

IPv6 maintenance Working Group (6man)
Internet-Draft
Intended status: BCP
Expires: July 15, 2012

F. Gont
UK CPNI
January 12, 2012

Security Assessment of the IPv6 Flow Label
draft-gont-6man-flowlabel-security-02

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

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#). This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 <http://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 July 15, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Internet-Draft

Flow Label Security

January 2012

Table of Contents

1.	Introduction	3
2.	Vulnerability analysis	4
2.1.	RFC3697 -compliant implementations	4
2.1.1.	DoS resulting from verification of Flow Label consistency	4
2.1.2.	Covert channels	5
2.1.3.	QoS theft	5
2.1.4.	Information Leaking	5
2.2.	RFC6437 -compliant implementations	6
3.	Selecting Flow Label values	7
4.	Secret-key considerations	12
5.	Security Considerations	13
6.	IANA Considerations	14
7.	Acknowledgements	15
8.	References	16
8.1.	Normative References	16
8.2.	Informative References	16
Appendix A.	Survey of Flow Label selection algorithms in use by some popular implementations	18
A.1.	FreeBSD	18
A.2.	Linux	18
A.3.	NetBSD	18
A.4.	OpenBSD	18
A.5.	OpenSolaris	18
Appendix B.	Changes from previous versions of the draft (to be removed by the RFC Editor before publication of this document as a RFC	19
B.1.	Changes from draft-gont-6man-flowlabel-security-01	19
B.2.	Changes from draft-gont-6man-flowlabel-security-00	19
Author's Address	20

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.

It was originally loosely specified in [RFC 2460](#) [Deering and Hinden, 1998] 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 2](#)[Section 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 they all must be sent with the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header). If any of those packets contains a Routing Header, then all packets belonging to that flow must be originated with 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](#)]).

Gont

Expires July 15, 2012

[Page 5]

Internet-Draft

Flow Label Security

January 2012

[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.

These requirements are similar to those of TCP port numbers, TCP Initial Sequence Numbers, and TCP timestamps. [CPNI-TCP] provides a detailed discussion of the requirements for such TCP parameters, and a number of algorithms that could be used to meet those requirements.

Considering that the Flow Label is a 20-bit field, and that Flow Label values must be unique for each (Source Address, Destination Address) pair at any given time, it might make sense to simply randomize the Flow Label value for each new flow.

This has been proposed in [I-D.blake-ipv6-flow-label-nonce].

In order to reduce the chances of collisions of Flow Label values, while still providing obfuscation, we propose that the Flow Label for new IPv6 flows be selected according the following scheme:

Flow Label = counter + F(Source Address, Destination Address, Secret Key)

where:

counter:

Is a variable that is initialized to some arbitrary value, and is incremented once for each flow label value that is selected.

F():

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

This scheme should be used 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).

This scheme was originally proposed in [[RFC1948](#)] for the selection of TCP Initial Sequence Numbers, and later proposed for the selection of TCP ephemeral ports [[RFC6056](#)] and for the selection of TCP timestamps [[CPNI-TCP](#)].

This scheme separates the Flow Label space for each pair of (Source Address, Destination Address), resulting in a sequence of monotonically-increasing Flow Label values (with a pseudo-random origin) within each of those Flow Label spaces.

The following figure illustrates this algorithm in pseudo-code:

```
/* Initialization at system boot time */
counter = 0;

/* Flow Label selection function */
offset = F(local_IP, remote_IP, secret_key);

count = 1048576;

do {
    flowlabel = (offset + counter) % 1048576;
    counter++;

    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

This algorithm should be executed 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 table shows a sample output of this scheme:

Nr.	Src. Addr.	Dst. Addr.	offset	counter	Flow Label
#1	2001:db8::1	2001:db8::2	1000	0	1000
#2	2001:db8::1	2001:db8::2	1000	1	1001
#3	2001:db8::1	2001:db8::4	4500	2	4502
#4	2001:db8::1	2001:db8::4	4500	3	4503
#5	2001:db8::1	2001:db8::2	1000	4	1004

Table 1: Sample output of the simple-hash algorithm

This scheme can be further improved by separating the increment spaces, such that the selection of a Flow Label within one space does not necessarily cause a Flow Label value to be skipped in the other spaces. To perform a separation of the increment spaces, the global counter would be replaced with an array of counters as follows:

$$\text{Flow Label} = \text{table}[\text{G}(\text{Source Address}, \text{Destination Address}, \text{Secret Key1})] + \text{F}(\text{Source Address}, \text{Destination Address}, \text{Secret Key2})$$

where:

table:

Is an array of counters that are initialized to some arbitrary value. The larger the array, the greater the obfuscation.

F():

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

G():

Is a hash function that should take as input both the Source Address and the Destination Address, and a secret key. The result

of F should not be computable without knowledge of all the parameters of the hash function.

As with the previous algorithm, 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 2

This scheme does not strictly comply with the requirement that a Flow Label value must not be reassigned assigned to new flows with the same address pair within 120 seconds of the termination of the

previous flow. However, by minimizing the Flow Label reuse frequency, it is expected that in virtually all real network scenarios such a requirement will be met.

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 2: Sample output of the double-hash algorithm

We recommend the implementation of this algorithm for the selection of the Flow Label.

[4.](#) 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).

[5.](#) Security Considerations

This document provides a security assessment of the IPv6 Flow Label header field, and possible strategies to mitigate them. Additionally, it proposes an algorithm for selecting the Flow Label values at hosts that complies with the current specification of the the Flow Label field

If the local offset function $F()$ results in identical offsets for different inputs at greater frequency than would be expected by chance, the port-offset mechanism proposed in this document would have a reduced effect.

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

[6.](#) 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.

[7.](#) 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 in this document 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.

[8.](#) References

[8.1.](#) 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.

[8.2.](#) 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.

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

[CPNI-IP] Gont, F., "Security Assessment of the Internet Protocol", <http://www.cpmi.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-01](#)

- o The document has been updated to contain an analysis of the revised Flow Label specification [[RFC6437](#)].
- o Minor editorial changes.

[B.2](#). Changes from [draft-gont-6man-flowlabel-security-00](#)

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

Gont

Expires July 15, 2012

[Page 19]

Internet-Draft

Flow Label Security

January 2012

Author's Address

Fernando Gont

UK Centre for the Protection of National Infrastructure

Email: fernando@gont.com.ar

URI: <http://www.cpni.gov.uk>

