Network Working Group
Internet-Draft
Updates: 2460 (if approved)
Intended status: Standards Track
Expires: July 21, 2013

M. Eubanks
AmericaFree.TV LLC
P. Chimento
Johns Hopkins University Applied Physics Laboratory
M. Westerlund
Ericsson

January 17, 2013

IPv6 and UDP Checksums for Tunneled Packets
draft-ietf-6man-udpchecksums-07

Abstract

This document provides an update of the Internet Protocol version 6 (IPv6) specification (RFC2460) to improve the performance in the use case where a tunnel protocol uses UDP with IPv6 to tunnel packets. The performance improvement is obtained by relaxing the IPv6 UDP checksum requirement for any suitable tunnel protocol where header information is protected on the "inner" packet being carried. This relaxation removes the overhead associated with the computation of UDP checksums on IPv6 packets used to carry tunnel protocols. The specification describes how the IPv6 UDP checksum requirement can be relaxed for the situation where the encapsulated packet itself contains a checksum. The limitations and risks of this approach are described, and restrictions specified on the use of the method.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 21, 2013.

Copyright Notice

Table of Contents

1. Introduction

This work constitutes an update of the Internet Protocol Version 6 (IPv6) Specification [RFC2460], in the use case where a tunnel protocol uses UDP with IPv6 to tunnel packets. With the rapid growth of the Internet, tunnel protocols have become increasingly important to enable the deployment of new protocols. Tunnel protocols can be deployed rapidly, while the time to upgrade and deploy a critical mass of routers, middleboxes and hosts on the global Internet for a new protocol is now measured in decades. At the same time, the increasing use of firewalls and other security-related middleboxes means that truly new tunnel protocols, with new protocol numbers, are also unlikely to be deployable in a reasonable time frame, which has resulted in an increasing interest in and use of UDP-based tunnel protocols. In such protocols, there is an encapsulated "inner" packet, and the "outer" packet carrying the tunneled inner packet is a UDP packet, which can pass through firewalls and other middleboxes that perform filtering that is a fact of life on the current Internet.

Tunnel endpoints may be routers or middleboxes aggregating traffic from a number of tunnel users, therefore the computation of an additional checksum on the outer UDP packet may be seen as an unwarranted burden on nodes that implement a tunnel protocol, especially if the inner packet(s) are already protected by a checksum. In IPv4, there is a checksum over the IP packet header, and the checksum on the outer UDP packet may be set to zero. However in IPv6 there is no checksum in the IP header and RFC 2460 [RFC2460] explicitly states that IPv6 receivers MUST discard UDP packets with a zero checksum. So, while sending a UDP datagram with a zero checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. To improve support for IPv6 UDP tunnels, this document updates RFC 2460 to allow endpoints to use a zero UDP checksum under constrained situations (primarily IPv6 tunnel transports that carry checksum-protected packets), following the applicability statements and constraints in [I-D.ietf-6man-udpzero].

"Unicast UDP Usage Guidelines for Application Designers" [RFC5405] should be consulted when reading this specification. It discusses both UDP tunnels (Section 3.1.3) and the usage of checksums (Section 3.4).

While the origin of this specification is the problem raised by the draft titled "Automatic Multicast Tunnels", also known as "AMT" [I-D.ietf-mboned-auto-multicast] we expect it to have wide applicability. Since the first version of this document, the need for an efficient UDP tunneling mechanism has increased. Other IETF Working Groups, notably LISP [I-D.ietf-lisp] and Softwires [RFC5619]

have expressed a need to update the UDP checksum processing in RFC 2460. We therefore expect this update to be applicable in the future to other tunnel protocols specified by these and other IETF Working Groups.

2. Some Terminology

This document discusses only IPv6, since this problem does not exist for IPv4. Therefore all reference to 'IP' should be understood as a reference to IPv6.

The document uses the terms "tunneling" and "tunneled" as adjectives when describing packets. When we refer to 'tunneling packets' we refer to the outer packet header that provides the tunneling function. When we refer to 'tunneled packets' we refer to the inner packet, i.e., the packet being carried in the tunnel.

2.1. Requirements Language

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 RFC 2119 [RFC2119].

3. Problem Statement

When using tunnel protocols based on UDP, there can be both a benefit and a cost to computing and checking the UDP checksum of the outer (encapsulating) UDP transport header. In certain cases, reducing the forwarding cost is important, e.g., for nodes that perform the checksum in software the cost may outweigh the benefit. This document provides an update for usage of the UDP checksum with IPv6. The update is specified for use by a tunnel protocol that transports packets that are themselves protected by a checksum.

4. Discussion

"Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero] describes issues related to allowing UDP over IPv6 to have a valid zero UDP checksum and is the starting point for this discussion. Sections 4 and 5 of [I-D.ietf-6man-udpzero], respectively identify node implementation and usage requirements for datagrams sent and received with a zero UDP checksum. These introduce constraints on the usage of a zero checksum for UDP over IPv6. The remainder of this section analyses the use of general tunnels and motivates why tunnel protocols are

being permitted to use the method described in this update. Issues with middleboxes are also discussed.

4.1. Analysis of Corruption in Tunnel Context

This section analyzes the impact of the different corruption modes in the context of a tunnel protocol. It indicates what needs to be considered by the designer and user of a tunnel protocol to be robust. It also summarizes why use of a zero UDP checksum is thought to be safe for deployment.

1. Context (i.e., tunneling state) should be established by exchanging application Protocol Data Units (PDUs) carried in checksummed UDP datagrams or by other protocols with integrity protection against corruption. These control packets should also carry any negotiation required to enable the tunnel endpoint to accept UDP datagrams with a zero checksum and identify the set of ports that are used. It is important that the control traffic is robust against corruption because undetected errors can lead to long-lived and significant failures that may affect much more than the single packet that was corrupted.

2. Keep-alive datagrams with a zero UDP checksum should be sent to validate the network path, because the path between tunnel endpoints can change and therefore the set of middleboxes along the path may change during the life of an association. Paths with middleboxes that drop datagrams with a zero UDP checksum will drop these keep-alives. To enable the tunnel endpoints to discover and react to this behavior in a timely way, the keep-alive traffic should include datagrams with a non-zero checksum and datagrams with a zero checksum.

3. Receivers should attempt to detect corruption of the address information in an encapsulating packet. A robust tunnel protocol should track tunnel context based on the 5-tuple (tunneled protocol number, IPv6 source address, IPv6 destination address, UDP source port, UDP destination port). A corrupted datagram that arrives at a destination may be filtered based on this check.

   * If the datagram header matches the 5-tuple and the node has the zero checksum enabled for this port, the payload is matched to the wrong context. The tunneled packet will then be decapsulated and forwarded by the tunnel egress.

   * If a corrupted datagram matches a different 5-tuple and the zero checksum was enabled for the port, the datagram payload is matched to the wrong context, and may be processed by the

wrong tunnel protocol, if it also passes the verification of that protocol.

* If a corrupted datagram matches a 5-tuple and the zero checksum has not been enabled for this port, the datagram will be discarded.

When only the source information is corrupted, the datagram could arrive at the intended applications/protocol, which will process the datagram and try to match it against an existing tunnel context. The likelihood that a corrupted packet enters a valid context is reduced when the protocol restricts processing to only the source addresses with established contexts. When both source and destination fields are corrupted, this increases the likelihood of failing to match a context, with the exception of errors replacing one packet header with another one. In this case, it is possible that both packets are tunnelled and therefore the corrupted packet could match a previously defined context.

4. Receivers should attempt to detect corruption of source-fragmented encapsulating packets. A tunnel protocol may reassemble fragments associated with the wrong context at the right tunnel endpoint, or it may reassemble fragments associated with a context at the wrong tunnel endpoint, or corrupted fragments may be reassembled at the right context at the right tunnel endpoint. In each of these cases, the IPv6 length of the encapsulating header may be checked (though [I-D.ietf-6man-udpzero] points out the weakness in this check). In addition, if the encapsulated packet is protected by a transport (or other) checksum, these errors can be detected (with some probability).

5. Tunnel protocols using UDP have some advantages that reduce the risk for a corrupted tunnel packet reaching a destination that will receive it, compared to other applications. This results from processing by the network of the inner (tunneled) packet after being forwarded from the tunnel egress using a wrong context:

   * A tunneled packet may be forwarded to the wrong address domain, for example, a private address domain where the inner packet's address is not routable, or may fail a source address check, such as Unicast Reverse Path Forwarding [RFC2827], resulting in the packet being dropped.

   * The destination address of a tunneled packet may not at all be reachable from the delivered domain. For example, an Ethernet

frame where the destination MAC address is not present on the LAN segment that was reached.

* The type of the tunneled packet may prevent delivery. For example, an attempt to interpret an IP packet payload as an Ethernet frame, would likely to result in the packet being dropped as invalid.

* The tunneled packet checksum or integrity mechanism may detect corruption of the inner packet caused at the same time as corruption to the outer packet header. The resulting packet would likely be dropped as invalid.

These checks each significantly reduce the likelihood that a corrupted inner tunneled packet is finally delivered to a protocol listener that can be affected by the packet. While the methods do not guarantee correctness, they can reduce the risk of relaxing the UDP checksum requirement for a tunnel application using IPv6.

4.2. Limitation to Tunnel Protocols

This document describes the applicability of using a zero UDP checksum to support tunnel protocols. There are good motivations behind this and the arguments are provided here.

- Tunnels carry inner packets that have their own semantics, which may make any corruption less likely to reach the indicated destination and be accepted as a valid packet. This is true for IP packets with the addition of verification that can be made by the tunnel protocol, the network processing of the inner packet headers as discussed above, and verification of the inner packet checksums. Non-IP inner packets are likely to be subject to similar effects that may reduce the likelihood of a misdelivered packet being delivered to a protocol listener that can be affected by the packet.

- Protocols that directly consume the payload must have sufficient robustness against misdelivered packets from any context, including the ones that are corrupted in tunnels and any other usage of the zero checksum. This will require an integrity mechanism. Using a standard UDP checksum reduces the computational load in the receiver to verify this mechanism.

- The design for stateful protocols or protocols where corruption causes cascade effects requires extra care. In tunnel usage, each encapsulating packet provides only a transport mechanism from tunnel ingress to tunnel egress. A corruption will commonly only affect the single tunneled packet, not the established protocol

state. One common effect is that the inner packet flow will only see a corruption and misdelivery of the outer packet as a lost packet.

- Some non-tunnel protocols operate with general servers that do not know the source from which they will receive a packet. In such applications, a zero UDP checksum is unsuitable because there is a need to provide the first level of verification that the packet was intended for the receiving server. A verification prevents the server from processing the datagram payload and without this it may spend significant resources processing the packet, including sending replies or error messages.

Tunnel protocols that encapsulate IP will generally be safe for deployment, since all IPv4 and IPv6 packets include at least one checksum at either the network or transport layer. The network delivery of the inner packet will then further reduce the effects of corruption. Tunnel protocols carrying non-IP packets may offer equivalent protection when the non-IP networks reduce the risk of misdelivery to applications. However, there is a need for further analysis to understand the implications of misdelievery of corrupted packets for that each non-IP protocol. The analysis above suggests that non-tunnel protocols can be expected to have significantly more cases where a zero checksum would result in misdelivery or negative side-effects.

One unfortunate side-effect of increased use of a zero-checksum is that it also increases the likelihood of acceptance when a datagram with a zero UDP checksum is misdelivered. This requires all tunnel protocols using this method to be designed to be robust to misdelivery.

4.3. Middleboxes

"Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero] notes that middleboxes that conform to RFC 2460 will discard datagrams with a zero UDP checksum and should log this as an error. Tunnel protocols intending to use a zero UDP checksum need to ensure that they have defined a method for handling cases when a middlebox prevents the path between the tunnel ingress and egress from supporting transmission of datagrams with a zero UDP checksum.

5. The Zero-Checksum Update

This specification updates IPv6 to allow a zero UDP checksum in the outer encapsulating datagram of a tunnel protocol. UDP endpoints

that implement this update MUST follow the node requirements in "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

The following text in [RFC2460] Section 8.1, 4th bullet should be deleted:

"Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error."

This text should be replaced by:

> An IPv6 node associates a mode with each active UDP port.
>
> Whenever originating a UDP packet for a port in the default mode, an IPv6 node MUST compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it MUST be changed to hex FFFF for placement in the UDP header. IPv6 receivers MUST by default discard UDP packets containing a zero checksum, and SHOULD log the error.
>
> As an alternative, certain protocols that use UDP as a tunnel encapsulation, MAY enable the zero-checksum mode for a specific port (or set of ports). Any node implementing the zero-checksum mode MUST follow the node requirements specified in Section 4 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].
>
> Any protocol that enables the zero-checksum mode for a specific port or ports MUST follow the usage requirements specified in Section 5 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].
>
> Middleboxes supporting IPv6 MUST follow requirements 9, 10 and 11 of the usage requirements specified in Section 5 of "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums" [I-D.ietf-6man-udpzero].

6. Additional Observations

This update was motivated by the existence of a number of protocols being developed in the IETF that are expected to benefit from the

change. The following observations are made:

- An empirically-based analysis of the probabilities of packet corruption (with or without checksums) has not (to our knowledge) been conducted since about 2000. At the time of publication, it is now 2012. We strongly suggest a new empirical study, along with an extensive analysis of the corruption probabilities of the IPv6 header.

- A key motivation for the increase in use of UDP in tunneling is a lack of protocol support in middleboxes. Specifically, new protocols, such as LISP [I-D.ietf-lisp], may prefer to use UDP tunnels to traverse an end-to-end path successfully and avoid having their packets dropped by middleboxes. If middleboxes were updated to support UDP-Lite [RFC3828], UDP-Lite would provide better protection than offered by this update. This may be suited to a variety of applications and would be expected to be preferred over this method for many tunnel protocols.

- Another issue is that the UDP checksum is overloaded with the task of protecting the IPv6 header for UDP flows (as is the TCP checksum for TCP flows). Protocols that do not use a pseudo-header approach to computing a checksum or CRC have essentially no protection from misdelivered packets.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

Less work is required to generate an attack using a zero UDP checksum than one using a standard full UDP checksum. However, this does not lead to significant new vulnerabilities because checksums are not a security measure and can be easily generated by any attacker. Properly configured tunnels should check the validity of the inner packet and perform security checks.

9. Acknowledgements

We would like to thank Brian Haberman, Dan Wing, Joel Halpern and the IESG of 2012 for discussions and reviews. Gorry Fairhurst has been

very diligent in reviewing and help ensuring alignment between this document and [I-D.ietf-6man-udpzero].

10. References

10.1. Normative References

[I-D.ietf-6man-udpzero]
Fairhurst, G. and M. Westerlund, "Applicability Statement for the use of IPv6 UDP Datagrams with Zero Checksums", draft-ietf-6man-udpzero-08 (work in progress), December 2012.

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

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

10.2. Informative References

[I-D.ietf-lisp]
Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-24 (work in progress), November 2012.

[I-D.ietf-mboned-auto-multicast]
Bumgardner, G., "Automatic Multicast Tunneling", draft-ietf-mboned-auto-multicast-14 (work in progress), June 2012.

[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.

[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.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC5619] Yamamoto, S., Williams, C., Yokota, H., and F. Parent, "Softwire Security Analysis and Requirements", RFC 5619, August 2009.

Authors' Addresses

Marshall Eubanks
AmericaFree.TV LLC
P.O. Box 141
Clifton, Virginia 20124
USA

Phone: +1-703-501-4376
Fax:
Email: marshall.eubanks@gmail.com

P.F. Chimento
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723
USA

Phone: +1-443-778-1743
Email: Philip.Chimento@jhuapl.edu

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com