Network Working Group M. Eubanks
Internet-Draft AmericaFree.TV LLC
Intended status: Standards Track P. Chimento
Expires: May 3, 2012 Johns Hopkins University Applied
Physics Laboratory
October 31, 2011

UDP Checksums for Tunneled Packets
draft-ietf-6man-udpchecksums-01

Abstract

This document provides an update of RFC 2460[RFC2460] in order to improve the performance of IPv6 in an increasingly important use case, the use of tunneling to carry new transport protocols. The performance improvement is obtained by relaxing the IPv6 UDP checksum requirement for suitable tunneling 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 tunneled IPv6 packets and thereby improves the efficiency of the traversal of firewalls and other network middleware by such new protocols. We describe how the IPv6 UDP checksum requirement can be relaxed in the situation where the encapsulated packet itself contains a checksum, the limitations and risks of this approach, and provides restrictions on the use of this relaxation to mitigate these risks.

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].

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 May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 4
2. Some Terminology 5
3. Problem Statement 5
4. Discussion 5
5. The Zero-Checksum Solution 7
6. Additional Observations 10
7. IANA Considerations 10
8. Security Considerations 10
9. Acknowledgements 11
10. References 11
  10.1. Normative References 11
  10.2. Informative References 11
Authors' Addresses 12

1. Introduction

This work constitutes the first upgrade of RFC 2460[RFC2460], in order to improve the performance of IPv6 with transport layer protocols carried encapsulated in tunnels. With the rapid growth of the Internet, tunneling protocols have become increasingly important to enable the deployment of new transport layer protocols. Tunneled protocols can be deployed rapidly, while the time to upgrade and deploy a critical mass of routers, switches and end hosts on the global Internet for a new transport protocol is now measured in decades. At the same time, the increasing use of firewalls and other security related middleware 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 tunneling 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 middleware filtering that is a fact of life on the current Internet.

As tunnel endpoints may be routers or middleware aggregating traffic from large numbers of tunnel users, the computation of an additional checksum on the outer UDP packet, when protected, is seen to be an unwarranted burden on the nodes implementing lightweight tunneling protocols, especially if the inner packet(s) are already protected by a checksum. In IPv4, there is a checksum on the IP packet itself, and the checksum on the outer UDP packet can be set to zero. However in IPv6 there is not a checksum on the IP packet and RFC 2460 [RFC2460] explicitly states that IPv6 receivers MUST discard UDP packets with a 0 checksum. So, while sending a UDP packet with a 0 checksum is permitted in IPv4 packets, it is explicitly forbidden in IPv6 packets. In order to meet the needs of the deployers of IPv6 UDP tunnels, this document modifies RFC 2460 to allow for the ignoring of UDP checksums under constrained situations (IPv6 tunneling where the inner packet exists and has a checksum), based on the considerations set forth in [I-D.ietf-6man-udpzero].

While the origin of this I-D is the problem raised by the draft titled "Automatic IP Multicast Without Explicit Tunnels", also known as "AMT," [I-D.ietf-mboned-auto-multicast] we expect it to have wide applicability, immediately to LISP [I-D.ietf-lisp], and also to other tunneling protocols to come out of Softwires and other IETF Working Groups.

Since the first version of this document, the need for an efficient, lightweight UDP tunneling mechanism has increased. Indeed, other workgroups, notably LISP [I-D.ietf-lisp] and Softwires [RFC5619] have also expressed a need to have exceptions to the RFC 2460 prohibition.

Other users of UDP as a tunneling protocol, for example, L2TP and Softwires may benefit from a relaxation of the RFC 2460 restriction.

The third version of this document benefited from a close read by Magnus Westerlund and Gorry Fairhurst.

2. Some Terminology

For the remainder of this document, we discuss only IPv6, since this problem does not exist for IPv4. So any reference to 'IP' should be understood as a reference to IPv6.

Although we will try to avoid them when possible, we may use 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.

3. Problem Statement

The argument is that since in the case of AMT multicast packets already have a UDP header with a checksum, there is no additional benefit and indeed some cost to nodes to both compute and check the UDP checksum of the outer (encapsulating) header. Consequently, IPv6 should make an exception to the rule that the UDP checksum MUST not be 0, and allow tunneling protocols to set the checksum field of the outer header only to 0 and skip both the sender and receiver computation.

4. Discussion

[I-D.ietf-6man-udpzero] describes the issues related to allowing UDP over IPv6 to have a valid checksum of zero and is not repeated here.

In Section 5.1 of [I-D.ietf-6man-udpzero], the authors propose nine (9) constraints on the usage of a zero checksum for UDP over IPv6. We agree with the restrictions proposed, and in fact proposed some of those restrictions ourselves in the previous version of the current draft. These restrictions are incorporated into the proposed changes below.

As has been pointed out in [I-D.ietf-6man-udpzero] and in many mailing lists, there is still the possibility of deep-inspection firewall devices or other middleboxes actually checking the UDP

checksum field of the outer packet and discarding the tunneling packets. This is would be an issue also for legacy systems which have not implemented the change in the IPv6 specification. So in any case, there may be packet loss of lightweight tunneling packets because of mixed new-rule and old-rule nodes.

As an example, we discuss how can errors be detected and handled in a lightweight UDP tunneling protocol when the checksum protection is disabled. Note that other (non-tunneling) protocols may have different approaches. We suggest that the following could be an approach to this problem:

- o Context (i.e. tunneling state) should be established via application PDUs that are carried in checksummed UDP packets. That is, any control packets flowing between the tunnel endpoints should be protected by UDP checksums. The control packets can also contain any negotiation that is necessary to set up the endpoint/adapters to accept UDP packets with a zero checksum.

- o Only UDP packets containing tunneled packets should have a UDP checksum equal to zero.

- o UDP keep-alive packets with checksum zero can be sent to validate paths, given that paths between tunnel endpoints can change and so middleboxes in the path may vary during the life of the association. Paths with middleboxes that are intolerant of a UDP checksum of zero will drop the keep-alives and the endpoints will discover that. Note that this need only be done per tunnel endpoint pair, not per tunnel context. Keep-alive traffic SHOULD include both packets with tunnel checksums and packets with checksums equal to zero to enable the remote end to distinguish between path failures and the blockage of packets with checksum equal to zero.

- o Corruption of the encapsulating IPv6 source address, destination address and/or the UDP source port, destination port fields : If the 9 restrictions in [I-D.ietf-6man-udpzero] are followed, the inner packets (tunneled packets) should be protected and run the usual (presumably small) risk of having undetected corruption(s). If lightweight tunneling protocol contexts contain (at a minimum) source and destination IP addresses and source and destination ports, there are 16 possible corruption outcomes. We note that these outcomes not equally likely, as most require multiple bit errors with errored bits in separate fields. The possible corruption outcomes fall out this way:

  - * Half of the 16 possible corruption combinations have a corrupted destination address. If the incorrect destination is

reached and the node doesn't have an application for the destination port, the packet will be dropped. If the application at the incorrect destination is the same lightweight tunneling protocol and if it has a matching context (which can be assumed to be a very low probability event) the inner packet will be decapsulated and forwarded. If it is some other application, with very high probability, the application will not recognize the contents of the packet.

  * Half of the 8 possible corruption combinations with a correct destination address have a corrupted source address. If the tunnel contexts contain all elements of the address-port 4-tuple, then the likelihood is that this corruption will be detected.

  * Of the remaining 4 possibilities, with valid source and destination IPv6 addresses, 1 has all 4 fields valid, the other three have one or both ports corrupted. Again, if the tunneling endpoint context contains sufficient information, these error should be detected with high probability.

- Corruption of source-fragmented encapsulating packets: In this case, a tunneling 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).

While this is not a perfect solution, it can reduce the risks of relaxing the UDP checksum requirement for IPv6.

5. The Zero-Checksum Solution

The solution to the overhead associated with UDP packets carrying encapsulated tunnel traffic is to allow a UDP checksum of zero on the outer encapsulating packet of a lightweight tunneling protocol. UDP endpoints that implement this solution MUST change their behavior and not discard UDP packets received with a 0 checksum on the outer packet of tunneling protocols. If this is done constraints in Section 5.1 of [I-D.ietf-6man-udpzero] also MUST be adopted.

Specifically, the text in [RFC2460] Section 8.1, 4th bullet is

amended. We refer to the following text:

"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 item should be taken out of the bullet list and should be modified as follows:

> Whenever originating a UDP packet, an IPv6 node SHOULD 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 SHOULD discard UDP packets containing a zero checksum, and SHOULD log the error. However, some protocols, such as lightweight tunneling protocols that use UDP as a tunnel encapsulation, MAY omit computing the UDP checksum of the encapsulating UDP header and set it to zero, subject to the constraints described in [I-D.ietf-6man-udpzero]. In cases where the encapsulating protocol uses a zero checksum for UDP, the receiver of packets sent to a port enabled to receive zero-checksum packets MUST NOT discard packets solely for having a UDP checksum of zero. Note that these constraints apply only to encapsulating protocols that omit calculating the UDP checksum and set it to zero. An encapsulating protocol can always choose to compute the UDP checksum, in which case, its behavior should be as specified originally.

1. IPv6 protocol stack implementations SHOULD NOT by default allow the new method. The default node receiver behavior MUST discard all IPv6 packets carrying UDP packets with a zero checksum.

2. Implementations MUST provide a way to signal the set of ports that will be enabled to receive UDP datagrams with a zero checksum. An IPv6 node that enables reception of UDP packets with a zero-checksum, MUST enable this only for a specific port or port-range. This may be implemented via a socket API call, or similar mechanism.

3. RFC 2460 specifies that IPv6 nodes should log UDP datagrams with a zero-checksum. A port for which zero-checksum has been

enabled MUST NOT log zero-checksum datagrams for that reason (of course, there might be other reasons to log such packets).

4. A stack may separately identify UDP datagrams that are discarded with a zero checksum. It SHOULD NOT add these to the standard log, since the endpoint has not been verified.

5. UDP Tunnels that encapsulate IP may rely on the inner packet integrity checks provided that the tunnel will not significantly increase the rate of corruption of the inner IP packet. If a significantly increased corruption rate can occur, then the tunnel MUST provide an additional integrity verification mechanism. An integrity mechanism is always recommended at the tunnel layer to ensure that corruption rates of the inner most packet are not increased.

6. Tunnels that encapsulate Non-IP packets MUST have a CRC or other mechanism for checking packet integrity, unless the Non-IP packet specifically is designed for transmission over lower layers that do not provide any packet integrity guarantee. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.

7. UDP applications that support use of a zero-checksum, SHOULD NOT rely upon correct reception of the IP and UDP protocol information (including the length of the packet) when decoding and processing the packet payload. In particular, the application must be designed so that corruption of this information does not result in accumulated state or incorrect processing of a tunneled payload.

8. If a method proposes recursive tunnels, it MUST provide guidance that is appropriate for all use-cases. Restrictions may be needed to the use of a tunnel encapsulations and the use of recursive tunnels (e.g. Necessary when the endpoint is not verified).

9. IPv6 nodes that receive ICMPv6 messages that refer to packets with a zero UDP checksum MUST provide appropriate checks concerning the consistency of the reported packet to verify that the reported packet actually originated from the node, before acting upon the information (e.g. validating the address and port numbers in the ICMPv6 message body).

Middleboxes MUST allow IPv6 packets with UDP checksum equal to zero to pass. Implementations of middleboxes MAY allow configuration of specific port ranges for which a zero UDP checksum is valid and may drop IPv6 UDP packets outside those ranges.

6. Additional Observations

The persistence of this issue among a significant number of protocols being developed in the IETF requires a definitive policy. The authors would like to make the following observations:

- An empirically-based analysis of the probabilities of packet corruptions (with or without checksums) has not (to our knowledge) been conducted since about 2000. It is now 2011. We strongly suggest that an empirical study is in order, along with an extensive analysis of IPv6 header corruption probabilities.

- A key cause of this issue generally is the lack of protocol support in middleboxes. Specifically, new protocols, such as LISP, are being forced to use UDP tunnels just to traverse an end-to-end path successfully and avoid having their packets dropped by middleboxes. If this were not the case, the use of UDP-lite might become more viable for some (but not necessarily all) lightweight tunneling protocols.

- Another cause of this issue is that the UDP checksum is overloaded with the task of protecting the IPv6 header for UDP flows (as it 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

It is of course less work to generate zero-checksum attack packets than ones with full UDP checksums. However, this does not lead to any significant new vulnerabilities as checksums are not a security measure and can be easily generated by any attacker, as properly

configured tunnels should check the validity of the inner packet and perform any needed security checks, regardless of the checksum status, and finally as most attacks are generated from compromised hosts which automatically create checksummed packets (in other words, it would generally be more, not less, effort for most attackers to generate zero UDP checksums on the host).

9. Acknowledgements

We would like to thank Brian Haberman, Magnus Westerlund and Gorry Fairhurst for discussions and reviews.

10. References

10.1. Normative References

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

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

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

[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.

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

10.2. Informative References

[I-D.ietf-6man-udpzero]
Fairhurst, G. and M. Westerlund, "IPv6 UDP Checksum Considerations", draft-ietf-6man-udpzero-04 (work in progress), October 2011.

[I-D.ietf-lisp]
Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-15 (work in progress), July 2011.

[I-D.ietf-mboned-auto-multicast]

Thaler, D., Talwar, M., Aggarwal, A., Vicisano, L., and T. Pusateri, "Automatic IP Multicast Without Explicit Tunnels (AMT)", draft-ietf-mboned-auto-multicast-11 (work in progress), July 2011.

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
Fax:
Email: Philip.Chimento@jhuapl.edu
URI: