Security Assessment of the IPv6 Flow Label
draft-gont-6man-flowlabel-security-03

Abstract

   This document discusses the security implications of the IPv6 "Flow
   Label" header field, and analyzes possible schemes for selecting the
   Flow Label value of IPv6 packets.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The flow label is a 20-bit field that allows a source to label
sequences of packets for which it requests special handling by IPv6
routers (e.g., non-default quality of service). It is specified in
[RFC6437].   RFC 6438 [RFC6438] specifies the use of the Flow Label
for Equal Cost Multipath Routing and Link Aggregation in Tunnels.

   The FLow Label was originally loosely specified in RFC 2460
   [RFC2460], and then later refined in [RFC3697]. Its specification
   has been recently revised by RFC 6437 [RFC6437]. [RFC6436]
   discusses the rationale for the update to the Flow Label
   specification in [RFC6437].

Section 2Section 2.1[RFC6437]Section 2.2[RFC6437]

2.   Vulnerability analysis

2.1.   RFC3697-compliant implementations

2.1.1.   DoS resulting from verification of Flow Label consistency

   [RFC2460] states that hosts and routers that do not support the
   functions of the Flow Label field are required to set this field to
   zero, pass the field unchanged when forwarding a packet, and ignore
   the field when forwarding a packet.

   If any packet belonging to a flow includes a Hop-by-Hop Options
   header, then all packets of that flow must contain a Hop-by-Hop
   Options header with the same contents (excluding the Next Header
   field of the Hop-by-Hop Options header). If any packet belonging to
   a flow contains a Routing Header, then all packets of that flow must
   have the same contents in all Extension Headers up to and including
   the Routing Header (but excluding the Next Header field of the
   Routing header).

   Appendix A of [RFC2460] states that routers and destinations are
   permitted, but not required, to verify that these conditions are
   satisfied. In order to perform this verification, the Hop-by-Hop
   Options header (and possibly the Destination Options header and the
   Routing header) used for the packets of each of the different flows
   should be kept in memory. This requirement, by itself, would open
   the door to at least two Denial of Service (DoS) vulnerabilities.

   Firstly, an attacker could forge a large number of packets with
   different values for the Flow Label field, thus leading the attacked
   system to record the Hop-by-Hop Options header (and possibly a
   Destination Options header and a Routing header) for each of the
   forged "flows". This might exhaust the attacked system's memory, and
   thus lead to a system crash or a Denial of Service (DoS) to
   legitimate flows.

   If a control protocol is used to convey the special handling for the
   flow, then such information could be recorded only upon receipt of
   the first packet belonging to a flow for which this "flow setup" has
   been completed. And thus this particular threat would be somewhat
   mitigated.

   If the nature of the special handling for the flow were carried in a
   hop-by-hop option, the system performing the aforementioned
   information would have to record the Hop-by-Hop Options header (and
   possibly a Destination Options header and a Routing header) of each
   packet belonging to a "new" flow. As a result, an attacker could
   simply send a large number of forged packets belonging to different

flows, thus leading the attacked system to tie memory for each of
these forged flows. This might exhaust the attacked system's memory,
and thus lead to a system crash or the Denial of Service (DoS) to
legitimate flows.

Secondly, rather than aiming at exhausting system resources, an
attacker could send forged packets with the intent of having the
attacked system record their headers, so that future legitimate
packets are discarded as a result of not including the same extension
headers that had been recorded upon receipt of the forged packets.

Therefore, while this verification might be of help to mitigate some
blind attacks by obfuscation, we believe the drawbacks of performing
such verification outweigh the potential benefits, and thus recommend
systems to not perform such verification.

2.1.2. Covert channels

As virtually every protocol header field, the Flow Label could be
used to implement a covert channel. In those network environments in
which the Flow Label is not used, middle-boxes such as packet
scrubbers could eliminate this covert channel by resetting the Flow
Label with zero, at the expense of disabling the use of the Flow
Label for e.g., load-balancing. Such a policy should be carefully
evaluated before being enabled, as it would prevent the deployment of
any legitimate technology that makes use of the Flow Label field.

It should be stress that is very difficult to eliminate all covert
channels in a communications protocol, and thus the enforcement of
the aforementioned policy should only be applied after careful
evaluation.

2.1.3. QoS theft

If a network identifies flows that will receive a specific QoS by
means of the Flow Label, an attacker could forge the packets with
specific Flow Label values such that those packets receive that QoS
treatment.

2.1.4. Information Leaking

If a host selects the Flow Label values of outgoing packets such that
the resulting sequence of Flow Label values is predictable, this
could result in an information leakage. Specifically, if a host sets
the Flow Label value of outgoing packets from a system-wide counter,
the number of "outgoing flows" would be leaked. This could in turn
be used for purposes such as "stealth port scanning" (see Section 3.5
of [CPNI-IP]).

2.2. RFC6437-compliant implementations

The security-wise main changes introduced in [RFC6437] are:

o Since Section 6 and Appendix A of RFC 2460 has been essentially
   obsoleted, the revised specification does not describe any
   verification for consistency of the Flow Label values of different
   packets of the same "flow". Therefore, the vulnerability
   described in Section 2.1.1 has been eliminated.

o The revised specification recommends that Flow Label values are
   not easily predictable, and therefore the vulnerabilities
   described in Section 2.1.3 and Section 2.1.4 are mitigated.

Note: the issue of "covert channels" described in Section 2.1.2
remains essentially the same. That is, unless the Flow Label value
is rewritten, it may be exploited as a covert channel. However,
[RFC6437] mentions this issue, and notes how this could be mitigated
in those network scenarios in which covert channels might be a
concern.

3. Selecting Flow Label values

   [RFC6437] specifies the requirements for a Flow Label generation
   algorithm. Essentially:

   o The Flow Label value must not be easily predictable by a third-
      party.

   o Flow Labels (together with the Source Address and the Destination
      Address) are meant to uniquely identify a packet "flow". Hence,
      to the extent that is possible each flow should result in a unique
      {Source Address, Destination Address, Flow Label} set of values at
      any given time.

   o In order to help with the use of the Flow Label for Equal Cost
      Multipath Routing (ECMP) and Link Aggregation (LAG) in Tunnels,
      Flow Labels should (ideally) have a uniform distribution.

   Section 3.1 specifies the RECOMMENDED algorithm for selecting Flow
   Label values. Section 3.2 specifies an alternative algorithm that
   MAY be used by those implementations concerned about the Flow Label
   reuse frequency of the RECOMMENDED algorithm.

3.1. Recommended algorithm

   Considering that the Flow Label is a 20-bit field, that Flow Label
   values must be unique for each (Source Address, Destination Address)
   pair at any given time, and that [RFC6437] relaxed the requirement of
   uniqueness that was enforced in [RFC3697], we RECOMMEND that the Flow
   Label of each flo be selected acording to a PRNG. That is, each Flow
   Label would be selected with:

                          Flow Label = random()

   where:

   random():
      Is a a Pseudo-Random Number Generator (PRNG).

3.2. Alternative Algorithm

   Implemenatations concerned with the Flow Label reuse frequency of the
   algorithm specified in Section 3.1 MAY use the following alternative
   scheme, which aims at minimizing the Flow Label reuse frequency by
   producing per-destination monotonically-increasing Flow Label values.

   Flow Label = F(Source Address, Destination Address, Secret Key2) +
                table[G(Source Address, Destination Address, Secret Key1)]

where:

table:
   Is an array of counters that are initialized to random values upon
   system bottstrap. The larger the array, the greater the
   separation of the "increments" space.

F():
   Is a hash function that should take as input both the Source
   Address and the Destination Address of the flow, and a secret key.
   The result of F() should not be computable without knowledge of
   all the parameters of the hash function.

      If random numbers are used as the only source of the secret
      key, they should be chosen in accordance with the
      recommendations given in [RFC4086].

G():
   Is a hash function that should take as input both the Source
   Address and the Destination Address of the flow, and a secret key.
   The result of G() should not be computable without knowledge of
   all the parameters of the hash function.

      If random numbers are used as the only source of the secret
      key, they should be chosen in accordance with the
      recommendations given in [RFC4086].

This scheme should be invoked when a new flow is to be created (e.g.,
when a new TCP connection is to be created). Once a Flow Label value
for such flow is selected, the Flow Label field of all the IPv6
packets corresponding to that flow would be set to the selected value
(until the flow is terminated).

The following figure illustrates this algorithm in pseudo-code:

```
        /* Initialization at system boot time */
        for(i = 0; i < TABLE_LENGTH; i++)
            table[i] = random();

        /* Flow Label selection function */
        offset = F(local_IP, remote_IP, secret_key1);
        index = G(local_IP, remote_IP, secret_key2);
        count = 1048576;

        do {
            flowlabel = (offset + table[index]) % 1048576;
            table[index]++;

            if(three-tuple is unique)
                return flowlabel;

            count--;

        } while (count > 0);

        /* Set the Flow Label to 0 if there is no
           unused Flow Label                      */

        return 0;
```

                              Figure 1

   The following table shows a sample output of this algorithm:

```
    +-----+------------+------------+------+----+------+------------+
    | Nr. |  Src. Addr. |  Dst. Addr. | off. |  i | t[i] | Flow Label |
    +-----+------------+------------+------+----+------+------------+
    |  #1 | 2001:db8::1 | 2001:db8::2 | 1000 | 10 |   5  |    1005    |
    +-----+------------+------------+------+----+------+------------+
    |  #2 | 2001:db8::1 | 2001:db8::2 | 1000 | 10 |   6  |    1006    |
    +-----+------------+------------+------+----+------+------------+
    |  #3 | 2001:db8::1 | 2001:db8::4 | 4500 | 15 |  10  |    4510    |
    +-----+------------+------------+------+----+------+------------+
    |  #4 | 2001:db8::1 | 2001:db8::4 | 4500 | 15 |  11  |    4511    |
    +-----+------------+------------+------+----+------+------------+
    |  #5 | 2001:db8::1 | 2001:db8::2 | 1000 | 10 |   7  |    1007    |
    +-----+------------+------------+------+----+------+------------+
```

           Table 1: Sample output of the double-hash algorithm

3.2.1. Secret-key considerations

   Every complex manipulation (like MD5) is no more secure than the
   input values, and in the case of ephemeral ports, the secret key. If
   an attacker is aware of which cryptographic hash function is being
   used by the victim (which we should expect), and the attacker can
   obtain enough material (e.g. Flow Label values selected by the
   victim), the attacker may simply search the entire secret key space
   to find matches.

   To protect against this, the secret key should be of a reasonable
   length. Key lengths of 128 bits should be adequate.

   Another possible mechanism for protecting the secret key is to change
   it after some time. If the host platform is capable of producing
   reasonably good random data, the secret key can be changed
   automatically.

   Changing the secret will cause abrupt shifts in the selected Flow
   Label values, and consequently collisions may occur. That is, upon
   changing the secret, the "offset" value used for each tuple (Source
   Address, Destination Address) will be different from that computed
   with the previous secret, thus possibly leading to the selection of a
   Flow Label value recently used for the same tuple (Source Address,
   Destination Address).

   Thus the change in secret key should be done with consideration and
   could be performed whenever one of the following events occur:

   o The system is being bootstrapped.

   o Some predefined/random time has expired.

   o The secret has been used N times (i.e. we consider it insecure).

   o There is little traffic (the performance overhead of collisions is
      tolerated).

   o There is enough random data available to change the secret key
      (pseudo-random changes should not be done).

4. Security Considerations

   This document provides a security assessment of the IPv6 Flow Label
   header field, and possible strategies to mitigate them.

5. IANA Considerations

   There are no IANA registries within this document. The RFC-Editor
   can remove this section before publication of this document as an
   RFC.

6. Acknowledgements

The author would like to thank (in alphabetical order) Shane Amante,
Ran Atkinson, Steven Blake, and Brian Carpenter for providing
valuable feedback on earlier versions of this document.

The offset function used by the algorithm in Section 3.1 was inspired
by the mechanism proposed by Steven Bellovin in [RFC1948] for
defending against TCP sequence number attacks.

This document is heavily based on the document "Security Assessment
of the Internet Protocol version 6 (IPv6)" [CPNI-IPv6] written by
Fernando Gont on behalf of the UK Centre for the Protection of
National Infrastructure (CPNI).

Fernando Gont would like to thank CPNI (http://www.cpni.gov.uk) for
their continued support.

7. References

7.1. Normative References

[RFC2460]   Deering, S. and R. Hinden, "Internet Protocol, Version 6
            (IPv6) Specification", RFC 2460, December 1998.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3697]   Rajahalme, J., Conta, A., Carpenter, B., and S. Deering,
            "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC4086]   Eastlake, D., Schiller, J., and S. Crocker, "Randomness
            Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4301]   Kent, S. and K. Seo, "Security Architecture for the
            Internet Protocol", RFC 4301, December 2005.

[RFC6437]   Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,
            "IPv6 Flow Label Specification", RFC 6437, November 2011.

[RFC6438]   Carpenter, B. and S. Amante, "Using the IPv6 Flow Label
            for Equal Cost Multipath Routing and Link Aggregation in
            Tunnels", RFC 6438, November 2011.

7.2. Informative References

[FreeBSD]   The FreeBSD Project, "http://www.freebsd.org".

[RFC1948]   Bellovin, S., "Defending Against Sequence Number Attacks",
            RFC 1948, May 1996.

[I-D.blake-ipv6-flow-label-nonce]
            Blake, S., "Use of the IPv6 Flow Label as a Transport-
            Layer Nonce to Defend Against Off-Path Spoofing Attacks",
            draft-blake-ipv6-flow-label-nonce-02 (work in progress),
            October 2009.

[RFC6056]   Larsen, M. and F. Gont, "Recommendations for Transport-
            Protocol Port Randomization", BCP 156, RFC 6056,
            January 2011.

[RFC6436]   Amante, S., Carpenter, B., and S. Jiang, "Rationale for
            Update to the IPv6 Flow Label Specification", RFC 6436,
            November 2011.

[CPNI-TCP]

               Gont, F., "CPNI Technical Note 3/2009: Security Assessment
               of the Transmission Control Protocol (TCP)", http://
               www.cpni.gov.uk/Docs/tn-03-09-security-assessment-TCP.pdf,
               2009.

   [CPNI-IP]   Gont, F., "Security Assessment of the Internet Protocol",
                http://www.cpni.gov.uk/Docs/InternetProtocol.pdf, 2008.

   [CPNI-IPv6]
               Gont, F., "Security Assessment of the Internet Protocol
               version 6 (IPv6)", UK Centre for the Protection of
               National Infrastructure, (available on request).

Appendix A. Survey of Flow Label selection algorithms in use by some
            popular implementations

A.1. FreeBSD

   ?

A.2. Linux

   ?

A.3. NetBSD

   ?

A.4. OpenBSD

   ?

A.5. OpenSolaris

   ?

Appendix B. **Changes from previous versions of the draft (to be removed by the RFC Editor before publication of this document as a RFC**

B.1. **Changes from draft-gont-6man-flowlabel-security-02**

   o The document now recommends randomized Flow Labels as the default
      approach, and describes the hash-based approach as an alternative
      method to be used if there are concerns about the Flow Label reuse
      frequency.

   o Minor editorial changes.

B.2. **Changes from draft-gont-6man-flowlabel-security-01**

   o The document has been updated to contain an analysis of the
      revised Flow Label specification [RFC6437].

   o Minor editorial changes.

B.3. **Changes from draft-gont-6man-flowlabel-security-00**

   o Clarified *when* Flow Labels are selected, in response to Shane
      Amante's feedback.

Author's Address

    Fernando Gont
    UK Centre for the Protection of National Infrastructure

    Email: fernando@gont.com.ar
    URI: http://www.cpni.gov.uk