Internet Engineering Task Force	S. Blake	
Internet-Draft	Extreme Networks	
Intended status: Standards Track	October 27, 2009	
Expires: April 30, 2010		

TOC

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

Status of this Memo

Copyright Notice

Abstract

TCP and other transport-layer protocols are vulnerable to spoofing attacks from off-path hosts. These attacks can be prevented through the use of cryptographic authentication. However, it is difficult to use cryptographic authentication in all circumstances. A variety of obfuscation techniques -- such as initial sequence number randomization and source port randomization -- increase the effort required of an attacker to successfully guess the packet header fields which uniquely identify a transport connection. This memo proposes the use of the IPv6 Flow Label field as a random, per-connection nonce value, to add entropy to the set of packet header fields used to identify a transport connection. This mechanism is easily implementable, allows for incremental deployment, and is fully compliant with the rules for Flow Label use defined in RFC 3697.

Table of Contents

TOC

1. Introduction

1.1. TCP's Vulnerability to Blind Spoofing Attacks

TOC

Recent effort has been directed towards identifying and reducing the vulnerability of TCP [RFC0793] (Postel, J., “Transmission Control Protocol,” September 1981.) to a variety of "blind" spoofed packet injection attacks from hosts that are off-path (i.e., not able to intercept communications between a pair of hosts) [RFC4953] (Touch, J., “Defending TCP Against Spoofing Attacks,” July 2007.)[RFC5082] (Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, “The Generalized TTL Security Mechanism (GTSM),” October 2007.) [I-D.ietf-tcpm-icmp-attacks] (Gont, F., “ICMP attacks against TCP,” March 2008.) [I-D.ietf-tcpm-tcpsecure] (Ramaiah, A., Stewart, R., and M. Dalal, “Improving TCP's Robustness to Blind In-Window Attacks,” July 2008.) [I-D.ietf-tsvwg-port-randomization] (Larsen, M. and F. Gont, “Port Randomization,” August 2008.). Off-path spoofing attacks against TCP require an attacker to correctly guess the 4-tuple of header fields <IP source address, TCP source port, IP destination address, TCP destination port> uniquely identifying a TCP connection, along with a valid (in-receive window) value for the 32-bit TCP sequence number. By correctly guessing values for these fields, an attacker is then able to inject ACK, DATA, RST, or SYN segments into a TCP connection, enabling throughput reduction, data corruption, or connection termination with a single correctly constructed packet. Similarly, by correctly guessing values for these fields, an attacker is able to forge ICMP messages to a host, with similar negative consequences [I-D.ietf-tcpm-icmp-attacks] (Gont, F., “ICMP attacks against TCP,” March 2008.).

Increased use of long-duration connections by applications, as well as faster access link speeds, increase the ability of attackers to transmit a sufficient number of spoof packets to successfully attack a connection, especially when either the destination or source ports are easily guessable. Cryptographic authentication mechanisms such as the TCP MD5 Authentication Option [RFC2385] (Heffernan, A., “Protection of BGP Sessions via the TCP MD5 Signature Option,” August 1998.), TCP Authentication Option [I-D.ietf-tcpm-tcp-auth-opt] (Touch, J., Mankin, A., and R. Bonica, “The TCP Authentication Option,” July 2008.), and IPsec [RFC4301] (Kent, S. and K. Seo, “Security Architecture for the Internet Protocol,” December 2005.) can secure against these attacks, as well as some on-path (man-in-the-middle) attacks. However, key management and computational overhead makes the deployment of cryptographic authentication prohibitively expensive in some environments and for some applications.

Network ingress filtering of IP source addresses has been widely deployed at network boundaries, significantly reducing the set of networks that a particular host can inject spoof packets into [RFC2827] (Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," May 2000.)[RFC3704] (Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks," March 2004.). But network ingress filtering is not universally deployed, leaving many networks vulnerable to spoofed packet attacks (including the attacker's network). Note also that network ingress filtering typically provides no protection against ICMP spoofing attacks, since the attacker does not need to spoof the IP source address in the ICMP packet header (only the IP destination address in the ICMP message payload).

Obfuscation techniques can be employed to increase the effort required of an attacker. Initial sequence number randomization [RFC1948] (Bellovin, S., "Defending Against Sequence Number Attacks," May 1996.) [I-D.ietf-tcpm-tcpsecure] (Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks," July 2008.) can be implemented by both the client (the host initiating a connection) and server. For typical window sizes of approximately 32 Kbytes, this technique forces an attacker to send approximately 57000 RST packets on average to reset a connection [RFC4953] (Touch, J., "Defending TCP Against Spoofing Attacks," July 2007.). Source port randomization [I-D.ietf-tsvwg-port-randomization] (Larsen, M. and F. Gont, "Port Randomization," August 2008.) can also be implemented by a client to increase the number of guesses an attacker must make to successfully attack a connection. This mechanism can provide an additional ~15 bits of entropy (depending on implementation). Source port randomization can also be used with other transport protocols.

Both obfuscation schemes are compliant with [RFC0793] (Postel, J., "Transmission Control Protocol," September 1981.), and are incrementally deployable. Both schemes used in combination introduce approximately 32 bits of entropy (~17 + ~15) with typical window sizes in use today. However, as access link speeds (and consequently, receive windows) increase in size, the amount of entropy declines just as the number of spoof packets an attacker can generate in a given interval increases. Therefore, the margin of protection provided by these obfuscation mechanisms will decrease over time.

1.2. IPv6 Flow Label

TOC

IPv6 [RFC2460] (Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," December 1998.) includes a 20-bit Flow Label field, which can be used by hosts to uniquely label a uni-directional sequence of packets from a host to a particular unicast, anycast, or multicast destination. The tuple of <IP source address, IP destination

address, Flow Label> is intended to uniquely identify a particular flow during its lifetime (plus a subsequent quarantine period). Rules for the generation and usage of Flow Label values are defined in [RFC3697] (Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, “IPv6 Flow Label Specification,” March 2004.). Because transport-layer port fields may be located at a variable offset within a packet due to IPv6 extension headers, or may be obscured due to encryption, the Flow Label provides a fixed field in the IPv6 header to facilitate flow classification in routers.

While originally intended to facilitate flow-specific packet handling in routers (e.g., QoS, fast switching), the Flow Label can also be used by hosts to uniquely label one or more transport connections. An originating host may select a random Flow Label value at the beginning of a connection, and continue to use it for the connection's duration. The host (or hosts for multicast) at the other end of the connection can record this Flow Label value, and use it as part of the connection demultiplexing key, while also labeling response packets with the same or a different Flow Label value. The originating host can similarly record the Flow Label value in response packets, and use it as part of its connection demultiplexing key. In this way an additional 20 bits of entropy is added to the set of header fields used to identify a transport connection. When used in addition to source port randomization, the total amount of entropy is approximately 34-35 bits. When TCP initial sequence number randomization is also used (i.e., in TCP), the entropy is increased to > 40 bits (even for large windows), making off-path snooping attacks impractical.

The Flow Label field is not included in the pseudo header checksum of any of the standard transport protocols. Corruption of the Flow Label value (that is not detected by any link-layer checksum) will be interpreted by the receiving host as evidence of an attack (rather than otherwise being ignored). The recommended response by the receiving host (to silently discard the packet) is the same as in the case of a packet with a checksum error.

The concept of labeling transport connections to prevent off-path spoofing attacks was proposed in [McGann05] (McGann, O. and D. Malone, “Flow Label Filtering Feasibility,” December 2005.), in the context of stateful firewalls. This scheme may be useful for other transport protocols such as SCTP [RFC4960] (Stewart, R., “Stream Control Transmission Protocol,” September 2007.), UDP [RFC0768] (Postel, J., “User Datagram Protocol,” August 1980.), UDP-Lite [RFC3828] (Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, “The Lightweight User Datagram Protocol (UDP-Lite),” July 2004.), DCCP [RFC4340] (Kohler, E., Handley, M., and S. Floyd, “Datagram Congestion Control Protocol (DCCP),” March 2006.), and RTP [RFC3550] (Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” July 2003.). Host implementations in compliance with [RFC3697] (Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, “IPv6 Flow Label Specification,” March 2004.) which do not allocate multiple flows to a single transport connection will

either label all packets in the connection with a Flow Label value of 0, or with some other constant. Therefore, this scheme is incrementally deployable by either peer in a connection. By introducing an incentive for hosts to begin utilizing the Flow Label, its utility for other network applications (e.g., as part of the ECMP load balancing key in routers) is improved.
Section 3 (Additional Requirements on Flow Label Value Generation and Use) specifies additional requirements on Flow Label generation. Section 4 (TCP Considerations) describes the use of this scheme with TCP. Section 5 (UDP Considerations) describes the use of this scheme with UDP and UDP-Lite. Section 6 (SCTP Considerations) describes the use of this scheme with SCTP. Section 7 (DCCP Considerations) describes the use of this scheme with DCCP. Section 8 (RTP Considerations) describes the use of this scheme with RTP over UDP or DCCP. Section 9 (NAT Considerations) describes the implications of IPv6 network address translation with respect to this scheme. Section 10 (Support for Transport Connection Sub-Flows) describes an alternative receiver behavior which allows support for multiple sub-flows within a transport connection.

---

2. Requirements Language

TOC

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 [RFC2119] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.).

---

3. Additional Requirements on Flow Label Value Generation and Use

TOC

[RFC3697] (Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification," March 2004.) specifies the rules governing use of the IPv6 Flow Label. The primary requirements relevant to our purpose are as follows (quoting directly):

- A Flow Label of zero is used to indicate packets not part of any flow.

- The Flow Label value set by the source MUST be delivered unchanged to the destination node(s).

*To enable Flow Label based classification, source nodes SHOULD assign each unrelated transport connection and application data stream to a new flow.

*The source node SHOULD be able to select unused Flow Label values for flows not requesting a specific value to be used.

*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 the new Flow Label value 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.

We wish to use the Flow Label value as an unguessable nonce. Hence, the following additional requirements are implied:

*Source hosts MUST assign each unrelated transport connection and application data stream to a new flow (i.e., with a non-zero Flow Label value).

*Source hosts MUST be able to select unused Flow Label values for flows not requesting a specific value to be used. The selected Flow Label value must remain constant for the duration of the flow.

*The Flow Label value MUST be practically unguessable, in a manner similar to the TCP source port or initial sequence number when they are randomized. A random number generator with good randomness properties (i.e., uniformly distributed) as specified in [RFC4086] (Eastlake, D., Schiller, J., and S. Crocker, “Randomness Requirements for Security,” June 2005.) MUST be used to generate Flow Label values not explicitly requested by an application.

*Flow Label state for a transport connection or application data stream MUST be cleaned-up by hosts as part of the transport connection/application data stream state clean-up.

*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 X seconds of the termination of the previous flow, where X is the maximum of either 120 seconds, or the duration for which transport connection state might linger at a host after traffic flow has ceased (e.g., TIME-WAIT state in TCP).

We assume that the requirement of [RFC3697] (Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, “IPv6 Flow Label Specification,” March 2004.) that "The Flow Label value set by the source MUST be delivered unchanged to the destination node(s)" applies also when the Flow Label value is 0. By implication we assume that intermediate nodes are not allowed to assign a packet to a flow, whether or not the source node did so.

For this particular application of the Flow Label field, no problem would be posed if multiple flows from a source host in unrelated transport connections/application data streams coincidentally shared the same Flow Label value, as long as the other previous requirements are adhered to. However, the prohibition in [RFC3697] (Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, “IPv6 Flow Label Specification,” March 2004.) against simultaneous reuse of Flow Label values MUST be observed. Any application request to assign a specific Flow Label value already in use by another flow MUST be rejected.

Transport-specific requirements on Flow Label use are provided in the subsequent sections. However, as a general requirement, if a packet is received on a transport connection/application data stream with an unexpected Flow Label value, the packet MUST be silently discarded. If excessive Flow Label errors are received, this event SHOULD be logged.

ICMPv6 error messages contain the IPv6 header of the packet triggering the error [RFC4443] (Conta, A., Deering, S., and M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” March 2006.). A host receiving an ICMPv6 error message can validate the Flow Label value in the message payload to protect against ICMPv6 spoofing attacks [I-D.ietf-tcpm-icmp-attacks] (Gont, F., “ICMP attacks against TCP,” March 2008.).

Use of the Flow Label value as an unguessable nonce is incrementally deployable, whether a source host fails to support setting the Flow Label to a non-zero value, or a destination host fails to check its value. However, a Flow Label value of 0 is easily guessable, so resistance to spoofing attacks is not improved. Hosts SHOULD NOT rely on the mechanisms defined in this document when operating in high-threat environments.

The additional requirements given here for Flow Label generation and use are not in conflict with the requirements in [RFC3697] (Rajahalme,

J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification," March 2004.). Therefore, additional applications of the Flow Label field (e.g., for special QoS handling, load balancing, etc.) can be applied simultaneously with the use of the Flow Label field as a transport-layer nonce, so long as an additional application does not limit the permissible values of the Flow Label in any way which violates the requirement that the value be unpredictable.

---

4. TCP Considerations

TOC

Uni-directional traffic in a TCP connection is assumed to constitute a single flow, and hence MUST be assigned a unique Flow Label value by the source host; either explicitly by the application or automatically by the host's TCP/IP stack. Given the Flow Label value's additional use as a packet classification field in routers (for QoS or other purposes), there is no compelling reason to sub-divide traffic within a TCP connection into multiple flows for classification purposes.
For this application of the Flow Label field, it would not pose a problem if multiple TCP connections from a source host (whether to one or a multiple of destination hosts) reused the same Flow Label value. However, because of the additional uses of the Flow Label field, a host MUST NOT assign the same Flow Label value to multiple TCP connections. Both directions of traffic flow in a TCP connection are not required to share the same Flow Label value, nor are they prohibited from doing so. A host originating a TCP connection (client) selects a unique Flow Label value for the connection, which it stores as the OUTGOING_FLOW_ID in its Transport Control Block (TCB) for this connection. The Flow Label selection algorithm can run simultaneously with TCP source port and initial sequence number selection (e.g., by generating a single random variable and assigning bit-ranges within it to each field). This Flow Label value is included in the first (and subsequent) SYN packet(s) sent to the destination host (server). The server receiving the first SYN packet records the Flow Label value in its TCB for this connection as the INCOMING_FLOW_ID. The server then selects a unique Flow Label value for the connection, which it stores as the OUTGOING_FLOW_ID in the connection's TCB. It includes this Flow Label value in the first (and subsequent) SYN-ACK packet(s) sent to the client. The client receiving the SYN-ACK packet from the server records the Flow Label value in its TCB for this connection as the INCOMING_FLOW_ID. It sends all additional packets of the connection to the server using OUTGOING_FLOW_ID, and checks all incoming packets of the connection from the server to ensure that they include INCOMING_FLOW_ID. The server performs identical processing. Any packets received with a Flow Label value that does not match INCOMING_FLOW_ID MUST be silently discarded. If a significant number of such packets are received, this event SHOULD be logged.

When a server implements a SYN cache and/or SYN cookies, the Flow Label value used in the SYN-ACK packet MUST be consistent with the Flow Label value used in subsequent packets [McGann05] (McGann, O. and D. Malone, "Flow Label Filtering Feasibility," December 2005.) [RFC4987] (Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations," August 2007.). For the SYN cache case, this can be handled easily by including INCOMING_FLOW_ID and OUTGOING_FLOW_ID as part of each cache entry. For SYN cookies, one approach to satisfying the requirement without storing state is to derive the Flow Label value from a hash of the the connection 4-tuple plus a random secret [McGann05] (McGann, O. and D. Malone, "Flow Label Filtering Feasibility," December 2005.). Another approach is to use the Flow Label value received in the SYN (INCOMING_FLOW_ID) as the Flow Label value in the SYN-ACK (OUTGOING_FLOW_ID). When the connection is established, the same Flow Label value will be used in both directions of traffic. This approach leaves a small window of vulnerability to spoofing before the connection is established.
[RFC0793] (Postel, J., "Transmission Control Protocol," September 1981.) specifies that a connection should remain in TIME_WAIT state for 2 * MSL (Maximum Segment Lifetime) seconds. [RFC0793] (Postel, J., "Transmission Control Protocol," September 1981.) specifies MSL as 120 seconds, although many implementations default to a lower value. The Flow Label value quarantine period MUST be no less than the maximum of either 2 * MSL for the connection, or 120 seconds. The specified behavior at the client and server will work even if either the client or server fails to set a non-zero outgoing Flow Label value, or check the incoming Flow Label value. However, resistance to spoofing attacks is not improved. Further, no mechanism for detecting whether a peer is supporting the Flow Label nonce is defined, although receipt of an initial packet with a non-zero Flow Label suggests that the sending host may support this specification. Therefore, some cryptographic authentication mechanism SHOULD be used when operating in a high-threat environment [RFC2385] (Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option," August 1998.) [I-D.ietf-tcpm-tcp-auth-opt] (Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option," July 2008.) [RFC4301] (Kent, S. and K. Seo, "Security Architecture for the Internet Protocol," December 2005.).

---

5. UDP Considerations

TOC

UDP is a connectionless protocol, which is also vulnerable to spoofing attacks. The level of vulnerability is specific to each application-layer protocol running over UDP. Source port randomization can be utilized with UDP, but UDP does not have sequence numbers, so it is arguably more vulnerable than TCP with source port and initial sequence

number randomization. With the exception of connected SOCk_DGRAM sockets, UDP/IP stacks (usually) do not maintain sufficient state to maintain INCOMING_FLOW_ID or OUTGOING_FLOW_ID values for each application data stream between a source host and a destination host or multicast group. Therefore, Flow Label generation and validation must happen at the application layer.

For purposes of discussion, we define a UDP connection as a flow of traffic matching the tuple <IP source address, UDP source port, IP destination/group address, UDP destination port>. Note that a UDP connection consists of uni-directional traffic flow between a pair of hosts, or between a host and the receivers of a multicast group. UDP applications MUST assign each connection to a unique flow, and hence MUST assign each connection a unique Flow Label value. One exception is where multiple application data streams are multiplexed onto the same address/port pairs. In this case UDP applications MUST assign application data streams to unique flows (as appropriate for the intended QoS or other handling), and MUST use application-layer demultiplexing to associate incoming data streams with flows. Maintenance of INCOMING_FLOW_ID and OUTGOING_FLOW_ID values for each flow MUST be provided by the application. Applications MUST check the Flow Label value of a received packet against INCOMING_FLOW_ID for the associated flow, and MUST silently discard the packet if the values do not match. If a significant number of such packets are received, this event SHOULD be logged. Note that an alternative to multiplexing multiple application data streams onto the same address/port pair is to utilize different source and/or destination port values for each data stream.

Note that the Flow Label nonce does not provide any additional protection for multicast applications. Source address spoofing is usually prevented through use of reverse path forwarding (RPF) checks as part of the multicast forwarding procedure, and in cases where RPF is not in use (e.g., in BIDIR-PIM) [RFC5015] (Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, “Bidirectional Protocol Independent Multicast (BIDIR-PIM),” October 2007.)[RFC5294] (Savola, P. and J. Lingard, “Host Threats to Protocol Independent Multicast (PIM),” August 2008.), the attacker can learn the Flow Label values used by one or more senders by joining the multicast group.

There is no standard, widely implemented sockets API for either setting the Flow Label value in outgoing packets, nor retrieving it in incoming packets [RFC3493] (Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, “Basic Socket Interface Extensions for IPv6,” February 2003.)[RFC3542] (Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, “Advanced Sockets Application Program Interface (API) for IPv6,” May 2003.). There is also no standard sockets API for specifying that a non-zero Flow Label value be used in outgoing packets. Therefore the requirements above cannot be satisfied, except where a non-standard API is available, or the functionality is provided automatically within the UDP/IP stack. It would be worthwhile to define a standard sockets API for Flow Label management.

One application where the use of the Flow Label as a nonce would be beneficial is in protection against blind DNS cache poisoning attacks [I-D.weaver-dnsext-comprehensive-resolver] (Weaver, N., “Comprehensive DNS Resolver Defenses Against Cache Poisoning,” September 2008.). If DNS queries are each assigned a unique Flow Label value, and if DNS servers send responses with an outgoing Flow Label value equal to the incoming Flow Label value received in the request, then the client can validate with high-probability that the request was generated by the targeted server, since the UDP source port, DNS transaction ID, and Flow Label together provide approximately 51 bits of entropy.
The procedures described above for UDP are equally applicable for UDP-Lite [RFC3828] (Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, “The Lightweight User Datagram Protocol (UDP-Lite),” July 2004.).

---

6. SCTP Considerations

TOC

Use of the Flow Label with SCTP will be discussed in a subsequent revision of this document.

---

7. DCCP Considerations

TOC

Use of the Flow Label with DCCP will be discussed in a subsequent revision of this document.

---

8. RTP Considerations

TOC

Use of the Flow Label with RTP applications will be discussed in a subsequent revision of this document.

---

9. NAT Considerations

TOC

IPv6 network address translators (NATs) are not common, and there is no widely accepted definition for their correct behavior. One proposal [I-D.mrw-behave-nat66] (Wasserman, M. and F. Baker, “IPv6-to-IPv6 Network Address Translation (NAT66),” March 2009.) assumes that the NAT66 device performs stateless one-to-one address translation between internal and external addresses. In this scenario there is no address multiplexing, and the Flow Label values SHOULD NOT be changed by the

NAT66 device. The same is true for protocols permitting locator translation, such as ILNP [I-D.rja-ilnp-intro] (Atkinson, R., “ILNP Concept of Operations,” February 2010.).
An IPv6 NAT device performing address multiplexing (e.g., a NAPT) must obey the same Flow Label rules in Section 3 (Additional Requirements on Flow Label Value Generation and Use) as any host; i.e., no two independent flows originating from the same translated address may share the same Flow Label value. Therefore, such a NAT device MUST modify the Flow Label value of any arriving flow of packets to ensure that it does not collide with any currently in-use Flow Label values originating from the same translated address.

---

10. Support for Transport Connection Sub-Flows

TOC

The procedures described in this specification mandate that each transport connection must be assigned to a new flow with a unique Flow Label value. This does not permit the use of multiple IPv6 flows within a TCP connection. Such a capability would be useful for some approaches to TCP multi-path, such as [I-D.van-beijnum-1e-mp-tcp] (Beijnum, I., “One-ended multipath TCP,” May 2009.), which use the same address/port pairs for all sub-flows of the connection, and which would like to utilize the IPv6 Flow Label as part of the ECMP load balancing key in routers.
To enable the instantiation of multiple IPv6 flows per-transport connection, while retaining the benefits of the Flow Label as a nonce, we propose the following alternative procedures:

- Source hosts MUST assign each flow within a single transport connection with an OUTGOING_FLOW_ID sharing the same value for the least-significant 16 bits.

- Destination hosts check on the least-significant 16 bits of INCOMING_FLOW_ID for each transport connection.

This modified procedure does not significantly weaken the overall strenght of Flow Label nonce mechanism (when combined with other obfuscation techniques), while enabling up to 16 sub-flows per-transport connection).

---

11. Acknowledgements

TOC

[McGann05] (McGann, O. and D. Malone, “Flow Label Filtering Feasibility,” December 2005.) describes the use of the Flow Label as a transport-layer nonce. If others are aware of when and where this

concept might have been discussed previously, please contact the author.

The author would like to thank Brian Carpenter, Gorry Fairhurst, Joe Touch, Bob Briscoe, Iljitsch van Beijnum, and Marcelo Bagnulo Braun for their valuable feedback.

This document was produced using the xml2rfc tool [RFC2629] (Rose, M., “Writing I-Ds and RFCs using XML,” June 1999.).

---

12. IANA Considerations

TOC

This memo includes no request to IANA.

---

13. Security Considerations

TOC

This memo describes the use of the IPv6 Flow Label as a transport-layer nonce to help protect transport connections and application data streams from blind spoofed packet injection attacks. Blind spoofed packet injection attacks have been described in several publications, and are well known to the community. This memo addresses the use of this mechanism with different transport protocols. This mechanism is only applicable for hosts communicating via the IPv6 protocol. This mechanism does not provide protection for any on-path (man-in-the-middle attacks); therefore, additional security mechanisms should be used in high threat environments.

---

14. References

TOC

14.1. Normative References

TOC

[RFC0768]	Postel, J., “User Datagram Protocol,” STD 6, RFC 768, August 1980 (TXT).
[RFC0793]	Postel, J., “Transmission Control Protocol,” STD 7, RFC 793, September 1981 (TXT).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2460]	

	Deering, S. and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC 2460, December 1998 (TXT, HTML, XML).
[RFC3550]	Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” STD 64, RFC 3550, July 2003 (TXT, PS, PDF).
[RFC3697]	Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, “IPv6 Flow Label Specification,” RFC 3697, March 2004 (TXT).
[RFC3828]	Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and G. Fairhurst, “The Lightweight User Datagram Protocol (UDP-Lite),” RFC 3828, July 2004 (TXT).
[RFC4086]	Eastlake, D., Schiller, J., and S. Crocker, “Randomness Requirements for Security,” BCP 106, RFC 4086, June 2005 (TXT).
[RFC4340]	Kohler, E., Handley, M., and S. Floyd, “Datagram Congestion Control Protocol (DCCP),” RFC 4340, March 2006 (TXT).
[RFC4443]	Conta, A., Deering, S., and M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” RFC 4443, March 2006 (TXT).
[RFC4960]	Stewart, R., “Stream Control Transmission Protocol,” RFC 4960, September 2007 (TXT).

14.2. Informative References

TOC

[RFC1948]	Bellovin, S., “Defending Against Sequence Number Attacks,” RFC 1948, May 1996 (TXT).
[RFC2385]	Heffernan, A., “Protection of BGP Sessions via the TCP MD5 Signature Option,” RFC 2385, August 1998 (TXT, HTML, XML).
[RFC2629]	Rose, M., “Writing I-Ds and RFCs using XML,” RFC 2629, June 1999 (TXT, HTML, XML).
[RFC2827]	Ferguson, P. and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing,” BCP 38, RFC 2827, May 2000 (TXT).
[RFC3493]	Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, “Basic Socket Interface Extensions for IPv6,” RFC 3493, February 2003 (TXT).
[RFC3542]	Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, “Advanced Sockets Application Program Interface (API) for IPv6,” RFC 3542, May 2003 (TXT).
[RFC3704]	Baker, F. and P. Savola, “Ingress Filtering for Multihomed Networks,” BCP 84, RFC 3704, March 2004 (TXT).
[RFC4301]	Kent, S. and K. Seo, “Security Architecture for the Internet Protocol,” RFC 4301, December 2005 (TXT).
[RFC4953]	Touch, J., “Defending TCP Against Spoofing Attacks,” RFC 4953, July 2007 (TXT).
[RFC4987]	Eddy, W., “TCP SYN Flooding Attacks and Common Mitigations,” RFC 4987, August 2007 (TXT).
[RFC5015]	Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, “Bidirectional Protocol Independent Multicast (BIDIR-PIM),” RFC 5015, October 2007 (TXT).
[RFC5082]	Gill, V., Heasley, J., Meyer, D., Savola, P., and C. Pignataro, “The Generalized TTL Security Mechanism (GTSM),” RFC 5082, October 2007 (TXT).
[RFC5294]	Savola, P. and J. Lingard, “Host Threats to Protocol Independent Multicast (PIM),” RFC 5294, August 2008 (TXT).
[I-D.ietf-tcpm-icmp-attacks]	Gont, F., “ICMP attacks against TCP,” draft-ietf-tcpm-icmp-attacks-03 (work in progress), March 2008 (TXT).
[I-D.ietf-tcpm-tcpsecure]	Ramaiah, A., Stewart, R., and M. Dalal, “Improving TCP's Robustness to Blind In-Window

	Attacks," draft-ietf-tcpm-tcpsecure-10 (work in progress), July 2008 (TXT).
[I-D.ietf-tsvwg-port-randomization]	Larsen, M. and F. Gont, "Port Randomization," draft-ietf-tsvwg-port-randomization-02 (work in progress), August 2008 (TXT).
[I-D.ietf-tcpm-tcp-auth-opt]	Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option," draft-ietf-tcpm-tcp-auth-opt-01 (work in progress), July 2008 (TXT).
[I-D.weaver-dnsext-comprehensive-resolver]	Weaver, N., "Comprehensive DNS Resolver Defenses Against Cache Poisoning," draft-weaver-dnsext-comprehensive-resolver-00 (work in progress), September 2008 (TXT).
[I-D.mrw-behave-nat66]	Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Address Translation (NAT66)," draft-mrw-behave-nat66-02 (work in progress), March 2009 (TXT).
[I-D.rja-ilnp-intro]	Atkinson, R., "ILNP Concept of Operations," draft-rja-ilnp-intro-03 (work in progress), February 2010 (TXT).
[I-D.van-beijnum-1e-mp-tcp]	Beijnum, I., "One-ended multipath TCP," draft-van-beijnum-1e-mp-tcp-00 (work in progress), May 2009 (TXT).
[McGann05]	McGann, O. and D. Malone, "Flow Label Filtering Feasibility," European Conference on Computer Network Defence, December 2005.

Author's Address

TOC

	Steven Blake
	Extreme Networks
	Pamlico Building One, Suite 100
	3306/08 E. NC Hwy 54
	RTP, NC 27709
	USA
Phone:	+1 919 884 3211
Email:	sblake@extremenetworks.com