IPv6 maintenance Working Group
(6man)
Internet-Draft
Intended status: BCP
Expires: February 13, 2011

F. Gont
UK CPNI

August 12, 2010

Security Assessment of the IPv6 Flow Label
draft-gont-6man-flowlabel-security-00

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 was loosely specified in RFC 2460 [Deering and Hinden, 1998] and its specification was later refined in [RFC3697].

> While the Flow Label could be used for e.g., load-sharing purposes, the author is not aware of any deployed use cases.

2. Threat analysis

2.1. 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.

2.2. 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.

2.3. 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.

2.4. 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 leakeage. 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]).

3. Selecting Flow Label values

[RFC3697] specifies how the Flow Label should be used by the processing nodes. It states that "source nodes SHOULD assign each unrelated transport connection and application data stream to a new flow". It recommends the following requirements for the assignment policy [RFC 3697]:

- A source node MUST ensure that it does not unintentionally reuse Flow Label values it is currently using or has recently used when creating new flows. Flow Label values previously used with a specific pair of source and destination addresses MUST NOT be assigned to new flows with the same address pair within 120 seconds of the termination of the previous flow.

- The source node SHOULD provide the means for the applications and transport protocols to specify quarantine periods longer than the default 120 seconds for individual flows.

- To avoid accidental Flow Label value reuse, the source node SHOULD select new Flow Label values in a well-defined sequence (e.g., sequential or pseudo-random) and use an initial value that avoids reuse of recently used Flow Label values each time the system restarts. The initial value SHOULD be derived from a previous value stored in non-volatile memory, or in the absence of such history, a randomly generated initial value using techniques that produce good randomness properties SHOULD be used.

  These requirements are very 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.

A simple strategy for selecting Flow Label values such that they are not reused too quickly would be to select them according to a global counter. However, if such a scheme were used, it could possibly be exploited in a similar way to that in which predictable IPv4 Identification values can be exploited (see Section 3.5 of [CPNI-TCP]). Therefore, the Flow Label should be obfuscated so that the chances of an off-path attacker of guessing the Flow Label of future flows are reduced.

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 the Flow Label field of IPv6 packets to be set 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 the knowledge of all the parameters of the hash function.

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 [I-D.ietf-tsvwg-port-randomization] 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

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:

Flow Label = table[G(Source Address, Destination Address, Secret Key1)] + F(Source Address, Destination Address, 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 the 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 the knowledge of all the parameters of the hash function.

```
/* 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:

- The system is being bootstrapped.

- Some predefined/random time has expired.

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

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

- 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 these threats.

This document proposes an algorithm for selecting the Flow Label values at hosts that complies with the current specification of the the Flow Label field, such that some threats are mitigated.

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 Brian Carpenter for providing valuable feedback on earlier versions of this document.

This document is heavily based on the upcoming document "Security Implications of the Internet Protocol version 6 (IPv6)" [CPNI-IPv6].

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.

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.

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.

[I-D.ietf-tsvwg-port-randomization] Larsen, M. and F. Gont, "Transport Protocol Port Randomization Recommendations", draft-ietf-tsvwg-port-randomization-08 (work in progress), May 2010.

[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, (to be published).

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

Author's Address

Fernando Gont
UK Centre for the Protection of National Infrastructure

Email: fernando@gont.com.ar