Independent Submission
Request for Comments: 7974
Category: Informational
ISSN: 2070-1721

B. Williams
Akamai, Inc.
M. Boucadair
Orange
D. Wing

October 2016

An Experimental TCP Option for Host Identification

Abstract

Recent RFCs have discussed issues with host identification in IP address-sharing systems, such as address/prefix-sharing devices and application-layer proxies. Potential solutions for revealing a host identifier in shared address deployments have also been discussed. This memo describes the design, deployment, and privacy considerations for one such solution in operational use on the Internet today that uses a TCP option to transmit a host identifier.

Independent Submissions Editor Note

This Informational document specifies an experimental TCP HOST_ID option that is already fairly widely deployed. It discusses that option's privacy considerations in considerable detail and highlights the care providers need to exercise in any actual deployment. The Independent Submissions Editor has chosen to publish this document in the Independent Stream so that potential deployers and implementors can understand all its details, so as to produce implementations that will interwork properly with other (existing) deployments.

IESG Note

This proposal was previously proposed for adoption by the TCPM working group and rejected as being an undesirable technical design for both transport and privacy reasons. This document specifies a new TCP option that uses the shared experimental options format. The use of experimental TCP options is specified in [RFC6994] for TCP options "that are not yet eligible for assigned codepoints". As this proposal has been rejected by the IETF community, it is not eligible for the registration of a TCP option codepoint. It should be further noted that for experimental TCP options, it "is only appropriate to use these values in explicitly-configured experiments; they MUST NOT be shipped as defaults in implementations" [RFC4727]. The IESG also carried out a review as described in [RFC5742] and concluded that this proposal violates IETF principles expressed in [RFC7258] about pervasive monitoring as an attack and should therefore not be published without IETF review and IESG approval. (The process

described in [RFC5742] nonetheless allows the Independent Submissions Editor to publish, as has been chosen in this case.) Deployments of this proprietary TCP option may be widely viewed as undermining privacy and are likely to encounter issues with reliability of transport.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7974.

Copyright Notice

Table of Contents

1. Introduction

A broad range of issues associated with address sharing have been documented in [RFC6269] and [RFC7620]. In addition, [RFC6967] provides an analysis of various solutions to the problem of revealing the sending host's identifier (HOST_ID) information to the receiver, indicating that a solution using a TCP [RFC793] option for this purpose is among the possible approaches that could be applied with limited performance impact and a high success ratio. The purpose of this memo is to describe a TCP HOST_ID option that is currently deployed on the public Internet using the TCP experimental option codepoint, including discussion of related design, deployment, and privacy considerations.

Multiple documents have defined TCP options for the purpose of host identification: [REVEAL], [HOSTID], and [OVERLAYPATH]. Specification of multiple option formats to serve the purpose of host identification increases the burden for potential implementers and presents interoperability challenges as well, so the authors of those documents have worked together to define a common TCP option that supersedes the formats from those three documents. This memo describes a version of that common TCP option format that is currently in use on the public Internet.

The option defined in this memo uses the TCP experimental option codepoint sharing mechanism defined in [RFC6994]. One of the earlier specifications, [OVERLAYPATH], is associated with unauthorized use of a TCP option kind number, and moving to the TCP experimental option codepoint has allowed the authors of that document to correct their error.

1.1. Important Use Cases

The authors' implementations have primarily focused on the following address-sharing use cases in which currently deployed systems insert the HOST_ID option:

Carrier-Grade NAT (CGN): As defined in [RFC6888], [RFC6333], and other sources, a CGN allows multiple hosts connected to the public Internet to share a single Internet routable IPv4 address. One important characteristic of the CGN use case is that it modifies IP packets in-path, but does not serve as the endpoint for the associated TCP connections.

Application Proxy: As defined in [RFC1919], an application proxy splits a TCP connection into two segments, serving as an endpoint for each of the connections and relaying data flows between the connections.

Overlay Network: An overlay network is an Internet-based system providing security, optimization, or other services for data flows that transit the system. A network-layer overlay will sometimes act much like a CGN, in that packets transit the system with NAT being applied at the edge of the overlay. A transport-layer or application-layer overlay [RFC3135] will typically act much like an application proxy, in that the TCP connection will be segmented with the overlay network serving as an endpoint for each of the TCP connections.

In this set of sender use cases, the TCP option is applied to an individual TCP packet either at the connection endpoint (e.g., an application proxy or a transport-layer overlay network) or at an address-sharing middlebox (e.g., a CGN or a network-layer overlay network). See Section 4 for additional details about the types of devices that add the option to a TCP packet, as well as existing limitations on use of the option when it is inserted by an address-sharing middlebox, including issues related to packet fragmentation.

The existing receiver use cases considered by this memo include the following:

- Differentiating between attack and non-attack traffic when the source of the attack is sharing an address with non-attack traffic.

- Application of per-subscriber policies for resource utilization, etc., when multiple subscribers are sharing a common address.

- Improving server-side load-balancing decisions by allowing the load for multiple clients behind a shared address to be assigned to different servers, even when session affinity is required at the application layer.

In all of the above cases, differentiation between address-sharing clients is performed by a network function that does not process the application-layer protocol (e.g., HTTP) or the security protocol (e.g., TLS), because the action needs to be performed prior to decryption or parsing the application layer. Due to this, a solution implemented within the application layer or security protocol was considered unable to fully meet the receiver-side requirements. At the same time, as noted in [RFC6967], use of an IP option for this purpose has a low success rate. For these reasons, using a TCP option to deliver the host identifier was deemed by the authors to be an effective way to satisfy these specific use cases. See Section 5 for details about receiver-side interpretation of the option.

1.2. Document Goals

Publication of this memo is intended to serve multiple purposes.

First and foremost, this document intends to inform readers about a mechanism that is in broad use on the public Internet. The authors are each affiliated with companies that have implemented, tested, and/or deployed systems that use the HOST_ID option on the public Internet. Other systems might encounter packets that contain this TCP option, and this document is intended to help others understand the nature of the TCP option when it is encountered so they can make informed decisions about how to handle it.

The testing effort documented in [HOSTID] indicated that a TCP option could be used for host identification purposes without significant disruption of TCP connectivity to legacy servers and networks that do not support the option. It also showed how mechanisms available in existing TCP implementations could make use of such a TCP option for diagnostics and/or packet filtering. The authors' use of the TCP option on the public Internet has confirmed that it can be used effectively for our use cases, but it has also uncovered some interoperability issues associated with the option's use on the public Internet, especially regarding interactions with other TCP options that support new transport capability being specified within the IETF. Section 6 discusses those interactions and limitations and explains how our systems handle associated issues.

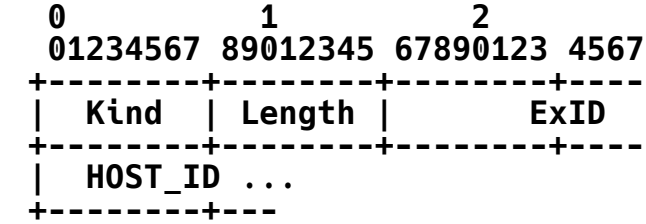
Discussions within the IETF have raised privacy concerns about the option's use, especially in regard to pervasive monitoring risks. Existing uses of the option limit the nature of the HOST_ID values that are used and the systems that insert them in order to mitigate pervasive monitoring risks. Sections 8 and 9 discuss the authors' assessments of the privacy and monitoring impact of this TCP option in its current uses and suggest behavior for some external systems when the option is encountered. Continued discussion following publication of this memo is expected to allow further refinement of requirements related to the values used to populate the option and how those values can be interpreted by the receiver. There is a trade-off between providing the expected functionality to the receiver and protecting the privacy of the sender, and continued assessment will be necessary in order to find the right balance.

2. Terminology

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].

3. Option Format

When used for host identification, the TCP experimental option uses the experiment identification mechanism described in [RFC6994] and has the following format and content.



Kind: The option kind value is 253.

Length: The length of the option is variable, based on the required size of the host identifier (e.g., a 2-octet HOST_ID will require a length of 6, while a 4-octet HOST_ID will require a length of 8).

ExID: The experiment ID value is 0x0348 (840).

HOST_ID: The host identifier is a value that can be used to differentiate among the various hosts sharing a common public IP address. See below for further discussion of this value.

4. Option Use

This section describes requirements associated with the use of the option, including expected option values, which hosts are allowed to include the option, and segments that include the option.

4.1. Option Values

The information conveyed in the HOST_ID option is intended to uniquely identify the sending host to the best capability of the machine that adds the option to the segment, while at the same time avoiding inclusion of information that does not assist this purpose. In addition, the option is not intended to be used to expose information about the sending host that could not be discovered by observing segments in transit on some portion of the Internet path between the sender and the receiver. Existing use cases have different requirements for receiver-side functionality, so this document attempts to provide a high degree of flexibility for the machine that adds the option to TCP segments.

The HOST_ID option value MUST correlate to IP addresses and/or TCP port numbers that were changed by the inserting host/device (i.e., some of the IP address and/or port number bits are used to generate the HOST_ID). Example values that satisfy this requirement include the following:

Unique ID: An inserting host/device could maintain a pool of locally unique ID values that are dynamically mapped to the unique source IP address values in use behind the host/device as a result of address sharing. This ID value would be meaningful only within the context of a specific shared IP address due to the local uniqueness characteristic. Such an ID value could be smaller than an IP address (e.g., 16 bits) in order to conserve TCP option space. This option is preferred because it does not increase IP address visibility on the forward side of the address-sharing system, and it SHOULD be used in cases where receiver-side requirements can be met without direct inclusion of the original IP address (e.g., some load-balancing uses).

IP Address/Subnet: An inserting host/device could simply populate the option value with the IP address value in use behind the host/device. In the case of IPv6 addresses, it could be difficult to include the full address due to TCP option space constraints, so the value would likely need to provide only a portion of the address (e.g., the first 64 bits).

IP Address and TCP Port: Some networks share public IP addresses among multiple subscribers with a portion of the TCP port number space being assigned to each subscriber [RFC6346]. When such a system is behind an address-sharing host/device, inclusion of both the IP address and the TCP port number will more uniquely identify the sending host than just the IP address on its own.

When multiple host identifiers are necessary (e.g., an IP address and a port number), the HOST_ID option is included multiple times within the packet, once for each identifier. While this approach significantly increases option space utilization when multiple identifiers are included, cases where only a single identifier is included are expected to be more common; thus, it is beneficial to optimize for those cases. Note that some middleboxes might reorder TCP options, so this method could be problematic if such a middlebox is in-path between the address-sharing system and the receiver. This has not proven to be a problem for existing use cases.

See Section 8 for discussion of privacy considerations related to selection of HOST_ID values.

4.2. Sending Host Requirements

The HOST_ID option MUST only be added by the sending host or any device involved in the forwarding path that changes IP addresses and/or TCP port numbers (e.g., NAT44 [RFC3022], L2-Aware NAT, DS-Lite Address Family Transition Router (AFTR) [RFC6333], IPv6-to-IPv6 Network Prefix Translation (NPTv6) [RFC6296], NAT64 [RFC6146], Dual-Stack Extra Lite [RFC6619], TCP Proxy, etc.). The HOST_ID option MUST NOT be added or modified en route by any device that does not modify IP addresses and/or TCP port numbers.

The sending host or intermediary device cannot determine whether the option value is used in a stateful manner by the receiver, nor can it determine whether SYN cookies are in use by the receiver. For this reason, the option MUST be included in all segments, both SYN and non-SYN segments, until return segments from the receiver positively indicate that the TCP connection is fully established on the receiver (e.g., the return segment either includes or acknowledges data).

4.2.1. Alternative SYN Cookie Support

The authors have also considered an alternative approach to SYN cookie support in which the receiving host (i.e., the host that accepts the TCP connection) echoes the option back to the sender in the SYN/ACK segment when a SYN cookie is being sent. This would allow the host sending HOST_ID to determine whether further inclusion of the option is necessary. This approach would have the benefit of not requiring inclusion of the option in non-SYN segments if SYN cookies had not been used. Unfortunately, this approach fails if the responding host itself does not support the option, since an intermediate node would have no way to determine that SYN cookies had been used.

4.2.2. Persistent TCP Connections

Some types of middleboxes (e.g., application proxy) open and maintain persistent TCP connections to regularly visited destinations in order to minimize the burden of connection establishment. Such middleboxes might use a single persistent TCP connection for multiple different client hosts over the life of the persistent connection.

This specification does not attempt to support the use of persistent TCP connections for multiple client hosts due to the perceived complexity of providing such support. Instead, the HOST_ID option is only allowed to be used at connection initiation. An inserting host/device that supports both the HOST_ID option and multi-client persistent TCP connections MUST NOT apply the HOST_ID option to TCP connections that could be used for multiple clients over the life of

the connection. If the HOST_ID option was sent during connection initiation, the inserting host/device MUST NOT reuse the connection for data flows originating from a client that would require a different HOST_ID value.

4.2.3. Packet Fragmentation

In order to avoid the overhead associated with in-path IP fragmentation, it is desirable for the inserting host/device to avoid including the HOST_ID option when IP fragmentation might be required. This is not a firm requirement though, because the HOST_ID option is only included in the first few packets of a TCP connection; thus, associated IP fragmentation will generally have minimal impact. The option SHOULD NOT be included in packets if the resulting packet would require local fragmentation.

It can be difficult to determine whether local fragmentation would be required. For example, in cases where multiple interfaces with different MTUs are in use, a local routing decision has to be made before the MTU can be determined, and in some systems, this decision could be made after TCP option handling is complete. Additionally, it could be true that inclusion of the option causes the packet to violate the path's MTU but the path's MTU has not been learned yet on the sending host/device.

In existing deployed systems, the impact of IP fragmentation that results from use of the option has been minimal.

4.3. Multiple In-Path HOST_ID Senders

The possibility exists that there could be multiple in-path hosts/devices configured to insert the HOST_ID option. For example, the client's TCP packets might first traverse a CGN device on their way to the edge of a public Internet overlay network. In order for the HOST_ID value to most uniquely identify the sender, it needs to represent both the identity observed by the CGN device (the subscriber's internal IP address, e.g., Shared Address Space [RFC6598]) and the identity observed by the overlay network (the shared address of the CGN device). The mechanism for handling the received HOST_ID value could vary depending upon the nature of the new HOST_ID value to be inserted, as described below.

The problem of multiple in-path HOST_ID senders has not been observed in existing deployed systems. For this reason, existing implementations do not consistently support this scenario. Some systems do not propagate forward the received HOST_ID option value in any way, while other systems follow the guidance described below.

An inserting host/device that uses the received packet's source IP address as the HOST_ID value (possibly along with the port) MUST propagate forward the HOST_ID value(s) from the received packet, since the source IP address and port only represent the previous in-path address-sharing device and do not represent the original sender. In the CGN-plus-overlay example, this means that the overlay will include both the CGN's HOST_ID value(s) and a HOST_ID with the source IP address received by the overlay.

An inserting host/device that sends a unique ID (as described in Section 4.1) has two options for how to handle the HOST_ID value(s) from the received packet:

1. A host/device that sends a unique ID MAY strip the received HOST_ID option and insert its own option, provided that it uses the received HOST_ID value as a differentiator for selecting the unique ID. What this means in the CGN-plus-overlay example above is that the overlay is allowed to drop the HOST_ID value inserted by the CGN provided that the HOST_ID value selected by the overlay represents both the CGN itself and the HOST_ID value inserted by the CGN.

2. A host/device that sends a unique ID MAY instead select a unique ID that represents only the previous in-path address-sharing host/device and propagate forward the HOST_ID value inserted by the previous host/device. In the CGN-plus-overlay example, this means that the overlay would include both the CGN's HOST_ID value and a HOST_ID with a unique ID of its own that was selected to represent the CGN's shared address.

An inserting host/device that sends a unique ID MUST use one of the above two mechanisms.

5. Option Interpretation

Due to the variable nature of the option value, it is not possible for the receiving machine to reliably determine the value type from the option itself. For this reason, a receiving host/device SHOULD interpret the option value as an opaque identifier.

This specification allows the inserting host/device to provide multiple HOST_ID options. The order of appearance of TCP options could be modified by some middleboxes, so receivers SHOULD NOT rely on option order to provide additional meaning to the individual options. Instead, when multiple HOST_ID options are present, their values SHOULD be concatenated together in the order in which they appear in the packet and treated as a single large identifier.

For both of the receiver requirements discussed above, this specification uses SHOULD rather than MUST because reliable interpretation and ordering of options could be possible if the inserting host and the interpreting host are under common administrative control and integrity-protect communication between the inserting host and the interpreting host. Mechanisms for signaling the value type(s) and integrity protection are not provided by this specification, and in their absence, the receiving host/device MUST interpret the option value(s) as a single opaque identifier.

6. Interaction with Other TCP Options

This section details how the HOST_ID option functions in conjunction with other TCP options.

6.1. Multipath TCP (MPTCP)

TCP provides for a maximum of 40 octets for TCP options. As discussed in Appendix A of MPTCP [RFC6824], a typical SYN from modern, popular operating systems contains several TCP options (MSS (Maximum Segment Size), window scale, SACK (selective acknowledgment) permitted, and timestamp), which consume 19-24 octets depending on word alignment of the options. The initial SYN from a multipath TCP client would consume an additional 16 octets.

HOST_ID needs at least 6 octets to be useful, so 9-21 octets are sufficient for many scenarios that benefit from HOST_ID. However, 4 octets are not enough space for the HOST_ID option. Thus, a TCP SYN containing all the typical TCP options (MSS, window scale, SACK permitted, and timestamp) and also containing multipath capable or multipath join as well as being word-aligned has insufficient space to accommodate HOST_ID. This means something has to give. The choices are either to avoid word alignment in that case (freeing 5 octets) or avoid adding the HOST_ID option. Each of these approaches is used in existing implementations and has been deemed acceptable for the associated use case.

6.2. Authentication Option (TCP-AO)

The TCP Authentication Option (TCP-AO) [RFC5925] is incompatible with address sharing due to the fact that it provides integrity protection of the source IP address. For this reason, the only use cases where it makes sense to combine TCP-AO and HOST_ID are those where the TCP-AO-NAT extension [RFC6978] is in use. Injecting a HOST_ID TCP option does not interfere with the use of TCP-AO-NAT because the TCP options are not included in the Message Authentication Code (MAC) calculation.

6.3. TCP Fast Open (TFO)

The TFO option [RFC7413] uses a zero-length cookie (total option length is 2 bytes) to request a TFO cookie for use on future connections. The server-generated TFO cookie is required to be at least 4 bytes long and allowed to be as long as 16 bytes (total option length is 6 to 18 bytes). The cookie request form of the option leaves enough room available in a SYN packet with the most commonly used options to accommodate the HOST_ID option, but a valid TFO cookie length longer than 13 bytes would prevent even the minimal 6-byte HOST_ID option from being included in the header.

There are multiple possibilities for allowing TFO and HOST_ID to be supported for the same connection, including:

- If the TFO implementation allows the cookie size to be configurable, the configured cookie size can be specifically selected to leave enough option space available in a typical TFO SYN packet to allow inclusion of the HOST_ID option.

- If the TFO implementation provides explicit support for the HOST_ID option, it can be designed to use a shorter cookie length when the HOST_ID option is present in the TFO cookie request SYN.

Reducing the TFO cookie size in order to include the HOST_ID option could have unacceptable security implications, so existing deployed systems that use the HOST_ID option consider TFO and HOST_ID to be mutually exclusive and do not support the use of both options on the same TCP connection.

It should also be noted that the presence of data in a TFO SYN increases the likelihood that there will be no space available in the SYN packet to support inclusion of the HOST_ID option without IP fragmentation, even if there is enough room in the TCP option space. This is an additional reason that the existing system considers TFO and HOST_ID to be mutually exclusive.

7. Security Considerations

Security (including privacy) considerations common to all HOST_ID solutions are discussed in [RFC6967].

The content of the HOST_ID option SHOULD NOT be used for purposes that require a trust relationship between the sender and the receiver (e.g., billing and/or subscriber policy enforcement). This requirement uses SHOULD rather than MUST because reliable interpretation of options could be possible if the inserting host and the interpreting host are under common administrative control and

integrity-protect communication between the inserting host and the interpreting host. Mechanisms for signaling the value type(s) and integrity protection are not provided by this specification, and in their absence, the receiving host/device MUST NOT use the HOST_ID value for purposes that require a trust relationship.

Note that the above trust requirement applies equally to HOST_ID option values propagated forward from a previous in-path host as described in Section 4.3. In other words, if the trust mechanism does not apply to all option values in the packet, then none of the HOST_ID values can be considered trusted, and the receiving host/device MUST NOT use any of the HOST_ID values for purposes that require a trust relationship. An inserting host/device that has such a trust relationship MUST NOT propagate forward an untrusted HOST_ID in such a way as to allow it to be considered trusted.

When the receiving network uses the values provided by the option in a way that does not require trust (e.g., maintaining session affinity in a load-balancing system), then use of a mechanism to enforce the trust relationship is OPTIONAL.

8. Privacy Considerations

Sending a TCP SYN across the public Internet necessarily discloses the public IP address of the sending host. When an intermediate address-sharing device is deployed on the public Internet, anonymity of the hosts using the device will be increased, with hosts represented by multiple source IP addresses on the ingress side of the device using a single source IP address on the egress side. The HOST_ID TCP option removes that increased anonymity, taking information that was already visible in TCP packets on the public Internet on the ingress side of the address-sharing device and making it available on the egress side of the device as well. In some cases, an explicit purpose of the address-sharing device is anonymity, in which case use of the HOST_ID TCP option would be incompatible with the purpose of the device.

A NAT device used to provide interoperability between a local area network (LAN) using private [RFC1918] IP addresses and the public Internet is sometimes specifically intended to provide anonymity for the LAN clients as described in the above paragraph. For this reason, address-sharing devices at the border between a private LAN and the public Internet MUST NOT insert the HOST_ID option.

The HOST_ID option MUST NOT be used to provide client geographic or network location information that was not publicly visible in IP packets for the TCP flows processed by the inserting host. For

example, the client's IP address MAY be used as the HOST_ID option value, but any geographic or network location information derived from the client's IP address MUST NOT be used as the HOST_ID value.

The HOST_ID option MAY provide differentiating information that is locally unique such that individual TCP flows processed by the inserting host can be reliably identified. The HOST_ID option MUST NOT provide client identification information that was not publicly visible in IP packets for the TCP flows processed by the inserting host, such as subscriber information linked to the IP address.

The HOST_ID value MUST be changed whenever the subscriber IP address changes. This requirement ensures that the HOST_ID option does not introduce a new globally unique identifier that persists across subscriber IP address changes.

The HOST_ID option MUST be stripped from IP packets traversing middleboxes that provide network-based anonymity services.

9. Pervasive Monitoring (PM) Considerations

[RFC7258] provides the following guidance: "Those developing IETF specifications need to be able to describe how they have considered PM, and, if the attack is relevant to the work to be published, be able to justify related design decisions." Legitimate concerns about host identification have been raised within the IETF. The authors of this memo have attempted to address those concerns by providing details about the nature of the HOST_ID values and the types of middleboxes that should and should not include the HOST_ID option in TCP headers, which describes limitations already imposed by existing deployed systems. This section is intended to highlight some particularly important aspects of this design and the related guidance/limitations that are relevant to the pervasive monitoring discussion.

When a generated identifier is used, this document prohibits the address-sharing device from using globally unique or permanent identifiers. Only locally unique identifiers are allowed. As with persistent IP addresses, persistent HOST_ID values could facilitate user tracking and are therefore prohibited. The specific requirements for permissible HOST_ID values are discussed in Sections 8 and 4.1.

This specification does not target exposing a host beyond what the original packet, issued from that host, would have already exposed on the public Internet without introduction of the option. The option is intended only to carry forward information that was conveyed to the address-sharing device in the original packet, and HOST_ID option

values that do not match this description are prohibited by requirements discussed in Section 8. This design does not allow the HOST_ID option to carry personally identifiable information, geographic location identifiers, or any other information that is not available in the wire format of the associated TCP/IP headers.

This document's guidance on option values is followed in the existing deployed system. Thus, the volatility of the information conveyed in a HOST_ID option is similar to that of the public, subscriber IP address. A distinct HOST_ID is used by the address-sharing function when the host reboots or gets a new public IP address from the subscriber network.

The described TCP option allows network identification to a similar level as the first 64 bits of an IPv6 address. That is, the server can use the bits of the TCP option to help identify a host behind an address-sharing device, in much the same way the server would use the host's IPv6 network address if the client and server were using IPv6 end to end.

Some address-sharing middleboxes on the public Internet have the express intention of providing originator anonymity. Publication of this document can help such middleboxes recognize the associated risk and take action to mitigate it (e.g., by stripping or modifying the option value).

10. IANA Considerations

This document specifies a new TCP option (HOST_ID) that uses the shared experimental options format [RFC6994], with ExID in network-standard byte order. IANA has registered HOST_ID (0x0348) in the "TCP Experimental Option Experiment Identifiers (TCP ExIDs)" registry.

11. References

11.1. Normative References

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <http://www.rfc-editor.org/info/rfc4727>.

[RFC5742] Alvestrand, H. and R. Housley, "IESG Procedures for Handling of Independent and IRTF Stream Submissions", BCP 92, RFC 5742, DOI 10.17487/RFC5742, December 2009, <http://www.rfc-editor.org/info/rfc5742>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

11.2. Informative References

[HOSTID] Abdo, E., Boucadair, M., and J. Queiroz, "HOST_ID TCP Options: Implementation & Preliminary Test Results", Work in Progress, draft-abdo-hostid-tcpopt-implementation-03, July 2012.

[OVERLAYPATH] Williams, B., "Overlay Path Option for IP and TCP", Work in Progress, draft-williams-overlaypath-ip-tcp-rfc-04, June 2013.

[REVEAL] Yourtchenko, A. and D. Wing, "Revealing hosts sharing an IP address using TCP option", Work in Progress, draft-wing-nat-reveal-option-03, December 2011.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[RFC1919] Chatel, M., "Classical versus Transparent IP Proxies", RFC 1919, DOI 10.17487/RFC1919, March 1996, <http://www.rfc-editor.org/info/rfc1919>.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <http://www.rfc-editor.org/info/rfc3022>.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <http://www.rfc-editor.org/info/rfc3135>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <http://www.rfc-editor.org/info/rfc5925>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <http://www.rfc-editor.org/info/rfc6146>.

[RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, DOI 10.17487/RFC6269, June 2011, <http://www.rfc-editor.org/info/rfc6269>.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <http://www.rfc-editor.org/info/rfc6296>.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, DOI 10.17487/RFC6333, August 2011, <http://www.rfc-editor.org/info/rfc6333>.

[RFC6346] Bush, R., Ed., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, DOI 10.17487/RFC6346, August 2011, <http://www.rfc-editor.org/info/rfc6346>.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, DOI 10.17487/RFC6598, April 2012, <http://www.rfc-editor.org/info/rfc6598>.

[RFC6619] Arkko, J., Eggert, L., and M. Townsley, "Scalable Operation of Address Translators with Per-Interface Bindings", RFC 6619, DOI 10.17487/RFC6619, June 2012, <http://www.rfc-editor.org/info/rfc6619>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888, April 2013, <http://www.rfc-editor.org/info/rfc6888>.

[RFC6967] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Potential Solutions for Revealing a Host Identifier (HOST_ID) in Shared Address Deployments", RFC 6967, DOI 10.17487/RFC6967, June 2013, <http://www.rfc-editor.org/info/rfc6967>.

[RFC6978] Touch, J., "A TCP Authentication Option Extension for NAT Traversal", RFC 6978, DOI 10.17487/RFC6978, July 2013, <http://www.rfc-editor.org/info/rfc6978>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

[RFC7620] Boucadair, M., Ed., Chatras, B., Reddy, T., Williams, B., and B. Sarikaya, "Scenarios with Host Identification Complications", RFC 7620, DOI 10.17487/RFC7620, August 2015, <http://www.rfc-editor.org/info/rfc7620>.

Acknowledgements

Many thanks to W. Eddy, Y. Nishida, T. Reddy, M. Scharf, J. Touch, A. Zimmermann, and A. Falk for their comments.

Authors' Addresses

Brandon Williams
Akamai, Inc.
8 Cambridge Center
Cambridge, MA 02142
United States of America

Email: brandon.williams@akamai.com

Mohamed Boucadair
Orange

Email: mohamed.boucadair@orange.com

Dan Wing

Email: dwing-ietf@fuggles.com