Network Working Group   V. Gill
Request for Comments: 3682   J. Heasley
Category: Experimental   D. Meyer
                                                          February 2004

                The Generalized TTL Security Mechanism (GTSM)

Status of this Memo

Copyright Notice

Abstract

   The use of a packet's Time to Live (TTL) (IPv4) or Hop Limit (IPv6)
   to protect a protocol stack from CPU-utilization based attacks has
   been proposed in many settings (see for example, RFC 2461). This
   document generalizes these techniques for use by other protocols such
   as BGP (RFC 1771), Multicast Source Discovery Protocol (MSDP),
   Bidirectional Forwarding Detection, and Label Distribution Protocol
   (LDP) (RFC 3036). While the Generalized TTL Security Mechanism
   (GTSM) is most effective in protecting directly connected protocol
   peers, it can also provide a lower level of protection to multi-hop
   sessions. GTSM is not directly applicable to protocols employing
   flooding mechanisms (e.g., multicast), and use of multi-hop GTSM
   should be considered on a case-by-case basis.

Table of Contents

1. Introduction

   The Generalized TTL Security Mechanism (GTSM) is designed to protect
   a router's TCP/IP based control plane from CPU-utilization based
   attacks. In particular, while cryptographic techniques can protect
   the router-based infrastructure (e.g., BGP [RFC1771], [RFC1772]) from
   a wide variety of attacks, many attacks based on CPU overload can be
   prevented by the simple mechanism described in this document. Note
   that the same technique protects against other scarce-resource
   attacks involving a router's CPU, such as attacks against
   processor-line card bandwidth.

   GTSM is based on the fact that the vast majority of protocol peerings
   are established between routers that are adjacent [PEERING]. Thus
   most protocol peerings are either directly between connected
   interfaces or at the worst case, are between loopback and loopback,
   with static routes to loopbacks. Since TTL spoofing is considered
   nearly impossible, a mechanism based on an expected TTL value can
   provide a simple and reasonably robust defense from infrastructure
   attacks based on forged protocol packets.

   Finally, the GTSM mechanism is equally applicable to both TTL (IPv4)
   and Hop Limit (IPv6), and from the perspective of GTSM, TTL and Hop
   Limit have identical semantics. As a result, in the remainder of
   this document the term "TTL" is used to refer to both TTL or Hop
   Limit (as appropriate).

   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 BCP 14, RFC 2119
   [RFC2119].

2. Assumptions Underlying GTSM

   GTSM is predicated upon the following assumptions:

   (i)   The vast majority of protocol peerings are between adjacent
          routers [PEERING].

(ii)   It is common practice for many service providers to ingress
        filter (deny) packets that have the provider's loopback
        addresses as the source IP address.

(iii)   Use of GTSM is OPTIONAL, and can be configured on a per-peer
        (group) basis.

(iv)   The router supports a method of classifying traffic destined
        for the route processor into interesting/control and
        not-control queues.

(iv)   The peer routers both implement GTSM.

2.1. GTSM Negotiation

   This document assumes that GTSM will be manually configured between
   protocol peers. That is, no automatic GTSM capability negotiation,
   such as is envisioned by RFC 2842 [RFC2842] is assumed or defined.

2.2. Assumptions on Attack Sophistication

   Throughout this document, we assume that potential attackers have
   evolved in both sophistication and access to the point that they can
   send control traffic to a protocol session, and that this traffic
   appears to be valid control traffic (i.e., has the source/destination
   of configured peer routers).

   We also assume that each router in the path between the attacker and
   the victim protocol speaker decrements TTL properly (clearly, if
   either the path or the adjacent peer is compromised, then there are
   worse problems to worry about).

   Since the vast majority of our peerings are between adjacent routers,
   we can set the TTL on the protocol packets to 255 (the maximum
   possible for IP) and then reject any protocol packets that come in
   from configured peers which do NOT have an inbound TTL of 255.

   GTSM can be disabled for applications such as route-servers and other
   large diameter multi-hop peerings. In the event that an the attack
   comes in from a compromised multi-hop peering, that peering can be
   shut down (a method to reduce exposure to multi-hop attacks is
   outlined below).

3. GTSM Procedure

   GTSM SHOULD NOT be enabled by default. The following process
   describes the per-peer behavior:

(i)   If GTSM is enabled, an implementation performs the following
       procedure:

       (a) For directly connected routers,

            o Set the outbound TTL for the protocol connection to 255.

            o For each configured protocol peer:

              Update the receive path Access Control List (ACL) or
              firewall to only allow protocol packets to pass onto the
              Route Processor (RP) that have the correct <source,
              destination, TTL> tuple. The TTL must either be 255
              (for a directly connected peer), or 255-(configured-
              range-of-acceptable-hops) for a multi-hop peer. We
              specify a range here to achieve some robustness to
              changes in topology. Any directly connected check MUST
              be disabled for such peerings.

              It is assumed that a receive path ACL is an ACL that is
              designed to control which packets are allowed to go to
              the RP. This procedure will only allow protocol packets
              from adjacent router to pass onto the RP.

       (b) If the inbound TTL is 255 (for a directly connected
            peer), or 255-(configured-range-of-acceptable-hops) (for
            multi-hop peers), the packet is NOT processed. Rather,
            the packet is placed into a low priority queue, and
            subsequently logged and/or silently discarded. In this
            case, an ICMP message MUST NOT be generated.

   (ii) If GTSM is not enabled, normal protocol behavior is followed.

3.1. Multi-hop Scenarios

   When a multi-hop protocol session is required, we set the expected
   TTL value to be 255-(configured-range-of-acceptable-hops). This
   approach provides a qualitatively lower degree of security for the
   protocol implementing GTSM (i.e., a DoS attack could theoretically be
   launched by compromising some box in the path). However, GTSM will
   still catch the vast majority of observed DDoS attacks against a
   given protocol. Note that since the number of hops can change
   rapidly in real network situations, it is considered that GTSM may
   not be able to handle this scenario adequately and an implementation
   MAY provide OPTIONAL support.

3.1.1. Intra-domain Protocol Handling

   In general, GTSM is not used for intra-domain protocol peers or
   adjacencies. The special case of iBGP peers can be protected by
   filtering at the network edge for any packet that has a source
   address of one of the loopback addresses used for the intra-domain
   peering. In addition, the current best practice is to further
   protect such peers or adjacencies with an MD5 signature [RFC2385].

4. Acknowledgments

   The use of the TTL field to protect BGP originated with many
   different people, including Paul Traina and Jon Stewart. Ryan
   McDowell also suggested a similar idea. Steve Bellovin, Jay
   Borkenhagen, Randy Bush, Vern Paxon, Pekka Savola, and Robert Raszuk
   also provided useful feedback on earlier versions of this document.
   David Ward provided insight on the generalization of the original
   BGP-specific idea.

5. Security Considerations

   GTSM is a simple procedure that protects single hop protocol
   sessions, except in those cases in which the peer has been
   compromised.

5.1. TTL (Hop Limit) Spoofing

   The approach described here is based on the observation that a TTL
   (or Hop Limit) value of 255 is non-trivial to spoof, since as the
   packet passes through routers towards the destination, the TTL is
   decremented by one. As a result, when a router receives a packet, it
   may not be able to determine if the packet's IP address is valid, but
   it can determine how many router hops away it is (again, assuming
   none of the routers in the path are compromised in such a way that
   they would reset the packet's TTL).

   Note, however, that while engineering a packet's TTL such that it has
   a particular value when sourced from an arbitrary location is
   difficult (but not impossible), engineering a TTL value of 255 from
   non-directly connected locations is not possible (again, assuming
   none of the directly connected neighbors are compromised, the packet
   hasn't been tunneled to the decapsulator, and the intervening routers
   are operating in accordance with RFC 791 [RFC791]).
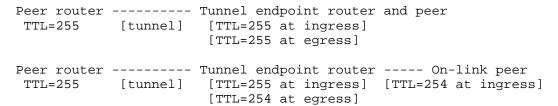
5.2. Tunneled Packets

   An exception to the observation that a packet with TTL of 255 is
   difficult to spoof occurs when a protocol packet is tunneled to a
   decapsulator who then forwards the packet to a directly connected
   protocol peer. In this case the decapsulator (tunnel endpoint) can
   either be the penultimate hop, or the last hop itself. A related
   case arises when the protocol packet is tunneled directly to the
   protocol peer (the protocol peer is the decapsulator).

   When the protocol packet is encapsulated in IP, it is possible to
   spoof the TTL. It may also be impossible to legitimately get the
   packet to the protocol peer with a TTL of 255, as in the IP in MPLS
   cases described below.

   Finally, note that the security of any tunneling technique depends
   heavily on authentication at the tunnel endpoints, as well as how the
   tunneled packets are protected in flight. Such mechanisms are,
   however, beyond the scope of this memo.

5.2.1. IP in IP

   Protocol packets may be tunneled over IP directly to a protocol peer,
   or to a decapsulator (tunnel endpoint) that then forwards the packet
   to a directly connected protocol peer (e.g., in IP-in-IP [RFC2003],
   GRE [RFC2784], or various forms of IPv6-in-IPv4 [RFC2893]). These
   cases are depicted below.

```
   Peer router ---------- Tunnel endpoint router and peer
     TTL=255     [tunnel]   [TTL=255 at ingress]
                            [TTL=255 at egress]


   Peer router ---------- Tunnel endpoint router ----- On-link peer
     TTL=255     [tunnel]   [TTL=255 at ingress]  [TTL=254 at ingress]
                            [TTL=254 at egress]
```

   In the first case, in which the encapsulated packet is tunneled
   directly to the protocol peer, the encapsulated packet's TTL can be
   set arbitrary value. In the second case, in which the encapsulated
   packet is tunneled to a decapsulator (tunnel endpoint) which then
   forwards it to a directly connected protocol peer, RFC 2003 specifies
   the following behavior:

      When encapsulating a datagram, the TTL in the inner IP header is
      decremented by one if the tunneling is being done as part of
      forwarding the datagram; otherwise, the inner header TTL is not
      changed during encapsulation. If the resulting TTL in the inner
      IP header is 0, the datagram is discarded and an ICMP Time

      Exceeded message SHOULD be returned to the sender. An
      encapsulator MUST NOT encapsulate a datagram with TTL = 0.

   Hence the inner IP packet header's TTL, as seen by the decapsulator,
   can be set to an arbitrary value (in particular, 255). As a result,
   it may not be possible to deliver the protocol packet to the peer
   with a TTL of 255.

5.2.2. IP in MPLS

   Protocol packets may also be tunneled over MPLS to a protocol peer
   which either the penultimate hop (when the penultimate hop popping
   (PHP) is employed [RFC3032]), or one hop beyond the penultimate hop.
   These cases are depicted below.

    Peer router ---------- Penultimate Hop (PH) and peer
      TTL=255   [tunnel]   [TTL=255 at ingress]
                            [TTL<=254 at egress]

    Peer router ---------- Penultimate Hop -------- On-link peer
      TTL=255   [tunnel]   [TTL=255 at ingress] [TTL <=254 at ingress]
                            [TTL<=254 at egress]

   TTL handling for these cases is described in RFC 3032. RFC 3032
   states that when the IP packet is first labeled:

      ... the TTL field of the label stack entry MUST BE set to the
      value of the IP TTL field. (If the IP TTL field needs to be
      decremented, as part of the IP processing, it is assumed that
      this has already been done.)

   When the label is popped:

      When a label is popped, and the resulting label stack is empty,
      then the value of the IP TTL field SHOULD BE replaced with the
      outgoing TTL value, as defined above. In IPv4 this also requires
      modification of the IP header checksum.

   where the definition of "outgoing TTL" is:

      The "incoming TTL" of a labeled packet is defined to be the value
      of the TTL field of the top label stack entry when the packet is
      received.

The "outgoing TTL" of a labeled packet is defined to be the larger
of:

    a) one less than the incoming TTL,
    b) zero.

In either of these cases, the minimum value by which the TTL could be
decremented would be one (the network operator prefers to hide its
infrastructure by decrementing the TTL by the minimum number of LSP
hops, one, rather than decrementing the TTL as it traverses its MPLS
domain). As a result, the maximum TTL value at egress from the MPLS
cloud is 254 (255-1), and as a result the check described in section
3 will fail.

5.3. Multi-Hop Protocol Sessions

While the GTSM method is less effective for multi-hop protocol
sessions, it does close the window on several forms of attack.
However, in the multi-hop scenario GTSM is an OPTIONAL extension.
Protection of the protocol infrastructure beyond what is provided by
the GTSM method will likely require cryptographic machinery such as
is envisioned by Secure BGP (S-BGP) [SBGP1,SBGP2], and/or other
extensions. Finally, note that in the multi-hop case described
above, we specify a range of acceptable TTLs in order to achieve some
robustness to topology changes. This robustness to topological
change comes at the cost of the loss of some robustness to different
forms of attack.

6. IANA Considerations

This document creates no new requirements on IANA namespaces
[RFC2434].

7. References

7.1. Normative References

    [RFC791]   Postel, J., "Internet Protocol Specification", STD 5, RFC
              791, September 1981.

    [RFC1771]   Rekhter, Y. and T. Li (Editors), "A Border Gateway
              Protocol (BGP-4)", RFC 1771, March 1995.

    [RFC1772]   Rekhter, Y. and P. Gross, "Application of the Border
              Gateway Protocol in the Internet", RFC 1772, March 1995.

    [RFC2003]   Perkins, C., "IP Encapsulation with IP", RFC 2003, October
              1996.

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

   [RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5
              Signature Option", RFC 2385, August 1998.

   [RFC2461] Narten, T., Nordmark, E. and W. Simpson, "Neighbor
              Discover for IP Version 6 (IPv6)", RFC 2461, December
              1998.

   [RFC2784] Farinacci, D., "Generic Routing Encapsulation (GRE)", RFC
              2784, March 2000.

   [RFC2842] Chandra, R. and J. Scudder, "Capabilities Advertisement
              with BGP-4", RFC 2842, May 2000.

   [RFC2893] Gilligan, R. and E. Nordmark, "Transition Mechanisms for
              IPv6 Hosts and Routers", RFC 2893, August 2000.

   [RFC3032] Rosen, E. Tappan, D., Fedorkow, G., Rekhter, Y.,
              Farinacci, D., Li, T. and A. Conta, "MPLS Label Stack
              Encoding", RFC 3032, January 2001.

   [RFC3036] Andersson, L., Doolan, P., Feldman, N., Fredette, A. and
              B. Thomas, "LDP Specification", RFC 3036, January 2001.

   [RFC3392] Chandra, R. and J. Scudder, "Capabilities Advertisement
              with BGP-4", RFC 3392, November 2002.

   [SBGP1]   Kent, S., C. Lynn, and K. Seo, "Secure Border Gateway
              Protocol (Secure-BGP)", IEEE Journal on Selected Areas in
              Communications, volume 18, number 4, April, 2000.

   [SBGP2]   Kent, S., C. Lynn, J. Mikkelson, and K. Seo, "Secure
              Border Gateway Protocol (S-BGP) -- Real World Performance
              and Deployment Issues", Proceedings of the IEEE Network
              and Distributed System Security Symposium, February, 2000.

7.2. Informative References

   [BFD]   Katz, D. and D. Ward, "Bidirectional Forwarding
              Detection", Work in Progress, June 2003.

   [PEERING] Empirical data gathered from the Sprint and AOL backbones,
              October, 2002.

   [RFC2028]   Hovey, R. and S. Bradner, "The Organizations Involved in
               the IETF Standards Process", BCP 11, RFC 2028, October
               1996.

   [RFC2434]   Narten, T., and H. Alvestrand, "Guidelines for Writing an
               IANA Considerations Section in RFCs", BCP 26, RFC 2434,
               October 1998.

   [RFC3618]   Meyer, D. and W. Fenner, Eds., "The Multicast Source
               Discovery Protocol (MSDP)", RFC 3618, October 2003.

8. Authors' Addresses

   Vijay Gill

   EMail: vijay@umbc.edu

   John Heasley

   EMail: heas@shrubbery.net

   David Meyer

   EMail: dmm@1-4-5.net

9. Full Copyright Statement

Acknowledgement