IPv6 and UDP Checksums for Tunneled Packets

Abstract

   This document updates the IPv6 specification (RFC 2460) to improve
   performance when a tunnel protocol uses UDP with IPv6 to tunnel
   packets. The performance improvement is obtained by relaxing the
   IPv6 UDP checksum requirement for tunnel protocols whose header
   information is protected on the "inner" packet being carried.
   Relaxing this requirement removes the overhead associated with the
   computation of UDP checksums on IPv6 packets that carry the tunnel
   protocol packets. This specification describes how the IPv6 UDP
   checksum requirement can be relaxed when the encapsulated packet
   itself contains a checksum. It also describes the limitations and
   risks of this approach and discusses the restrictions on the use of
   this method.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   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/rfc6935.

Copyright Notice

Table of Contents

1. Introduction

   This document constitutes an update of the IPv6 specification
   [RFC2460] for cases 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 new protocol on a critical mass of
   routers, middleboxes, and hosts on the global Internet 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. The result is 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 the
   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 packets are already protected by a checksum.
   IPv4 has a checksum over the IP packet header, and the checksum on
   the outer UDP packet may be set to zero. However, IPv6 has 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 for IPv6 tunnel transports that carry checksum-protected
   packets), following the applicability statements and constraints in
   [RFC6936].

   When reading this document, the advice in "Unicast UDP Usage
   Guidelines for Application Designers" [RFC5405] is applicable. 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"
   [AMT], we expect it to have wide applicability. Since the first
   draft of this RFC was written, the need for an efficient UDP
   tunneling mechanism has increased. Other IETF Working Groups,
   notably LISP [RFC6830] 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. Terminology

This document discusses only IPv6, because the problem being
addressed does not exist for IPv4. Therefore, all references to "IP"
should be understood as references 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, where
reducing the forwarding cost is important, the cost of the
computation may outweigh the benefit of the checksum. 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" [RFC6936] 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 [RFC6936], respectively,
identify node implementation and usage requirements for datagrams
sent and received with a zero UDP checksum. These sections introduce
constraints on the usage of a zero checksum for UDP over IPv6. The
remainder of this section analyzes the use of general tunnels and
explains the motivations for why tunnel protocols are being permitted
to use the method described in this update. It also discusses issues
with middleboxes.

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 specifies what needs to be
   considered by the designer and user of a tunnel protocol for the
   protocol to be robust. It also summarizes why use of a zero UDP
   checksum is thought to be safe for deployment.

   o Context (i.e., tunneling state) should be established by
      exchanging application Protocol Data Units (PDUs) carried in
      checksummed UDP datagrams or by using other protocols that provide
      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.

   o Keepalive 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 keepalives. To enable the tunnel endpoints to discover and
      react to this behavior in a timely way, the keepalive traffic
      should include datagrams with a non-zero checksum and datagrams
      with a zero checksum.

   o 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
         enabled the zero checksum 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
         node has enabled zero checksum for the port, the datagram
         payload is matched to the wrong context and may be processed by
         the wrong tunnel protocol, provided that it also passes the
         verification of that protocol.

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

       When only the source information is corrupted, the datagram could
       arrive at the intended applications or 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 also decreases
       the likelihood of matching a context. However, the exception is
       when errors replace one packet header with another, so both
       packets could be tunneled, and therefore the corrupted packet
       could match a previously defined context.

   o 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, 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
      (although [RFC6936] 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).

   o Compared to other applications, tunnel protocols using UDP have
      some advantages that reduce the risk for a corrupted tunnel packet
      reaching a destination that will receive it. These advantages
      result from processing by the network of the inner (tunneled)
      packet after it is forwarded from the tunnel egress using a wrong
      context:

       * A tunneled packet may be forwarded to the wrong address domain,
          for example, to a private address domain where the inner
          packet's address is not routable, or it 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 be
          reachable at all from the delivered domain. An example is 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.

   Each of these checks significantly reduces 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 risks 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.

   o 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.

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

   o The design for stateful protocols or protocols where corruption
      causes cascade effects requires extra care. In tunnel usage, each
      encapsulating packet provides no functions other than a transport
      from tunnel ingress to tunnel egress. A corruption will commonly
      affect only the single tunneled packet, not the established

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

o 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 it is
   necessary 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; without
   this, the server may spend significant resources processing the
   packet, including sending replies or error messages.

Tunnel protocols that encapsulate IP will generally be safe for
deployment, because 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, further analysis is necessary
to understand the implications of misdelivery of corrupted packets
for 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 in the face
of misdelivery.

4.3. Middleboxes

"Applicability Statement for the Use of IPv6 UDP Datagrams with Zero
Checksums" [RFC6936] specifies requirements for middleboxes and
tunnels that need to traverse middleboxes. 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. This is
especially important as middleboxes that conform to RFC 2460 are
likely to discard datagrams with a zero UDP checksum.

5. The Zero UDP 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" [RFC6936].

   The following text in [RFC2460], Section 8.1, fourth 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 used UDP port (for
      sending and/or receiving packets).

      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,
      the checksum MUST be changed to hex FFFF for placement in the UDP
      header, as specified in [RFC2460]. 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 zero-checksum mode for a specific port
      (or set of ports) for sending and/or receiving. Any node
      implementing 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" [RFC6936].

      Any protocol that enables 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" [RFC6936].

      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"
      [RFC6936].

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:

   o 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 2013. We strongly suggest that a new empirical study be
      performed, along with extensive analysis of the corruption
      probabilities of the IPv6 header. This could potentially allow
      revising the recommendations in this document.

   o 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 [RFC6830], 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. UDP-Lite may be suited to
      a variety of applications and would be expected to be preferred
      over this method for many tunnel protocols.

   o 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. 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.

   In general, any user of zero UDP checksums should apply the checks
   and context verification that are possible to minimize the risk of
   unintended traffic to reach a particular context. This will,
   however, not protect against an intentional attack that creates
   packets with the correct information. Source address validation can
   help prevent injection of traffic into contexts by an attacker.

   Depending on the hardware design, the processing requirements may
   differ for tunnels that have a zero UDP checksum and those that
   calculate a checksum. This processing overhead may need to be
   considered when deciding whether to enable a tunnel and to determine

an acceptable rate for transmission. This processing overhead can become a security risk for designs that can handle a significantly larger number of packets with zero UDP checksums compared to datagrams with a non-zero checksum, such as a tunnel egress. An attacker could attempt to inject non-zero checksummed UDP packets into a tunnel forwarding zero checksum UDP packets and cause overload in the processing of the non-zero checksums, e.g., if this happens in a router's slow path. Therefore, protection mechanisms should be employed when this threat exists. Protection may include source-address filtering to prevent an attacker from injecting traffic, as well as throttling the amount of non-zero checksum traffic. The latter may impact the functioning of the tunnel protocol.

8. Acknowledgments

We would like to thank Brian Haberman, Dan Wing, Joel Halpern, David Waltermire, J.W. Atwood, Peter Yee, Joe Touch, and the IESG of 2012 for discussions and reviews. Gorry Fairhurst has been very diligent in reviewing and helping to ensure alignment between this document and [RFC6936].

9. References

9.1. Normative References

[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.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, April 2013.

9.2. Informative References

[AMT]   Bumgardner, G., "Automatic Multicast Tunneling", 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.

   [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
              Locator/ID Separation Protocol (LISP)", RFC 6830,
              January 2013.

Authors' Addresses

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

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

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

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

   Magnus Westerlund
   Ericsson
   Farogatan 6
   SE-164 80 Kista
   Sweden

   Phone: +46 10 719 00 00
   EMail: magnus.westerlund@ericsson.com