A new lightweight approach to mitigate the version number attack in RPL-based IoT networks

Mohammed BELKHEIR
LTTNS Laboratory, University Djilali Liabes, Sidi Bel Abbes, Algeria
LIMA Laboratory, Nour Bachir University Center. El-Bayadh, Algeria
belkheir_m@yahoo.fr

Mehdi ROUISSAT
Department of Electronics
Nour Bachir University Center. El-Bayadh,
STIC Laboratory, University Aboubek Belkaid, Tlemcen, Algeria.
Mehdi.m.rouissat@gmail.com

Hicham Sid Ahmed Belkhira
Department of Electronics
Nour Bachir University Center. El-Bayadh, Algeria
belkhira.hichem@gmail.com

Mohamed Achraf Boukhobza
Department of Electronics
Nour Bachir University Center. El-Bayadh, Algeria
bzaachraf@gmail.com

Mohamed Salim Abdoun
Department of Electronics
Nour Bachir University Center. El-Bayadh, Algeria
abdouns33@gmail.com

***Abstract*— In the recent few years, the cybersecurity challenge stood out to the research community with the drastic evolution of IoT networks. The present article deals with the version number attack, a major security concern for IoT networks using RPL as routing protocol. Although a number of techniques have been proposed to mitigate this kind of attack, the main challenge of the proposed technique remains to contemplate the limited capabilities of the embedded devices composing the network in terms of computing, storage and energy. This makes decentralized heavy algorithms and cryptographic solutions not suitable to cope with the inherited resources of nodes and the network core routing operations.**

The main purpose of our present work is to implement a new lightweight approach aiming to reduce the destructive effect of the version number attack, whereby the node takes into consideration only the version number received from its preferred parent and ignores other versions of its neighborhood. This leads to propagate the same version number value of the root node over the entire network and ensures the network integrity, by reducing the repeated global repair process caused by falsified values of the version number.

The proposed method is implemented under Contiki 3.0 operating system and simulations are performed using COOJA for different scenarios while changing the intruder node position in the network. The obtained results obviously depict that our proposed method yields better aftermath, where the energy saving is between 10% and 50%, depending on the intruder position, while the control overhead is between 2% and 50%.

Keywords*— *IoT, RPL, DODAG, Version number, DIO, COOJA

I. INTRODUCTION

Nowadays, the networking domain undergoes a massive evolution by implementing new trend of wireless networks able to receive, process and transmit data. Such a well-known network is The Internet of Things (IoT) [1], where the main purpose is to gather data from different environments and relay it to the internet for further computing and decisions. The market witnesses the wide spread of this technology in various domains of our lives like healthcare, energy industries, wearable technologies, automotive, military…etc [2]. An IoT network is essentially composed by small smart devices called sensors that are expected to collect data from a supervised environment. These sensors are connected to a sink playing the gateway role to the internet for a real-time supervision and process [3]. The network connectivity of an IoT in the lower layers is assured by implementing well-known technologies such as IPV6, 6LoWPAN, IEEE802.15.4 [4]. Despite the advantages brought by this trend of networks, many concerns remain in challenge such as the inter-connectivity, the topology changing and the security issues [5].

In this paper we are focusing on the security concern, whereas IoT networks witness a drastic scalability evolution from a closed network, to a decentralized wide area network composed by a millions of devices. Statistics held by [6] reveal that the number of the connected IoT devices is expected to outstrip the 50 billion in 2030. Likewise, the inter-connectivity poses an enormous security challenge against external threats, especially when the IoT is integrated to the new internet platforms as cloud computing or Big Data systems. However, the proposed secure mechanisms should take into consideration the network constraints, especially the limited computing, storage and energy characteristics of the IoT-enabled devices, even though they are based on Artificial Intelligence algorithms or cryptographic solutions [7].

The main purpose of our present work is to propose a lightweight technique for reducing the tampering effect of the version number attack in RPL-based IoT networks. Therefore, the rest of the paper is organized as follows: At first, in section (II) we give a brief description of the RPL routing protocol and its global repair mechanism based on the version number. After that, in section (III) we describe how an intruder node can perform a version number attack and its malicious effect by exhausting the network resources. In the section IV, we explain our proposed method and its outperformed action to lessen the version number attack affect, while simulating different scenarios of an IoT network. Finally, the conclusion resumes our work and highlights the efficiency of our proposed approach.

II. RPL AND THE DODAG VERSION NUMBER

A. RPL protocol description

RPL (Routing Protocol for Low Power and Lossy Networks) has been designed by IETF as routing protocol for constrained networks, named LLNs (Low power and Lossy Networks). RPL specification is defined by IETF in the RFC6550 [8]. The RPL is based on the IPV6 protocol, where the topology adopted is mesh/tree, forming thereby an acyclic graph named DODAG (Destination Oriented Directed Acyclic Graphs) built from the root which represents the data sink of the graph and plays the gateway role of the network. For the core routing operation and managing the DODAG, RPL defines four control messages: DIS, DIO, DAO and DAO-ACK messages [9], as depicted by figure 1.

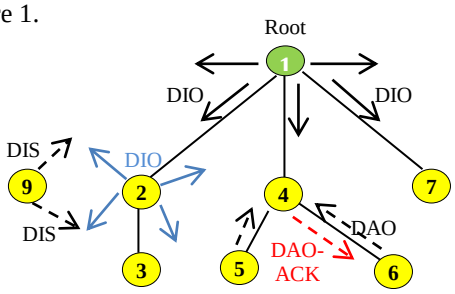

Fig.1. RPL Topology and control messages

The DIO message (DODAG Information Object) represents a reply to a DIS message (DODAG Information Solicitation) which is sent by a node to request DODAG information in order to join the network. The DIO message vehicles DODAG parameters as: DODAG ID, Version Number, Rank value and MOP (Mode of Operation) that allow each node to joining the DODAG. The latters are updated where a topology changing ensues, by using a trickle timer algorithm. The aim of the trickle timer is to ensure a fewer transmission of DIO messages to save nodes resources and reduce the control overhead. In addition, RPL implements an Objective Function (OF) that allow nodes performing upwards paths to the root and selecting their parents according to specific parameters [10].

The DODAG version number is an important parameter to establish a DODAG topology and maintain its integrity. This 8-bit integer value is initiated by the root node and broadcast via DIO messages (by default the version number value is 240) [11]. Once a topology change occurs, the root triggers a RPL mechanism called "global repair" by incrementing the current version number. Thus, receiving nodes compare the existing version number with their own, if the new value is higher; nodes should reset their trickle timer in order to recalculate their new willingness parameters (Rank) to rebuild the DODAG.

B. Version number attack

Since RPL has no mechanism to keep the root version number unchanged through the entire network [12], a malicious node can easily counterfeit the value and propagate it via its DIO messages to its neighborhood. Consequently, a receiving node initiates a *"local repair"* process by leaving its preferred parent and selecting a new parent among its parent list, based on the rank value. Then, the receiving node resets its trickle timer and broadcast this new version to their neighbors as well. Therefore, every node in the network should perform the same process aforesaid. Upon this new version received by the root, the latter triggers a "global repair" process, by incrementing the current version and broadcast it again via DIO messages to its neighborhood. This continually upsetting behavior is well-known as a "version number attack", and it causes a harmful volatility problem in the network. This unpredictable state affects the routing process by creating loops, making the established routes unavailable and forcing the unnecessary rebuilt of the DODAG. Thus, nodes should consume more their processing resources and generate an extra control traffic to rebuild their repositories which leads to exhausting their intrinsic resources.

C. The proposed DIO message process

At first, we should notice that RPL allows nodes to trust any version number received by DIO messages from their neighbors, even from their children. Therefore, our present work is focusing on guaranteeing that the same version number is disseminated downward to the entire network, in order to avoid loops and a subsequent global repair triggering.

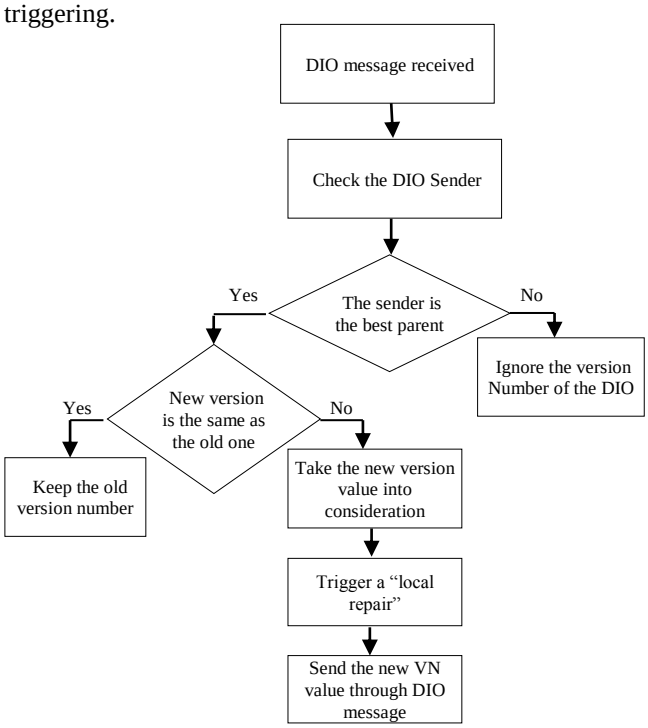

Fig.2. The proposed process of DIO message

Therefore, the node must take into consideration only the version number received from its best parent, and ignores any version number from its children set. The proposed DIO process is depicted by the figure 2, in which

we ensure the avoidance of the coexisting different version number values.

In the following section we illustrate how this process is implemented, and we analyze the positive effect of our approach for different scenarios of the version number attack.

III. SIMULATIONS AND ANALYSIS

Simulations are performed using COOJA under Contiki 3.0, operating system, a widely known OS for IoTs [13], whereas "Wireshark" tool is used to analyze the traffic over the network.

In all our performed simulations, we used Z1 motes[14], where characteristics are summarized in Table 1.

TABLE I. SIMULATION PARAMETERS

Parameter		Value
OS		Contiki 3.0
Traffic Analysis		Wireshark (1.7.2)
Power analysis		Powertrace
Z1 mote	µController	MSP430F2617 – 16 bits RISC
	CPU Frequency	16 MHz
	Transceiver	CC2420 (IEEE.802.15.4), 2.4 GHz
	RAM	8KB with 92KB (flash memory)
	Voltage	3V
	TX Current	17.4 mA
	RX Current	18.8 mA
	LPM Current	0.426 mA
RTimer		32768 ticks per second
Measurement interval		10 seconds
Simulation Time		10 Mn

A. Reference Network

At first, we build a reference network composed by 7 nodes and 1 sink as root.

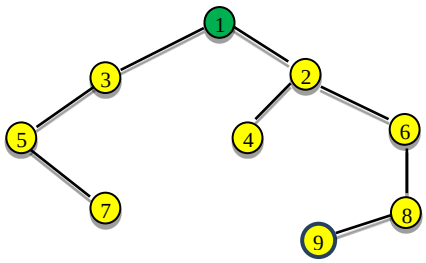

Fig.3 The reference network topology

After three minutes of simulation, the network has converged as illustrated by figure 3, where nodes 2 and 3 are directly connected to the sink (1). Node 2 connects node 4 and 6, and node 3 connects only node 5. Node 6 connects node 8 and the latter is a parent for the node 9. Nodes 4,7,9 are considered as leaves nodes. In this case, all the nodes are considered as legitimate and there is no harmful behavior.

by the following, simulations results of the reference network will be used as a benchmark to highlight the effect of the version number attack, and then the aftermath of our proposed approach.

B. Version number attack effect

The version number attack is implemented by adding a malicious behavior on node (Id 9), as illustrated by figure 4.

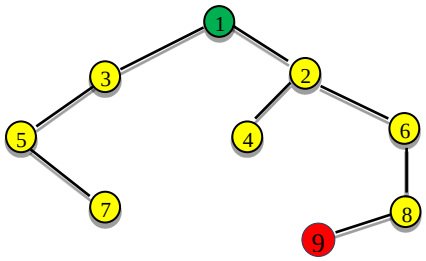

Fig.4 Version number attack with node 9 acting as malicious

To implement the attack in COOJA, the file containing the DIO processing *"Contiki 3.0_core_net_rpl_rpl-icmp.c"*, is modified as follows:

```
/* DAG Information Object */
pos = 0;
buffer = UIP_ICMP_PAYLOAD;
buffer[pos++] = instance->instance_id;
buffer[pos++] = dag->version++;
```

The last instruction shows that the malicious node continually increments its current version number and inserts it within its outgoing DIO message. Upon receiving this counterfeit version by the neighboring nodes, they reset their trickle timer and broadcast the new value as well. When the root detects this unauthorized increment, it directly triggers the global repair mechanism which produces an instable state of the network. In this upsetting case, simulations reveal that the network after 30 mn the convergence didn't converge compared to the normal case where the DODAG is built just after 3 mn of simulation. This explains the hurtful effect of the malicious node leading the network to an instable state.

C. Network lifetime analyzis

In order to evaluate the network lifetime, we calculate the consumed energy by means of a tool named "Powertrace" under COOJA. Powertrace collects data power based on number of ticks, and the formula (1) allows calculating the residual energy by node [15]:

$$Energy\ (mj) = \frac{Energest_{Value} \cdot Current \cdot Voltage}{RTIMER_{Second}} \quad (1)$$

Where:

- $Energest_{Value}$: Difference between the number of ticks in two time intervals
- $Current$: Consumed current in each operating state of the node (in mA)
- $Voltage$: Consumed voltage by the node (in Volts)

- $RTIMER_{Second}$: Number of ticks per seconds (see Table 1)

Figure 5 illustrates the obtained results of the consumed energy by each node in the reference network after ten minutes of simulation

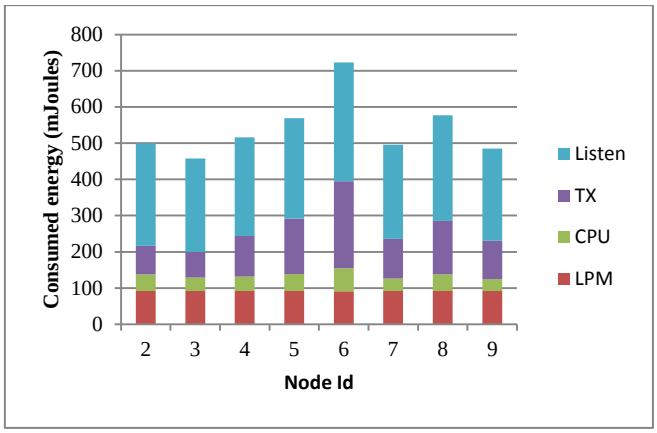

Fig.5 Energy consumed during 10 Mn of simulation in case of the reference network

As shown by figure 5, the residual energy depends on how long each mode the mode is operating [16]. The four usual modes are:

- LPM: low power mode or sleep mode
- CPU: processing Mode
- Radio Listen: includes the listening, the receiving, and the idle states
- Radio Transmit: refers to the TX mode

Figure 6 depicts a comparison between the total energy consumed by each node in the case of the attack with respect to the reference network.

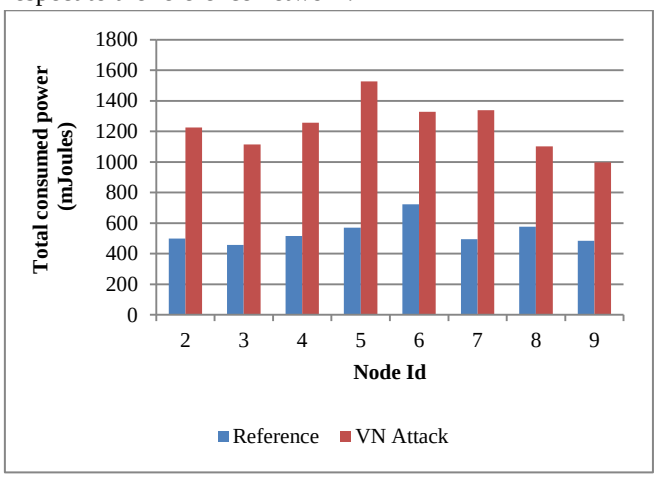

Fig.6 Energy consumption comparison (Reference network Vs Attack case)

It can be noticed that the energy consumed in the reference network case is between [485 mJ, 722mJ], while in the case of attack the interval is [998 mJ, 1526mJ]. Thus, the surge of the energy consumption between the two cases is about 227%, which is very important. This is mainly caused by the nodes activity which increased in the case of the version number attack, where the behavior of the intruder node forces nodes to send more control packets in order to accomplish the network convergence and to maintain the topology consistency. Nodes exchange unnecessary traffic to perform their repositories among their neighborhood which increases the sending, receiving and forwarding tasks of a node.

In addition, the energy gap between nodes in the case of the attack is about 528 mJ with respect to the reference network where the gap is about 237 mJ. This important gap may induce the extinction of some active nodes so earlier than others and leads to create holes in the network, even if the dead node is gaining a good position in the DODAG. Thus, the version number attack has an intrinsic effect by exhausting nodes resources, and an extrinsic effect related to the core networking activity by disrupting established routes and creating holes in the network.

D. Control overhead analyzis

The control overhead is an essential parameter to analyze the effect of the version number attack. It refers to the number of control packets broadcast over the entire network to establish and maintain the topology. The higher the control overhead is for a given network, the lower is its efficiency leading to exhausting the nodes resources. The control overhead is calculated using the following formula:

$$Control\ Overhead\ = \sum_1^n DIS + \sum_1^n DIO + \sum_1^n DAO \qquad (2)$$

Where n is the number of nodes in the network.

After ten minutes of simulations, the control overhead results are shown by the figure 5.

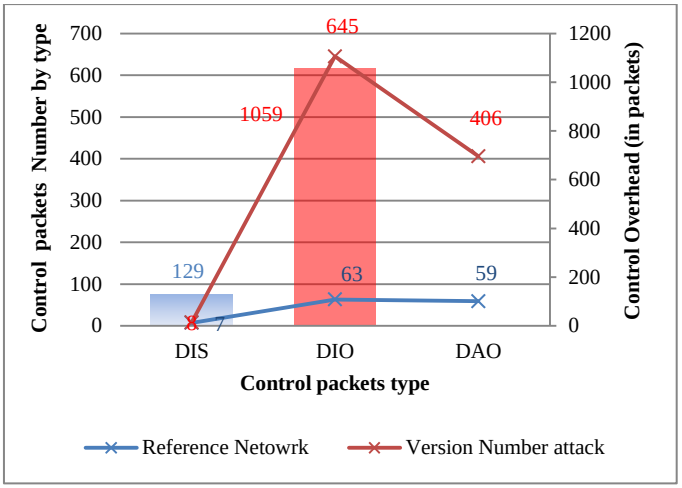

Fig.7 Control overhead comparison between the reference network and the network under attack

We notice that the control overhead in the case of the attack is 1059 packets, with the respect of the reference network where it is 129 packets; therefore it has increased 8 times more. Moreover, there are more DIO messages exchanged than other control messages, which explains the harmful behavior of the malicious node (Id 9), where it continually increments the version number and broadcast it to its neighborhood. Thus, every node receiving the new

value resets its trickle timer and broadcast the new version via DIO messages, which explain the increase of the control overhead.

E. Implementation of the proposed approach

In order to implement our new process illustrated by figure 2, the file *"Contiki 3.0_core_net_rpl_rpl-icmp.c"* is updated by adding the following code:

```
possible_version = buffer[i++];  // A temporary variable
....
instance = rpl_get_instance(dio.instance_id);
dag = instance->current_dag;
if(uip_ipaddr_cmp(&from , rpl_get_parent_ipaddr
(instance->current_dag->preferred_parent)) )
// if the DIO is received from the parent so do as follows
  {
  dio.version = possible_version;          // take the VN
  }
  else  {
  dio.version = dag->version ; // Don't take the VN
  }
...
```

Consequently, that the node performing the DIO process is taking into consideration the version number received only from its parent and ignoring other coming version number from its neighborhood.

To demonstrate the efficiency of the new proposed DIO processing, we consider 3 scenarios (figure 6), while changing the malicious node ranking in the DODAG. In the first scenario the intruder acts as a leaf node and it is connected to node 8. However, in the 2nd scenario the intruder has a good position where it is a parent of 1 node (node 8), then it is placed to be a neighbor of the root in the 3rd scenario.

In what follows we analyze the energy consumed and the control overhead by keeping the same simulation parameters, in order to better highlight the aftermath of our proposed approach.

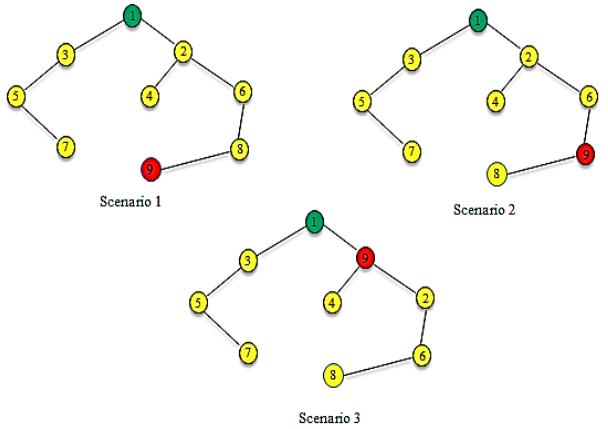

Fig.6 The three analyzed scenarios for our proposed approach

The energy computation result is shown by figure 7 for the three simulated scenarios.

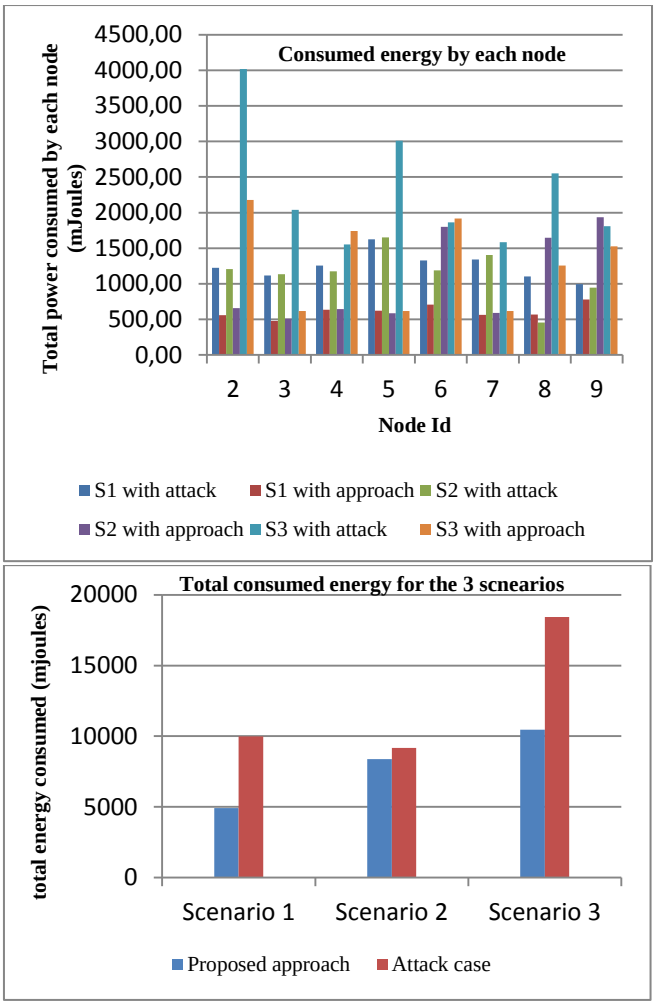

Fig.7 Energy consumption analysis for the three scenarios

It's obviously illustrated by the two plots of the figure 7 that the total energy consumed is much higher in case of the attack, and it decreases where we implement our method. This saving is about 50% for scenarios 1&3, and 10% for scenario 2. In addition, in the case of our approach, the increase in the total energy is regular, which confirms that the new process performs a deterministic behavior in the network, compared to the version number attack case. This deterministic behavior leads to avoid holes and preserve the network lifetime. Therefore, our proposed method acts better for each simulated scenario and is more efficient to mitigate the damage caused by the attacker node.

Let's see how our proposed approach is carrying out in terms of the control overhead for the three simulated scenarios. The figure 8 shows the control overhead evolution in the case of the attack (red color) and after implementing our method (blue color). We can observe that the control overhead increases in the two cases, starting from scenario 1 to scenario 3.

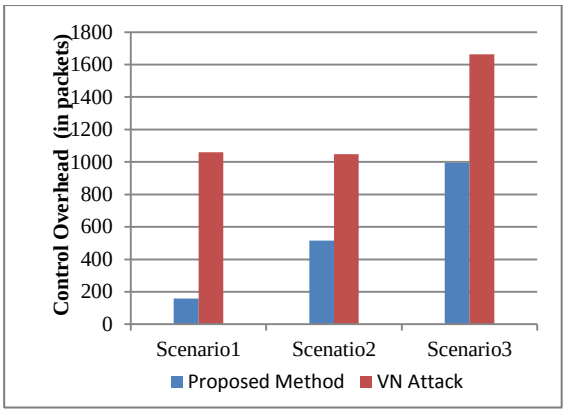

Fig.8 The Control overhead analysis for the three simulated scenarios

This depends on the intruder rank in the DODAG, where it was a simple leaf node in scenario 1 to having a good position being the root's neighbor. In the scenario 1, the harmful action of the attacker (Node 9) is directly eliminated by its parent (Node 8), according to our proposed method. Thus, it ignores the falsified version number being carried out by the intruder DIO messages. However, for scenarios 2 and 3, where the intruder is placed in a good position as a parent for other nodes (node 8 for scenario 2 and nodes 2,4,6,8 for scenario 3), it disseminates the falsified version number to its children and forces them to continually resetting their trickle timer to broadcast the falsified version number. Thus, the control overhead increases as long as the malicious node climbs the DODAG.

Simulation results depicts that the average value of the decrease is about 700 messages, which explains the important efficiency brought by our method. Based on the percentage difference analysis, the efficiency of our approach is about 148% for the scenario 1, 68% for scenario 2 and 50% for scenario 3. So, our approach acts better especially for the scenario 1 where the intruder is acting as a simple leaf node. The difference of the control packets for this scenario is about 900 messages. This is mainly due to our approach, where the parent ignores the falsified version number coming from the intruder child.

IV. Conclusion and Future Work

In this paper, by modifying the DIO processing of a node, we proposed and presented a new lightweight approach for mitigating the version number attack in an IoT network. The aim of our method is to allow the version number propagating downward the DODAG, where the node takes into consideration only the version number received from its preferred parent and ignores any other version from its neighborhood.

Simulations have been conducted for different scenarios, while changing the intruder node position in the network, where the obtained results obviously depict the aftermath of our proposed method in terms of the network lifetime and the control overhead. The energy saving is between 10% and 50%, depending on the malicious node position, while the control overhead is between 2% and 50%. These results demonstrate that our proposed method acts better to mitigate the harmful effect of the version number attack in a RPL-based IoT network.

As a future work, we expect to enhance the accurate action of our proposed process by delaying the reset of the trickle timer to allowing each node to gather a set of version number values in order to select the trusted one from its neighborhood.

References

[1] S. Rachit, Bhatt and P.R Ragiri, “Security trends in Internet of Things: a survey”. SN Appl. Sci, Vol3, 121, Jan. 2021. https://doi.org/10.1007/s42452-021-04156-9

[2] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal and B. Sikdar, "A survey on IoT security: Application areas, security threats, and solution architectures," in IEEE Access, vol. 7, pp. 82721-82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[3] F. Al-Turjman, “Artificial Intelligence in IoT”. Transactions on computer intelligence and computational Intelligence. 2019. Springer.

[4] M. A. Boudouaia, A. Ali-Pacha, A. Abouaissa and P. Lorenz, "Security Against Rank Attack in RPL Protocol," in IEEE Network, vol. 34, no. 4, pp. 133-139, August 2020.
doi: 10.1109/MNET.011.1900651.

[5] H. Touqeer, S. Zaman, R. Amin, H. Mudassar, F. Al-Turjman and M. Bilal, “Smart home security: challenges, issues and solutions at different IoT layers”. J Supercomput. May 2021. https://doi.org/10.1007/s11227-021-03825-1

[6] https ://www.statista.com/statistics /47126 4/iot-number-ofconnected-devices-world wide/. (Accessed July 2021)

[7] M. Kuzlu, C. Fair and O.Guler, “Role of Artificial Intelligence in the Internet of Things (IoT) cybersecurity. Discover Internet of Things, 1, 7. Februay 2021. https://doi.org/10.1007/s43926-020-00001-4

[8] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.P. Vasseur, R. Alexander, “RFC 6550: RPL IPv6 routing protocol for low-power and lossy networks”. https://tools.ietf.org/html/rfc6550, 2012. Accessed 2 April 2021.

[9] S.W. Min, SH. Chung, H.J. Lee and Y.V. Ha, “Downward traffic retransmission mechanism for improving reliability in RPL environment supporting mobility”. International Journal of Distributed Sensor Networks. vol 16, issue 1, Februay 2020. https://doi.org/10.1177/1550147720903605

[10] A. Lamaazi and N. Benamar, “A comprehensive survey on enhancements and limitations of the RPL protocol: A focus on the objective function”. Ad Hoc Networks, Vol 96, January 2020. https://doi.org/10.1016/j.adhoc.2019.102001

[11] Z. Almusaylim, N. Jhanjhi, and A. Alhumam, “Detection and mitigation of RPL rank and version number attacks in the Internet of Things: SRPL-RP”. Sensors. Vol 21: 5997, October 2020. https://doi.org/10.3390/s20215997

[12] S, Bhatt and PR Ragiri, “Security trends in Internet of Things: a survey”. SN Applied Sciences, vol 3 (121). Janaury 2021. https://doi.org/10.1007/s42452-021-04156-9

[13] S. Mittal, SK. Nagdeo and J. Mahapatro, "Performance Study of IoT enabled Wireless Body Area Network in Contiki Platform”. IEEE Global Conference for Advancement in Technology (GCAT), pp. 1-5, October 2019. doi: 10.1109/GCAT47503.2019.8978467

[14] Z1 datasheet. www.zolertia.com. (Accessed April 2021)

[15] S. Cakir, S. Toklu and N. Yalcin, "RPL Attack Detection and Prevention in the Internet of Things Networks Using a GRU Based Deep Learning". IEEE Access, vol. 8, pp. 183678-183689, 2020, doi: 10.1109/ACCESS.2020.3029191

[16] N.N. Hadaya and S.A. Alabady, “ Improved RPL protocol for Low-Power and Lossy Network for IoT environment”. SN Computer Science., vol 2 (341), June 2021. https://doi.org/10.1007/s42979-021-00742-1