Network Working Group
Internet-Draft
Updates: 5880 (if approved)
Intended status: Standards Track
Expires: February 6, 2021

M. Jethanandani
Kloud Services
S. Agarwal
Cisco Systems, Inc
A. Mishra
O3b Networks
A. Saxena
Ciena Corporation
A. Dekok
Network RADIUS SARL
August 5, 2020

Secure BFD Sequence Numbers
draft-ietf-bfd-secure-sequence-numbers-06

Abstract

This document describes a security enhancement for the sequence number used in BFD control packets. This document updates RFC 5880.

Status of This Memo

Copyright Notice

Table of Contents

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 incrementing sequence numbers are predictable and vulnerable to more complex attack vectors. This document proposes the use of non-monotonically-incrementing sequence numbers in the BFD authentication section 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. Since the monotonically increasing sequence number does not appear on the wire, it is difficult for a third party to launch a replay attack.

2. 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].

3. Theory of operation

Instead of inserting a monotonically, sometimes occasionally, increasing sequence number in BFD control packets, a hash is inserted. The hash is computed, using a shared key, on the sequence number. 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 [I-D.ietf-bfd-optimizing-authentication] sequence number is independent of this enhancement, it would benefit by using the computed hash.

As currently defined in BFD [RFC5880], a BFD packet with authentication will undergo the following steps, where:

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

[S]: pseudo random sequence number

[A]: Authentication

```
                Sender                    Receiver

                [O] [S] [A] ------------- [A] [S] [O]
```

This document proposes that for enhanced security in sequence number encoding, 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 document. 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 used in determining Local Discriminator value for the session or by using a random number.

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'((Packet - H2), k) == H2 ? Good packet : bad packet

hash'(H1, k) > previously received s ? Good sequence number : bad sequence number

```
                 Sender                 Receiver

                 [O] [H1] [A] -------- [A] [H1] [O]
```

The above diagram describes how the sender encodes and receiver decodes the sequence number. The sender starts by taking the monotonically increasing sequence number and hashing it. It replaces the sequence number with the hash. It then calculates the hash for the entire packet and appends the hash value to the end of the packet, before transmitting it.

The receiver hashes the entire packet without H2, and compares the hash value with the received hash (H2). If the hash values are equal, it is a good packet, else it is a bad packet. It then calculates the hash on the received sequence number to retreive s. If it is greater than the previously received monotically increasing sequence number, then the receiver knows it's a valid sequence number.

4. 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.

5. IANA Considerations

This document makes no request of IANA.

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

6. Security Considerations

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

Because of the fast rate of BFD sesions 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. The 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.

7. Acknowledgements

8. References

8.1. 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>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

8.2. Informative References

[I-D.ietf-bfd-optimizing-authentication]
Jethanandani, M., Mishra, A., Saxena, A., and M. Bhatia, "Optimizing BFD Authentication", draft-ietf-bfd-optimizing-authentication-11 (work in progress), July 2020.

Authors' Addresses

Mahesh Jethanandani
Kloud Services

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

Ashesh Mishra
O3b Networks

Email: mishra.ashesh@gmail.com

Ankur Saxena
Ciena Corporation
3939 North First Street
San Jose, CA 95134
USA

Email: ankurpsaxena@gmail.com

Alan DeKok
Network RADIUS SARL
100 Centrepointe Drive #200
Ottowa, ON K2G 6B1
Canada

Email: aland@freeradius.org