Internet Area
Internet-Draft
Intended status: Informational
Expires: April 30, 2009

C. Ng
B. Lim
M. Jeyatharan
Panasonic

October 27, 2008

Tunnel Loops and its Detection
draft-ng-intarea-tunnel-loop-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 30, 2009.

Abstract

Many protocols in the Internet Protocol suite use packet encapsulations. This runs into the danger of forming a tunnel loop. Since each tunnel entry point encapsulates the inner packet with a tunnel packet header that contains a new hop count, a packet entering a tunnel loop may be routed infinitely, consuming network resources. Although there exist methods to cause a packet in a tunnel loop to be discarded eventually, it would be more desirable to detect the presence of a tunnel loop and act accordingly. This draft explores the possibility for tunnel entry points to detect the presence of a tunnel loop by using an extra identifier tagged to the outer packet header.

Table of Contents

1. Introduction

1.1. Background

Many protocols in the Internet Protocol suite use packet encapsulation [1][2]. For instance, Virtual Private Network (VPN) relies on tunneling technology to interconnect two or more networks at different locations to form a large enterprise private network. Mobility Support in IPv6 (MIPv6) [3] and Network Mobility [4] uses tunneling between mobile nodes and home agents to allow a mobile node (or network) to be always reachable at its home address (or network prefix). The transition from IPv4 to IPv6 itself relies on tunneling [5] in order to be seamless.

Because tunneling hides the source and destination addresses of the inner packet (which may be encrypted), the existing routing infrastructure can only make routing decision based on the outer packet. This may lead to a phenomenon known as tunnel loop when a tunnel packet is routed back to the tunnel entry node before reaching the tunnel exit node. Tunnel loop is more likely to happen if a packet needs to go through several levels of encapsulation. Figure 1 and Figure 2 show two example scenarios where tunnel loop can occur.

1.2. Example Scenarios

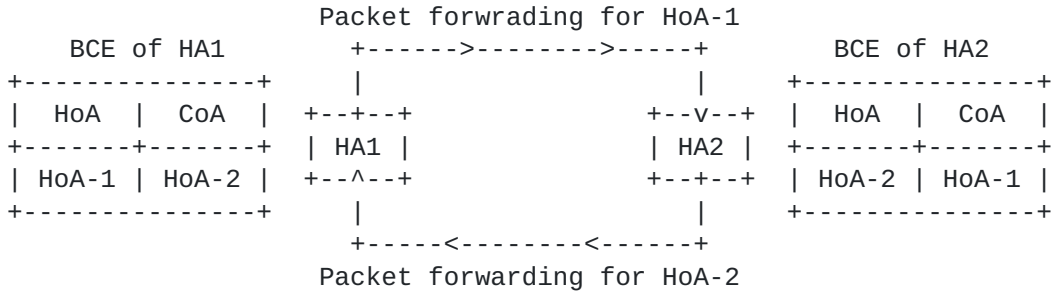

Figure 1: Tunnel Loop Formed with MIPv6

In Figure 1, a mobile node has two home agents, HA1 and HA2, from different service providers. It is assigned home address HoA-1 from HA1 and HoA-2 from HA2. The mobile node then proceeds to bind HoA-2 as a care-of address for HoA-1 at HA1 and bind HoA-1 as a care-of address for HoA-2 at HA2. This will form a tunnel loop as packets forwarded by HA1 will be re-routed back to HA1 by HA2 with an additional encapsulation. This is also noted in the current 3GPP TS 24.303 [6].

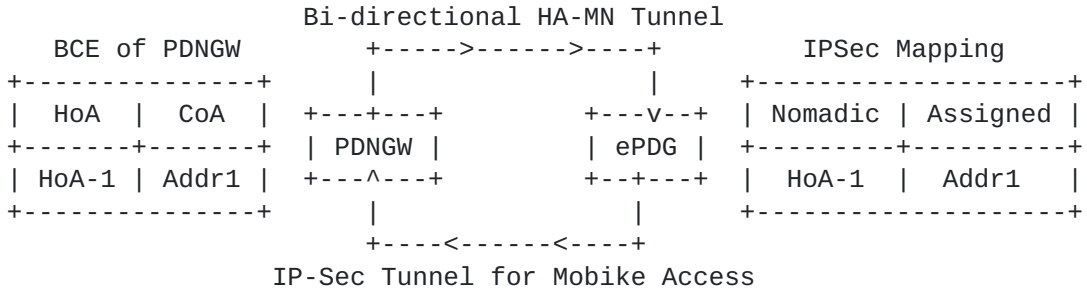

Figure 2: Tunnel Loop Formed with Mobike and MIPv6 in 3GPP

In Figure 2, a 3GPP User Equipment (UE) gains access to the Evolved Packet Core (EPC) by setting up an IPSec tunnel with a evolved Packet Distribution Gateway (ePDG) possibly via Mobike. The UE is assigned HoA-1 from the Packet Distribution Network Gateway (PDNGW) and Addr1 by the ePDG. The UE then proceeds to bind Addr1 as care-of address to HoA-1 at the PDNGW and re-map its Mobike IPSec tunnel with HoA-1 as the nomadic address. This will form a tunnel loop as packets forwarded by PDNGW will be re-routed back to PDNGW by the ePDG with an additional IPSec encapsulation.

1.3. Existing Control Measures

Since each encapsulation conceals the source address of the inner packet, a tunnel entry node may not realize that it has already tunneled a packet previously. Tunnel loop is highly undesirable because it quickly eats up network resources. A packet in a tunnel loop gets routed infinitely because each encapsulation packet will have a new Hop Limit field, thus rendering the existing mechanism of using Hop Limit to safeguard against routing loop ineffective. Furthermore, each encapsulation adds an extra packet header to the packet, causing the packet to grow in size. The packet size may grow so large that fragmentation occurs, effectively introducing another packet into the tunnel loop.

To prevent the catastrophic consequences of a tunnel loop, RFC 2473 [1] specifies the use of a Tunnel Encapsulation Limit (TEL) option for IPv6 which is a destination header option that contains the maximum number of encapsulations a packet can undergo. It is required that all tunnel entry nodes to inspect the destination header of a packet before encapsulation. If a TEL option is found in the packet, the tunnel entry node must first check that the maximum number of encapsulations allowed in the TEL option (given in the TEL value) is not zero before encapsulation. If the TEL value is zero, the tunnel entry node will discard the packet and sent to the originator an Internet Control Message Protocol (ICMP) error

informing the originator of the problem. If the TEL value is non-zero, the tunnel entry node will proceed with encapsulating the packet, and add a TEL option to the new tunnel packet header containing the value equals to the original TEL value minus one. On the other hand, if the original packet does not contain a TEL option, the tunnel entry node proceeds with encapsulation and adds to the tunnel packet header a TEL option containing a default (operator configured) number of maximum encapsulations.

A similar mechanism (RFC 1701 [2]) is used in IPv4 whereby a 3-bit recursion field is used to limit the number of further encapsulations permitted on a packet.

1.4. Inadequacies

Although the use of TEL option (and Recur field) prevents a packet in a tunnel loop from being routed indefinitely, it is an inadequate solution to a complex problem. In particular, the use of TEL option does not allow the recipient of an ICMP error to determine the cause of TEL value reaching zero is due to a tunnel loop, or merely due to the initial TEL value is insufficient for the number of tunnels a packet need to traverse to reach its final destination. Thus, it is unclear how a tunnel entry node should respond to an ICMP error that says the tunnel encapsulation limit has been reached. The tunnel entry node could increase its default TEL value in an attempt to allow the packet to get through. This may lead to an infinite sequence of receiving ICMP error and increasing the default TEL value, if there is indeed a tunnel loop. Alternatively, the tunnel entry node can simply refuse to tunnel packets with the same destination address, assuming that there is a tunnel loop. However, this may cause unnecessary denial-of service when the actual reason for ICMP error was because the number of tunnels a packet needs to traverse is greater than the TEL value set in order for the packet to reach its final destination.

From the above discussion, one can see that the problem with using the TEL option (and Recur field) is that it does not contain sufficient information for a tunnel entry node to differentiate between the case where a tunnel loop has formed, and the case where the number of tunnels a packet needs to traverse is greater than the default TEL value set. With that in mind, this draft explores the possibilities of adding an identifier field to the outer packet to enable tunnel entry points to reliably detect the presence of a tunnel loop.

2. Basic Approach

In order for a tunnel loop to be detected by tunnel entry points, we proposed that a tunnel identifier to be added to the outer packet of a tunnel. The basic idea is for the tunnel entry point to monitor this tunnel identifier on received packets, and flag a "Loop Detected" error when packets with certain tunnel identifier values are received.

The basic approach will be for a tunnel entry point to first inspect the header of a packet received that is to be tunneled, and check if an tunnel identifier is present. If a tunnel identifier is present, the tunnel entry point checks if the tunnel identifier indicates the presence of a tunnel loop. If so, error is flagged. Else, the received packet is processed, and the tunnel identifier is added to the outer header before sending it out. On the other hand, if a tunnel identifier cannot be found in the received packet, the tunnel entry point generates a tunnel identifier to be added to the outer header.

With this basic approach, we will next see how we can implement it for both IPv6 and IPv4.

2.1. Approach for IPv6

For IPv6 tunneling, the Generic IPv6 Tunneling [1] should be used. Hence, the obvious approach is for the tunnel entry point to insert the generated identifier in the TEL Option. This will also allow the tunnel packet to have the benefit of the TEL Option of limiting the number of encapsulations that can be performed on the packet. There are two sub-approaches to do so: firstly, we can extend the TEL Option to include an identifier field; or secondly, we can use multiple TEL Options to encode the identifier field. A third approach is to use an entirely new Destination Header option to carry the identifier.

These are discussed briefly in the following sub-sections.

2.1.1. Extending the Tunnel Encapsulation Limit Option

Since each tunnel entry point is expected to inspect the Tunnel Encapsulation Limit Option, adding the identifier into the TEL Option is a natural extension. This can be done by appending a Tunnel Encapsulation Identifier field to the TEL Option as illustrated in Figure 3 below.

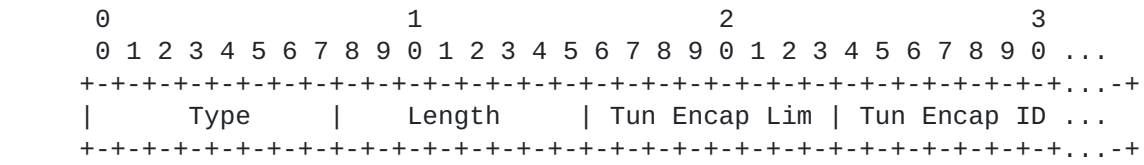

Figure 3: Extending the TEL Option

Type

The Option Type containing the octet 00000100 in binary (decimal value 4), where the highest-order two bits are set to zero to indicate the option is skipped if not recognized, the third highest-order bit is set to zero to indicate this option does not change en-route, and the remaining 5 bits identify the option as the TEL option.

Length

The Option Length specifies the length of the option in octets excluding the Option Type and Length fields. This value is variable depending on the length of the Tunnel Encapsulation Identifier field.

Tunnel Encapsulation Limit (Tun Encap Lim)

The Tunnel Encapsulation Limit is a 8-bit unsigned integer specifying how many further levels of encapsulation are permitted for the packet.

Tunnel Encapsulation Identifier (Tun Encap ID)

The Tunnel Encapsulation Identifier (TEI) is the identifier inserted by the first tunnel entry point, and copied by subsequent tunnel entry points. The length of this field is TBD.

2.1.2. Using Multiple Tunnel Encapsulation Limit Options

The problem with extending TEL option as described in Section 2.1.1 is that legacy tunnel entry point would not be able to process the new TEI field, and would thus omit copying the field when adding an extra level of encapsulation, thereby rendering the approach ineffective. To resolve this, it is possible to use multiple TEL options to encode the identifier. The trick here is to use the first TEL option as the base, and add the identifier value to the TEL field of subsequent TEL options.

As an example, consider a tunnel entry point adding a 16-bits identifier 0x1234 to a tunnel packet. It can add three TEL options to the packet illustrated in Figure 4 below.

```
Outer IPv6 Header
Destination Header {
    TEL Option {
        Option Type = 0x04
        Option Length = 1
        Tun Encap Lim = 5
    }
    TEL Option {
        Option Type = 0x04
        Option Length = 1
        Tun Encap Lim = 0x12+5 = 0x17
    }
    TEL Option {
        Option Type = 0x04
        Option Length = 1
        Tun Encap Lim = 0x34+5 = 0x39
    }
}
Inner Packet
```

Figure 4: Using Multiple TEL Options

In this way, when subsequent tunnel entry points decrement the TEL value for each options, the identifier can still be derived by taking the difference between the TEL values in subsequent TEL options with the TEL value of the first TEL option.

2.1.3. New Tunnel Identifier Option

A third approach is to have a new Destination Header Option that carries the identifier. This is similar to the approach used in Section 2.1.1 except that the Tunnel Encapsulation Identifier field is carried in a new option called the Tunnel Encapsulation Identifier Option. This option should be simply copied to the outer packet by subsequent tunnel entry points.

2.2. Approach for IPv4

The situation for IPv4 tunneling is a bit more complicated, as there is no independent Destination Header that one can rely on. Hence, each tunneling protocol will have to be extended separately in order to detect the presence of a tunneling loop. This will be an almost impossible task, in view of the number of different protocols and

legacy nodes already deployed. Nonetheless, since most tunneling mechanisms in IPv4 utilizes the Generic Routing Encapsulation (GRE) defined in RFC 1701 [2], we could extend the protocol and enable majority of the IPv4 tunnel entry points to detect tunneling loops.

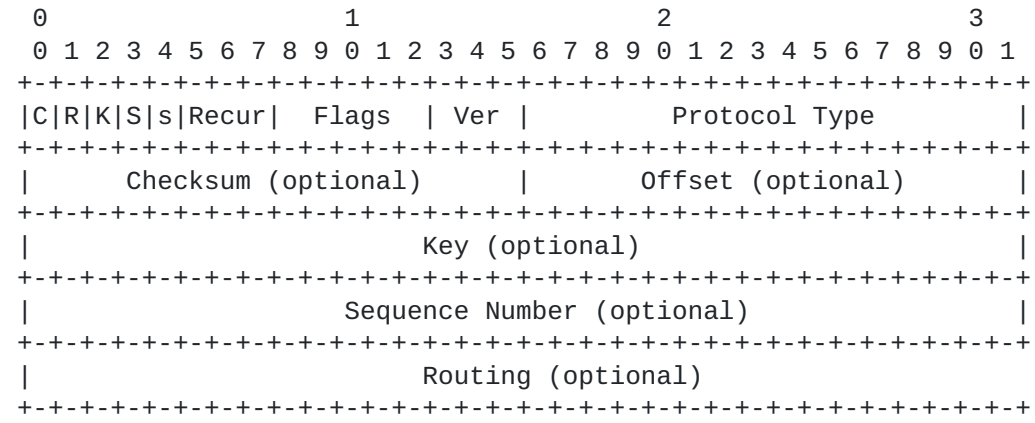

Figure 5: GRE Header

Figure 5 shows the basic header for GRE as defined RFC 1701. Note that RFC 2784 [7] simplified the informational GRE header described in RFC 1701 and reduces the header to only contains the Checksum field (omitting the Key, Sequence Number, and Routing fields). Later, RFC 2890 [8] updates the proposed standard and re-introduces the Key and Sequence Number fields. It is possible to re-use the key and sequence number field to act as an identifier field for the purpose of loop detection, and maintain compatibility with legacy nodes that already implements RFC 2890 or RFC 1701.

2.3. Types of Identifier

Different types of identifier can be used to detect tunnel loop, such as a unique identifier associated with each tunnel entry point, a hash value generated based on the contents of the innermost packet, or a randomly generated number. Factors to consider when selecting the type of identifiers include the possibility of erroneously detecting a loop, the possibility of failing to detect a loop and the overhead added. Analysis of the type of identifiers is left for future version.

3. Conclusion

This memo highlights the problem of tunnel loops, and demonstrated the need for not only limiting the number of encapsulations on a packet (as is available in current standards) but also the need to detect tunnel loops so that appropriate actions may be taken. To do so, we proposed that an identifier be inserted into the outer packet header which should be copied by subsequent tunnel entry points. This will enable a tunnel entry point to detect that a packet it is about to encapsulate has already been encapsulated by itself before, and thus signal the presence of a tunnel loop.

4. Informative Reference

[1] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[2] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, October 1994.

[3] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[4] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, January 2005.

[5] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[6] "Mobility Management based on Dual-Stack Mobile IPv6", 3GPP TS 24.303, ver 1.3.0, October 2008.

[7] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[8] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, September 2000.

Appendix A. Change Log

- draft-ng-intarea-tunnel-loop-00:
  - Initial version.

Authors' Addresses

Chan-Wah Ng
Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Seng Ave #06-3530
Tai Seng Industrial Estate
Singapore 534415
SG

Phone: +65 65505420
Email: chanwah.ng@sg.panasonic.com

Benjamin Lim
Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Seng Ave #06-3530
Tai Seng Industrial Estate
Singapore 534415
SG

Phone: +65 65505478
Email: benjamin.limck@sg.panasonic.com

Mohana Jeyatharan
Panasonic Singapore Laboratories Pte Ltd
Blk 1022 Tai Seng Ave #06-3530
Tai Seng Industrial Estate
Singapore 534415
SG

Phone: +65 65505494
Email: mohana.jeyatharan@sg.panasonic.com

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.