Stream:	Internet Engineering Task Force (IETF)	
RFC:	9467	
Updates:	8967	
Category:	Standards Track	
Published:	January 2024	
ISSN:	2070-1721	
Authors:	J. Chroboczek	T. Høiland-Jørgensen
	IRIF, University of Paris-Cité	*Red Hat*

RFC 9467
Relaxed Packet Counter Verification for Babel MAC Authentication

Abstract

This document relaxes the packet verification rules defined in "MAC Authentication for the Babel Routing Protocol" (RFC 8967) in order to make the protocol more robust in the presence of packet reordering. This document updates RFC 8967.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The design of the Babel MAC authentication mechanism [RFC8967] assumes that packet reordering is an exceptional occurrence, and the protocol drops any packets that arrive out-of-order. The assumption that packets are not routinely reordered is generally correct on wired links, but turns out to be incorrect on some kinds of wireless links.

In particular, IEEE 802.11 (Wi-Fi) [IEEE80211] defines a number of power-saving modes that allow stations (mobile nodes) to switch their radio off for extended periods of time, ranging in the hundreds of milliseconds. The access point (network switch) buffers all multicast packets and only sends them out after the power-saving interval ends. The result is that multicast packets are delayed by up to a few hundred milliseconds with respect to unicast packets, which, under some traffic patterns, causes the Packet Counter (PC) verification procedure in RFC 8967 to systematically fail for multicast packets.

This document defines two distinct ways to relax the PC validation:

- using two separate receiver-side states, one for unicast and one for multicast packets (Section 3.1), which allows arbitrary reordering between unicast and multicast packets, and

- using a window of previously received PC values (Section 3.2), which allows a bounded amount of reordering between arbitrary packets.

We assume that reordering between arbitrary packets only happens occasionally, and, since Babel is designed to gracefully deal with occasional packet loss, usage of the former mechanism is **RECOMMENDED**, while usage of the latter is **OPTIONAL**. The two mechanisms **MAY** be used simultaneously (Section 3.3).

This document updates RFC 8967 by relaxing the packet verification rules defined therein. It does not change the security properties of the protocol.

2. Specification of Requirements

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Relaxing PC Verification

The Babel MAC authentication mechanism prevents replay by decorating every sent packet with a strictly increasing value, the Packet Counter (PC). Notwithstanding the name, the PC does not actually count packets: a sender is permitted to increment the PC by more than one between two successively transmitted packets.

A receiver maintains the highest PC received from each neighbour. When a new packet is received, the receiver compares the PC contained in the packet with the highest received PC:

- if the new value is smaller or equal, then the packet is discarded;
- otherwise, the packet is accepted, and the highest PC value for that neighbour is updated.

Note that there does not exist a one-to-one correspondence between sender states and receiver states: multiple receiver states track a single sender state. The receiver states corresponding to a single sender state are not necessarily identical, since only a subset of receiver states are updated when a packet is sent to a unicast address or when a multicast packet is received by a subset of the receivers.

3.1. Multiple Highest PC Values

Instead of maintaining a single highest PC value for each neighbour, an implementation of the procedure described in this section uses two values: the highest multicast value PCm and the highest non-multicast (unicast) value PCu. More precisely, the (Index, PC) pair contained in the neighbour table (Section 3.2 of [RFC8967]) is replaced by a triple (Index, PCm, PCu), where:

- Index is an arbitrary string of 0 to 32 octets, and
- PCm and PCu are 32-bit (4-octet) integers.

When a Challenge Reply is successful, both highest PC values are updated to the value contained in the PC TLV from the packet containing the successful challenge. More precisely, the last sentence of the fourth bullet point of Section 4.3 of [RFC8967] is replaced as follows:

OLD:

> If the packet contains a successful Challenge Reply, then the PC and Index contained in the PC TLV **MUST** be stored in the neighbour table entry corresponding to the sender (which already exists in this case), and the packet is accepted.

NEW:

> If the packet contains a successful Challenge Reply, then the Index contained in the PC TLV **MUST** be stored in the Index field of the neighbour table entry corresponding to the sender (which already exists in this case), the PC contained in the TLV **MUST** be stored in both the PCm and PCu fields of the neighbour table entry, and the packet is accepted.

When a packet that does not contain a successful Challenge Reply is received, the PC value that it contains is compared to either the PCu or the PCm field of the corresponding neighbour entry, depending on whether or not the packet was sent to a multicast address. If the comparison is successful, then the same value (PCm or PCu) is updated. More precisely, the last bullet point of Section 4.3 of [RFC8967] is replaced as follows:

OLD:

> At this stage, the packet contains no successful Challenge Reply, and the Index contained in the PC TLV is equal to the Index in the neighbour table entry corresponding to the sender. The receiver compares the received PC with the PC contained in the neighbour table; if the received PC is smaller or equal than the PC contained in the neighbour table, the packet **MUST** be dropped and processing stops (no challenge is sent in this case, since the mismatch might be caused by harmless packet reordering on the link). Otherwise, the PC contained in the neighbour table entry is set to the received PC, and the packet is accepted.

NEW:

> At this stage, the packet contains no successful Challenge Reply and the Index contained in the PC TLV is equal to the Index in the neighbour table entry corresponding to the sender. The receiver compares the received PC with either the PCm field (if the packet was sent to a multicast IP address) or the PCu field (otherwise) in the neighbour table. If the received PC is smaller than or equal to the value contained in the neighbour table,

the packet **MUST** be dropped and processing stops. Note that no challenge is sent in this case, since the mismatch might be caused by harmless packet reordering on the link. Otherwise, the PCm (if the packet was sent to a multicast address) or the PCu (otherwise) field contained in the neighbour table entry is set to the received PC, and the packet is accepted.

3.1.1. Generalisations

Modern networking hardware tends to maintain more than just two queues, and it might be tempting to generalise the approach taken to more than just the two last PC values. For example, one might be tempted to use distinct last PC values for packets received with different values of the Type of Service (TOS) field, or with different IEEE 802.11 access categories. However, choosing a highest PC field by consulting a value that is not protected by the Message Authentication Code (MAC) (Section 4.1 of [RFC8967]) would no longer protect against replay. In effect, this means that only the destination address and port number as well as the data stored in the packet body may be used for choosing the highest PC value, since these are the only fields that are protected by the MAC (in addition to the source address and port number, which are already used when choosing the neighbour table entry and therefore provide no additional information). Since Babel implementations do not usually send packets with differing TOS values or IEEE 802.11 access categories, this is unlikely to be an issue in practice.

The following example shows why it would be unsafe to select the highest PC depending on the TOS field. Suppose that a node B were to maintain distinct highest PC values for different values T1 and T2 of the TOS field, and that, initially, all of the highest PC fields at B have value 42. Suppose now that a node A sends a packet P1 with TOS equal to T1 and PC equal to 43; when B receives the packet, it sets the highest PC value associated with TOS T1 to 43. If an attacker were now to send an exact copy of P1 but with TOS equal to T2, B would consult the highest PC value associated with T2, which is still equal to 42, and accept the replayed packet.

3.2. Window-Based Verification

Window-based verification is similar to what is described in Section 3.4.3 of [RFC4303]. When using window-based verification, in addition to retaining within its neighbour table the highest PC value PCh seen from every neighbour, an implementation maintains a fixed-size window of booleans corresponding to PC values directly below PCh. More precisely, the (Index, PC) pair contained in the neighbour table (Section 3.2 of [RFC8967]) is replaced by:

- a triple (Index, PCh, Window), where Index is an arbitrary string of 0 to 32 octets, PCh is a 32-bit (4-octet) integer, and Window is a vector of booleans of size S (the default value S=128 is **RECOMMENDED**).

The window is a vector of S boolean values numbered from 0 (the "left edge" of the window) up to S-1 (the "right edge"); the boolean associated with the index i indicates whether a packet with a PC value of (PCh - (S-1) + i) has been seen before. Shifting the window to the left by an integer amount k is defined as moving all values so that the value previously at index n is now at index (n - k); k values are discarded at the left edge, and k new unset values are inserted at the right edge.

Whenever a packet is received, the receiver computes its index i = (PC - PCh + S - 1). It then proceeds as follows:

1. If the index i is negative, the packet is considered too old, and it **MUST** be discarded.
2. If the index i is non-negative and strictly less than the window size S, the window value at the index is checked. If this value is already set, the received PC has been seen before and the packet **MUST** be discarded. Otherwise, the corresponding window value is marked as set, and the packet is accepted.
3. If the index i is larger or equal to the window size (i.e., PC is strictly larger than PCh), the window **MUST** be shifted to the left by (i - S + 1) values (or, equivalently, by the difference PC - PCh), and the highest PC value PCh **MUST** be set to the received PC. The value at the right of the window (the value with index S - 1) **MUST** be set, and the packet is accepted.

When receiving a successful Challenge Reply, the remembered highest PC value PCh **MUST** be set to the value received in the Challenge Reply, and all of the values in the window **MUST** be reset except the value at index S - 1, which **MUST** be set.

3.3. Combining the Two Techniques

The two techniques described above serve complementary purposes:

- splitting the state allows multicast packets to be reordered with respect to unicast ones by an arbitrary number of PC values, while
- the window-based technique allows arbitrary packets to be reordered but only by a bounded number of PC values.

Thus, they can profitably be combined.

An implementation that uses both techniques **MUST** maintain, for every entry of the neighbour table, two distinct windows, one for multicast and one for unicast packets. When a successful Challenge Reply is received, both windows **MUST** be reset. When a packet that does not contain a Challenge Reply is received, if the packet's destination address is a multicast address, the multicast window **MUST** be consulted and possibly updated, as described in Section 3.2. Otherwise, the unicast window **MUST** be consulted and possibly updated.

4. Security Considerations

The procedures described in this document do not change the security properties described in Section 1.2 of [RFC8967]. In particular, the choice between the multicast and the unicast packet counter is made by examining a packet's destination IP address, which is included in the pseudo-header and therefore participates in MAC computation. Hence, an attacker cannot change the destination address without invalidating the MAC, and therefore cannot replay a unicast packet as a multicast one or vice versa.

While these procedures do slightly increase the amount of per-neighbour state maintained by each node, this increase is marginal (between 4 and 36 octets per neighbour, depending on implementation choices), and should not significantly impact the ability of nodes to survive denial-of-service attacks.

5. IANA Considerations

This document has no IANA actions.

6. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8967] Dô, C., Kolodziejak, W., and J. Chroboczek, "MAC Authentication for the Babel Routing Protocol", RFC 8967, DOI 10.17487/RFC8967, January 2021, <https://www.rfc-editor.org/info/rfc8967>.

7. Informative References

[IEEE80211] IEEE, "IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", DOI 10.1109/IEEESTD.2021.9363693, IEEE Std 802.11-2020, February 2021, <https://ieeexplore.ieee.org/document/9363693>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

Acknowledgments

The authors are greatly indebted to Daniel Gröber, who first identified the problem that this document aims to solve and first suggested the solution described in Section 3.1.

Authors' Addresses

Juliusz Chroboczek
IRIF, University of Paris-Cité
Case 7014
75205 Paris CEDEX 13
France
Email: jch@irif.fr

Toke Høiland-Jørgensen
Red Hat
Email: toke@toke.dk