Topology Authentication in RPL

Martin Landsmann
Heiner Perrey, Osman Ugus
HAW Hamburg
firstname.lastname@haw-hamburg.de

Matthias Wählisch
Freie Universität Berlin
m.waehlisch@fu-berlin.de

Thomas C. Schmidt
HAW Hamburg
t.schmidt@ieee.org

***Abstract*—The Routing Protocol for Low-Power and Lossy Networks (RPL) is a proposed standard by the *Internet Engineering Task Force* (IETF). Although RPL defines basic security modes, it is still subject to topology attacks. VeRA is an authentication scheme which protects against attacks, based on the version number and rank. This work presents two rank attacks which are not mitigated by VeRA. In the first attack, the adversary can decrease its rank arbitrarily. Hence, it can impersonate even the root node. In the second attack, the adversary can decrease its rank to that of any node within its access range. We present an enhancement of VeRA to mitigate the first attack. Additionally, a basic approach for mitigating the second attack is introduced.**

I. Introduction

RPL [1] is a routing protocol for low-power and lossy networks (LLN). It is organized as a Destination Oriented Directed Acyclic Graph (DODAG) directed towards a special node, called root node. Routes to the root are established via parent nodes. That is, each node sends its data to a parent, which is located closer to the root in the topology. In turn, each parent node sends the received data until the root node is reached.

The *rank* of a node represents its distance to the root and defines the relationship of a parent and its children. The rank increases monotonically with distance to the root node. Hence, a node with a smaller rank forwards more traffic than a node with a larger rank. Accordingly, to get a greater impact on the network, an attacker tries to obtain the lowest rank possible.

RPL uses a unique and increasing version number for each DODAG. Occasionally, the DODAG needs to be updated, e.g., to repair routing inconsistencies. During a version update, the nodes (re-)select their parents according to the rank values disseminated by their neighbors. Hence, an adversary can simply announce a small rank value to become a parent of a large number of sensor nodes. To address this problem, VeRA [2] proposes an authentication scheme, in which every node can check, if the version number is updated by the root, and if the rank of the parents is monotonically increasing[1]. However, VeRA still allows two different rank attacks. While the first one allows the attacker to successfully get any rank of its choice, the second still enables the adversary to replay its parents rank. In this work we summarize the VeRA protocol and show two attacks on this approach (II). We propose a solution for the first attack and a partial solution for the second attack (III).

[1]If a node is running DODAG version $i = 0$, the next version must be $i+1$. Similarly, if a node has the rank j, its parent must have the rank $j-\Delta$, where Δ depends on the use-case. For simplicity, we assume $\Delta = 1$ in this work.

II. VeRA - Version number and Rank Authentication

A. The Protocol

VeRA [2] uses hash chains to prevent the adversary from increasing the version number of the DODAG as well as decreasing its rank. A version number increase causes a global repair in the DODAG and provides a chance for an adversary to decrease its own rank and to move closer to the root node.

In VeRA, each version V_i is represented by an element of a hash chain $V_n, \dots, V_0$, where $V_i = h^{n+1-i}(r)$, r is a random seed, h is a one-way hash function, and n is a sufficiently large parameter denoting the largest version number possible. Moreover, the ranks in a version V_i are represented by a rank chain $R_{i,0}, \dots, R_{i,l}$, where $R_{i,l} = h^{l+1}(x_i)$, x_i is a random number, $R_{i,j}$ is the element representing the rank j, and l is the largest rank value possible for a version V_i. In the initialization phase, the root node disseminates the packet $\langle V_0, Init_{VN}, R_{1,l}, \{V_0, MAC_{V_1}(R_{1,l})\}_{sign}\rangle$, where $Init_{VN}$ is the actual version number. The receiving nodes verify the signature and accept the parameters V_0 and $R_{1,l}$, if the signature is valid. In subsequent version updates, say V_i, the root node disseminates the packet $\langle V_i, Init_{VN} + i, MAC_{V_{i+1}}(R_{i+1,l})\rangle$. Each node then verifies the version number by checking that $h(V_i) == V_{i-1}$ holds. Each node verifies the rank, j, announced by its parent in a version V_i by checking if $MAC_{V_i}(R_{i,l}) == MAC_{V_i}(h^{l-j}(R_{i,j}))$.

B. Vulnerabilities of VeRA

The VeRA approach is still subject to two rank attacks: (1) The first attack requires the adversary to prevent the sensor nodes from receiving two subsequent version updates initiated by the root. For instance, if the attacker withholds the updates V_1 and V_2, he has the ability to create any MAC, $MAC_{V_2}(R'_{2,l})$, by using the key V_2 he received in the second update. Once he releases version V_2, the adversary can claim any rank j by sending the forged hash chain $R'_{2,j}$ to its children. (2) An adversary may replay the rank hash of its parent in its access range to decrease its own rank by one in the topology.

III. Enhancing VeRA

Mitigating Attack – 1: The vulnerability exploited by this attack originates from the missing correlation between the rank hash chain and the version number hash chain in VeRA. We propose a nested encryption to entangle both hash chains. Prior to the initialization phase, the root node creates a version hash chain as in the original version of VeRA. The difference in our method is that the root also creates the rank hash chains for all versions in advance. Additionally, the last element of the rank hash chains are nested in each other as follows: for $i = (n-1), \dots, 1$, each element i is encrypted using the encryption of the $(i+1)$th element as the key k_i. That is, $c_i = enc_{k_i}(R_{i,l})$, where $k_i = c_{i+1}$ and $c_n = R_{n,l}$ (see Table I). Then, each cipher c_{i+1} is distributed in a version update V_i following the original VeRA as depicted in Figure 1.

The receiving nodes verify the version number like in VeRA. In contrast to VeRA, the rank hash chain in a version V_i is verified differently. First, the nodes decrypt the cipher c_i received in V_{i-1} using the cipher c_{i+1} as the key to obtain $R_{i,l} = dec_{c_{i+1}}(c_i)$. Then, $R_{i,l}$ is verified using the MAC as described in VeRA. Finally, the nodes verify the rank of their parents, j, by checking if $MAC_{V_i}(R_{i,l}) == MAC_{V_i}(h^{l-j}(R_{i,j}))$. Modifying a cipher c_i breaks the decryption of the cipher c_{i-1} leading to an invalid $R'_{i-1,l}$ which cannot be verified by the MAC.

Mitigating Attack – 2: To mitigate the rank-replay attack, we propose to use a challenge response procedure. The main idea behind our approach is that two honest neighboring nodes with the same rank r have a parent with the rank $r-1$. Considering this observation, our protocol is based on a challenge that can only be solved, if the node has indeed a parent with rank $r-1$ in its transmission range.

Let us assume that an honest node H with rank j and a malicious node M within the range of H. Choosing H as a parent, M calculates the rank $j+1$. In a replay attack the node M claims to have rank j. This is possible, since M learns the $R_{i,j}$ disseminated from a parent node in its transmission range.

To mitigate such an attack, the honest node H performs a challenge response protocol: H chooses a random number α and encrypts it with $R_{i,j-1}$ and sends the ciphertext to its parent P. P decrypts the ciphertext and encrypts α with its parent's hash $R_{i,j-2}$ and sends the challenge $\beta = enc_{R_{i,j-2}}(\alpha)$ back to H. Finally, H challenges the malicious node M with β. Node M can provide a correct response only if it has a parent with rank $j-1$.

A remaining issue is how to detect the attack. We propose an inquiry in which a node is requested to multicast its rank to all neighbors using a RPL control message with minor adjustments. Since both parents and children receive this message, an inconsistency (or attack) can be detected. A malicious node either uses the *true* or *false* rank. The false rank is detected by the parents. Using the true rank renders the attack harmless.

$R_{i,l}$	use key k_i	cipher $c_i = enc_{k_i}()$
$R_{n,l}$	–	$c_n = R_{n,l}$
$R_{n-1,l}$	$k_{n-1} = c_n$	$c_{n-1} = enc_{k_{n-1}}(R_{n-1,l})$
…	…	…
$R_{2,l}$	$k_2 = c_3$	$c_2 = enc_{k_2}(R_{2,l})$
$R_{1,l}$	$k_1 = c_2$	$c_1 = enc_{k_1}(R_{1,l})$

TABLE I
CREATION OF THE RANK ENCRYPTION CHAIN

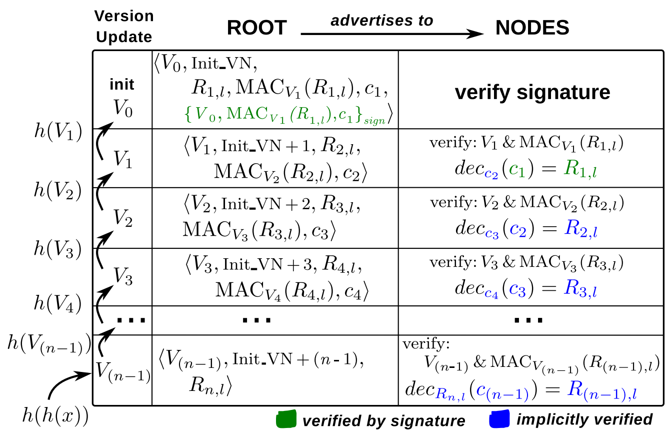
Version Update
ROOT
advertises to
NODES
init
V_0
$\langle V_0, \text{Init\_VN}, R_{1,l}, \text{MAC}_{V_1}(R_{1,l}), c_1, \{V_0, \text{MAC}_{V_1}(R_{1,l}), c_1\}_{sign}\rangle$
verify signature
$h(V_1)$
V_1
$\langle V_1, \text{Init\_VN}+1, R_{2,l}, \text{MAC}_{V_2}(R_{2,l}), c_2\rangle$
verify: V_1 & $\text{MAC}_{V_1}(R_{1,l})$
$dec_{c_2}(c_1) = R_{1,l}$
$h(V_2)$
V_2
$\langle V_2, \text{Init\_VN}+2, R_{3,l}, \text{MAC}_{V_3}(R_{3,l}), c_3\rangle$
verify: V_2 & $\text{MAC}_{V_2}(R_{2,l})$
$dec_{c_3}(c_2) = R_{2,l}$
$h(V_3)$
V_3
$\langle V_3, \text{Init\_VN}+3, R_{4,l}, \text{MAC}_{V_4}(R_{4,l}), c_4\rangle$
verify: V_3 & $\text{MAC}_{V_3}(R_{3,l})$
$dec_{c_4}(c_3) = R_{3,l}$
$h(V_4)$
...
$h(V_{(n-1)})$
$V_{(n-1)}$
$\langle V_{(n-1)}, \text{Init\_VN}+(n-1), R_{n,l}\rangle$
verify: $V_{(n-1)}$ & $\text{MAC}_{V_{(n-1)}}(R_{(n-1),l})$
$dec_{R_{n,l}}(c_{(n-1)}) = R_{(n-1),l}$
$h(h(x))$
🟩 verified by signature
🟦 implicitly verified

Fig. 1. AUTHENTICATION OF THE RANK ENCRYPTION CHAIN

IV. Conclusions & Outlook

RPL is vulnerable to different topology attacks. A malicious node can exploit the rank value or version number to illegitimately improve its position in the DODAG. VeRA proposes a version number and rank authentication method to mitigate such attacks. In this poster, we identified two new vulnerabilities of VeRA, rank forgery and replay.

We proposed an encryption chain linking the version number and rank hash chain to prevent an attacker from propagating forged hash chains. Since this solution cannot mitigate rank-replay attacks, the attacker can still decrease its rank by one step. To address this problem, we introduced a challenge response and rank inquiry procedure. The combination of both approaches for a security scheme to defend against the rank-replay attack is subject to a forthcoming publication.

Acknowledgment

This work is partly supported by the BMBF within the projects SAFEST and SKIMS.

References

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF, RFC 6550, March 2012.

[2] A. Dvir and T. Holczer and L. Buttyan, "VeRA - Version Number and Rank Authentication in RPL," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, oct. 2011, pp. 709 –714.