protocols in IoT Applications: A Review", *IEEE Sensors Journal*, vol. 19, no. 15, pp. 5952-5967, doi: 10.1109/JSEN.29 10881, 2019.

[14] A. Verma and V. Ranga, "Mitigation of DIS flooding attacks in RPL-based 6LoWPAN networks, *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, pp. e3802, doi:10.1002/ett.3802, 2020.

[15] M. Sain, Y. J. Kang, and H. J. Lee, "Survey on security in Internet of Things: State of the art and challenges", *19th International Conference on Advanced Communication Technology (ICACT)*, pp. 699–704, 2017.

[16] A. Verma and V. Ranga, "Security of RPL Based 6LoWPAN Networks on the Internet of Things: A Review", *IEEE Sensors Journal*, vol. 20, no. 11, pp. 5666-5690, doi: 10.1109/JSEN.2973677, 2020.

[17] R. Sahay *et al*, "A holistic framework for prediction of routing attacks in IoT-LLNs", *The Journal of Supercomputing*, vol. 78, pp 1409-1433, doi: 10.1007/s11227-0211-03922-1, 2021.

[18] A. Mayzaud *et al*, "A Taxonomy of Attacks in RPL-based Internet of Things", *Int. J. Netw. Secur.*, vol. 18, pp 459-473, doi: 10.6633/IJNS.05.18(3).07, 2016.

[19] P. O. Kamgueu *et al*, "Survey on RPL enhancements: A focus on topology, security and mobility", *Computer Communications*, vol. 120, pp. 10-21, doi: 10.1016/j.comcom.02.011, 2018.

[20] A. Aris and S. F. Oktug, "Analysis of the RPL Version Number Attack with Multiple Attackers", *International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1-8, doi: 10.1109/CyberSA49311..9139695, 2020.

[21] P. P. Ioulianou *et al*, "Battery Drain Denial-of-Service Attacks and Defenses in the Internet of Things", *Journal of Telecommunications and Information Technology*, no. 2, pp. 37-45., doi: 10.26636/jtit.131919, 2019.

[22] A. Aris, S. F. Oktug, and S. B. O. Yalcin, "RPL version number attacks: In-depth study", *NOMS - IEEE/IFIP Network Operations and Management Symposium*, pp. 776-779, doi: 10.1109/NOMS.7502897, 2016.

[23] A. Mayzaud, R. Badonel, and I. Chrissment "A Distributed Monitoring Strategy for Detecting Version Number Attacks in RPL-Based Networks", *IEEE Transactions on Network and Service Management*, vol. 14, pp. 472-486, doi: 10.1109/TNSM.2705290, 2017.

[24] Contiki rpl, https: //github.com/Contiki-os/contiki/blob /release-3-0/, 2021,.

[25] M. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things", *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1389–1406, Third, doi: 10.1109/SURV.2012.111412.00158, 2013.

[26] I. Wadhaj *et al*, "Mitigation Mechanisms Against the DAO Attack on the Routing Protocol for Low Power and Lossy Networks (RPL)" *IEEE Access*, vol. 8, pp. 43665-43675, doi: 10.1109/ACCESS.2977476, 2020.

[27] F. Osterlind, "A Sensor Network Simulator for the Contiki OS", *Technical Report, Tech. Rep*, May, 2006.

[28] S. L. Advancare, *Zolertia: Z1 datasheet*, http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf, 2010.

[29] M. A. Nasab, S. Shamshirband, A. Chronopoulos, A. Mosavi, and N. Nabipour, "Energy-Efficient Method for Wireless Sensor Networks Low-Power Radio Operation in Internet of Things", *Electronics*, vol. 9, no. 2, pp. 320–333, Feb., doi:10.3390/electronics9020320, 2020.

[30] S. S. Solapure and H. H. Kenchannavar, "Design and analysis of RPL objective functions using variant routing metrics for IoT applications", *Wireless Networks*, vol. 26, no. 3, doi: 10.1007/s11276-020-02348-6.

Received 28 Junly 2022

Mehdi Rouissat received his PhD degree in 2013 from the university of Tlemcen, Algeria in the topic of Wireless optics. Currently, he is a lecturer researcher at the University Center Nour El-Bachir of Elbayadh. He also worked as a Postdoctoral Research Fellow at Uninettuno University, Rome Italy, from September 2014 till July 2015. His research interests include Wireless and Optical networks, Routing Protocols and Security in Wireless Sensor Networks.

Mohammed Belkheir received his Telecom Engineer degree (1998) from National Telecoms Institute (INTTIC), Algeria, Msc Degree (2002) from France Telecom Formation (France), Magister (2007) and PhD (2015) Degrees from Djilali Liabes University of Sidi Bel Abbes (UDL) Algeria in mobile ad hoc and wireless sensor networks field. Currently he is an associate professor at University center of Nour Bachir El Bayadh as lecturer researcher in Telecoms and networking. His research interests are wired and wireless networks, Network Virtualization, QoS and security concerns for IoTs.

Hicheme Sid Ahmed Belkhira is Associate Professor in the Technology department Nour Bachir university center of Elbayadh, Algeria He received an engineering degree from Tahar Moulay University of Saida, Algeria in 2006, the MS degree from Sciences and Technology University of Oran,Algeria in 2010, the PhD degree from Djillali Liabes University (UDL) of Sidi Bel Abbes, Algeria and University of Haute alsace UHA France in 2020. His research interests are in networking, including wireless ad-hoc, sensor networks, vehicular networks, IoT, 5G, network security and QoS.