Secure BFD Sequence Numbers
draft-ietf-bfd-secure-sequence-numbers-04

Abstract

This document describes a security enhancements for the BFD packet’s sequence number.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 27, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

BFD [RFC5880] section 6.7 describes the use of monotonically incrementing 32-bit sequence numbers for use in authentication of BFD packets. While this method protects against simple replay attacks, the monotonically increasing sequence numbers are predictable and vulnerable to more complex attack vectors. This document proposes the use of non-monotonically-incrementing sequence numbers in BFD authentication TLVs to enhance the security of BFD sessions. Specifically, the document presents a method to generate pseudo-random sequence numbers on the frame by algorithmically hashing monotonically increasing sequence numbers. Further security may be introduced by resetting un-encrypted sequence to a random value when the 32-bit sequence number rolls-over.

2. Theory of operations

Instead of monotonically increasing the sequence number or even occasionally monotonically increasing the sequence number, the next sequence number is generated by computing a hash on what would have been the next sequence number using a shared key. That computed hash is then inserted into the sequence number field of the packet. In case of BFD Authentication [I-D.ietf-bfd-optimizing-authentication], the sequence number used in computing an authenticated packet would be this new computed hash. Even though the BFD Authentication
sequence number is independent of this enhancement, it would benefit by using the computed hash.

A normal BFD packet with authentication will undergo the following steps, where:

[O]: original RFC 5880 packet with monotonically increasing sequence number

[S]: psuedo random sequence number

[A]: Authentication

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>[O] [S] [A]</td>
<td>[A] [S] [O]</td>
</tr>
</tbody>
</table>

In order to encode a sequence number, the sender would identify a hash algorithm (symmetric) that would create a 32 bit hash. The hashing key is provisioned securely on the sender and receiver of the BFD session. The mechanism of provisioning such a key is outside the scope of this draft. Instead of using the sequence number, the sender encodes the sequence number with the hashing key to produce a hash.

Upon receiving the BFD Control packet, the receiver compares the received sequence number against the expected sequence number. The mechanism used for comparing is an implementation detail (implementations may pre-calculate the expected hashed sequence number, or decrypt the received sequence number before comparing against expected value). To tolerate dropped frames, the receiver MUST compare the received sequence number against the current expected sequence number (previous received sequence number + 1) and N subsequent expected sequence numbers (where N is greater than or equal to the detect multiplier). Note: The first sequence number can be obtained using the same logic as the My Discriminator value.

k: hashing key

s: sequence number

O: original RFC 5880 packet with monotonically increasing sequence number

R: remainder of packet

H1: hash of s
H2: hash of entire packet
A: H2 + insertion in packet

hash(s, k) = H1
hash((H1 + R), k) = H2
hash'(H1, k) == s ? Good sequence number : bad sequence number

Sender                Receiver
[O] [H1] [A] -------- [A] [H1] [O]

3. Impact of using a hash

Under this proposal, every packet’s sequence number is encoded within a hash. Therefore there is some impact on the system and its performance while encoding/decoding the hash. As security measures go, this enhancement greatly increases the security of the packet with or without authentication of the entire packet.

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

5. Security Considerations

While the proposed mechanism improves overall security of BFD mechanism, the security considerations are listed below:

Because of the fast rate of BFD sessions and it is difficult to change the keys (used for hashing the sequence number) during the operation of a BFD session without affecting the stability of the BFD session. It is, therefore, recommended to administratively disable the BFD session before changing the keys. If the keys are not changed, an attacker can use a replay attack.

Using this method allows the BFD end-points to detect a malicious packet (the decrypted sequence number will not be in sequence) the behavior of the session when such a packet is detected is based on the implementation. A flood of such malicious packets may cause a session to report BFD session to be operationally down.
The hashing algorithm and key size will determine the difficulty for an attacker to decipher the key from the transmitted BFD frames. Sequential nature of the payload (sequence numbers) simplifies the decoding of the key. It is, therefore, recommended to use longer keys or more secure hashing algorithms.

6. Acknowledgements

7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Mahesh Jethanandani
Cisco Systems, Inc
170 West Tasman Drive
San Jose, CA 95070
USA
Email: mjethanandani@gmail.com

Sonal Agarwal
Cisco Systems, Inc
170 W. Tasman Drive
San Jose, CA 95070
USA
Email: agarwaso@cisco.com
URI: www.cisco.com