                 The Generalized TTL Security Mechanism (GTSM)

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   The use of a packet's Time to Live (TTL) (IPv4) or Hop Limit (IPv6)
   to verify whether the packet was originated by an adjacent node on a
   connected link has been used in many recent protocols. This document
   generalizes this technique. This document obsoletes Experimental RFC
   3682.

Table of Contents

1. Introduction

   The Generalized TTL Security Mechanism (GTSM) is designed to protect
   a router's IP-based control plane from CPU-utilization based attacks.
   In particular, while cryptographic techniques can protect the router-
   based infrastructure (e.g., BGP [RFC4271], [RFC4272]) 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. Thus, most
   protocol peerings are either directly between connected interfaces
   or, in 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 from outside the network. Note,
   however, that GTSM is not a substitute for authentication mechanisms.
   In particular, it does not secure against insider on-the-wire
   attacks, such as packet spoofing or replay.

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 RFC 2119 [RFC2119].

2. Assumptions Underlying GTSM

   GTSM is predicated upon the following assumptions:

   1. The vast majority of protocol peerings are between adjacent
       routers.

   2. Service providers may or may not configure strict ingress
       filtering [RFC3704] on non-trusted links. If maximal protection
       is desired, such filtering is necessary as described in
       Section 2.2.

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

   4. The peer routers both implement GTSM.

   5. The router supports a method to use separate resource pools
       (e.g., queues, processing quotas) for differently classified
       traffic.

   Note that this document does not prescribe further restrictions that
   a router may apply to packets not matching the GTSM filtering rules,
   such as dropping packets that do not match any configured protocol
   session and rate-limiting the rest. This document also does not
   suggest the actual means of resource separation, as those are
   hardware and implementation-specific.

   However, the possibility of denial-of-service (DoS) attack prevention
   is based on the assumption that classification of packets and
   separation of their paths are done before the packets go through a
   scarce resource in the system. In practice, the closer GTSM
   processing is done to the line-rate hardware, the more resistant the
   system is to DoS attacks.

2.1. GTSM Negotiation

   This document assumes that, when used with existing protocols, GTSM
   will be manually configured between protocol peers. That is, no
   automatic GTSM capability negotiation, such as is provided by RFC
   3392 [RFC3392], is assumed or defined.

   If a new protocol is designed with built-in GTSM support, then it is
   recommended that procedures are always used for sending and
   validating received protocol packets (GTSM is always on, see for
   example [RFC2461]). If, however, dynamic negotiation of GTSM support
   is necessary, protocol messages used for such negotiation MUST be
   authenticated using other security mechanisms to prevent DoS attacks.

   Also note that this specification does not offer a generic GTSM
   capability negotiation mechanism, so messages of the protocol
   augmented with the GTSM behavior will need to be used if dynamic
   negotiation is deemed necessary.

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., it 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).

   For maximal protection, ingress filtering should be applied before
   the packet goes through the scarce resource. Otherwise an attacker
   directly connected to one interface could disturb a GTSM-protected
   session on the same or another interface. Interfaces that aren't
   configured with this filtering (e.g., backbone links) are assumed to
   not have such attackers (i.e., are trusted).

   As a specific instance of such interfaces, we assume that tunnels are
   not a back-door for allowing TTL-spoofing on protocol packets to a
   GTSM-protected peering session with a directly connected neighbor.
   We assume that: 1) there are no tunneled packets terminating on the
   router, 2) tunnels terminating on the router are assumed to be secure
   and endpoints are trusted, 3) tunnel decapsulation includes source
   address spoofing prevention [RFC3704], or 4) the GTSM-enabled session
   does not allow protocol packets coming from a tunnel.

Since the vast majority of 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 that do NOT have an inbound TTL of 255.

GTSM can be disabled for applications such as route-servers and other
multi-hop peerings. In the event that an attack comes in from a
compromised multi-hop peering, that peering can be shut down.

3. GTSM Procedure

If GTSM is not built into the protocol and is used as an additional
feature (e.g., for BGP, LDP, or MSDP), it SHOULD NOT be enabled by
default in order to remain backward-compatible with the unmodified
protocol. However, if the protocol defines a built-in dynamic
capability negotiation for GTSM, a protocol peer MAY suggest the use
of GTSM provided that GTSM would only be enabled if both peers agree
to use it.

If GTSM is enabled for a protocol session, the following steps are
added to the IP packet sending and reception procedures:

   Sending protocol packets:

      The TTL field in all IP packets used for transmission of
      messages associated with GTSM-enabled protocol sessions MUST be
      set to 255. This also applies to the related ICMP error
      handling messages.

      On some architectures, the TTL of control plane originated
      traffic is under some configurations decremented in the
      forwarding plane. The TTL of GTSM-enabled sessions MUST NOT be
      decremented.

   Receiving protocol packets:

      The GTSM packet identification step associates each received
      packet addressed to the router's control plane with one of the
      following three trustworthiness categories:

      + Unknown: these are packets that cannot be associated with
         any registered GTSM-enabled session, and hence GTSM cannot
         make any judgment on the level of risk associated with them.

      + Trusted: these are packets that have been identified as
         belonging to one of the GTSM-enabled sessions, and their TTL
         values are within the expected range.

+   Dangerous: these are packets that have been identified as
    belonging to one of the GTSM-enabled sessions, but their TTL
    values are NOT within the expected range, and hence GTSM
    believes there is a risk that these packets have been
    spoofed.

The exact policies applied to packets of different
classifications are not postulated in this document and are
expected to be configurable. Configurability is likely
necessary in particular with the treatment of related messages
(ICMP errors). It should be noted that fragmentation may
restrict the amount of information available for
classification.

However, by default, the implementations:

+   SHOULD ensure that packets classified as Dangerous do not
    compete for resources with packets classified as Trusted or
    Unknown.

+   MUST NOT drop (as part of GTSM processing) packets
    classified as Trusted or Unknown.

+   MAY drop packets classified as Dangerous.

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, Alfred Hoenes, Vern Paxon, Robert Raszuk,
and Alex Zinin also provided useful feedback on earlier versions of
this document. David Ward provided insight on the generalization of
the original BGP-specific idea. Alex Zinin, Alia Atlas, and John
Scudder provided a significant amount of feedback for the newer
versions of the document. During and after the IETF Last Call,
useful comments were provided by Francis Dupont, Sam Hartman, Lars
Eggert, and Ross Callon.

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. In particular, it does not protect against the wide
range of on-the-wire attacks; protection from these attacks requires
more rigorous security mechanisms.

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 per router. 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
   has not been tunneled to the decapsulator, and the intervening
   routers are operating in accordance with RFC 791 [RFC0791]).

5.2. Tunneled Packets

   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.

   An exception to the observation that a packet with TTL of 255 is
   difficult to spoof may occur when a protocol packet is tunneled and
   the tunnel is not integrity-protected (i.e., the lower layer is
   compromised).

   When the protocol packet is tunneled directly to the protocol peer
   (i.e., the protocol peer is the decapsulator), the GTSM provides some
   limited added protection as the security depends entirely on the
   integrity of the tunnel.

   For protocol adjacencies over a tunnel, if the tunnel itself is
   deemed secure (i.e., the underlying infrastructure is deemed secure,
   and the tunnel offers degrees of protection against spoofing such as
   keys or cryptographic security), the GTSM can serve as a check that
   the protocol packet did not originate beyond the head-end of the
   tunnel. In addition, if the protocol peer can receive packets for
   the GTSM-protected protocol session from outside the tunnel, the GTSM
   can help thwart attacks from beyond the adjacent router.

   When the tunnel tail-end decapsulates the protocol packet and then
   IP-forwards the packet to a directly connected protocol peer, the TTL
   is decremented as described below. This means that the tunnel

decapsulator is the penultimate node from the GTSM-protected protocol
peer's perspective. As a result, the GTSM check protects from
attackers encapsulating packets to your peers. However, specific
cases arise when the connection from the tunnel decapsulator node to
the protocol peer is not an IP forwarding hop, where TTL-decrementing
does not happen (e.g., layer-2 tunneling, bridging, etc). In the
IPsec architecture [RFC4301], another example is the use of Bump-in-
the-Wire (BITW) [BITW].

5.2.1. IP Tunneled over 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. Examples of tunneling IP over
   IP include IP-in-IP [RFC2003], GRE [RFC2784], and various forms of
   IPv6-in-IPv4 (e.g., [RFC4213]). These cases are depicted below.

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

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

   In both cases, the encapsulator (origination tunnel endpoint) is the
   (supposed) sending protocol peer. The TTL in the inner IP datagram
   can be set to 255, since 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.

   In the first case, the encapsulated packet is tunneled directly to
   the protocol peer (also a tunnel endpoint), and therefore the
   encapsulated packet's TTL can be received by the protocol peer with
   an arbitrary value, including 255.

   In the second case, the encapsulated packet is tunneled to a
   decapsulator (tunnel endpoint), which then forwards it to a directly
   connected protocol peer. For IP-in-IP tunnels, RFC 2003 specifies
   the following decapsulator behavior:

       The TTL in the inner IP header is not changed when decapsulating.
       If, after decapsulation, the inner datagram has TTL = 0, the
       decapsulator MUST discard the datagram. If, after decapsulation,
       the decapsulator forwards the datagram to one of its network

interfaces, it will decrement the TTL as a result of doing normal
IP forwarding. See also Section 4.4.

And similarly, for GRE tunnels, RFC 2784 specifies the following
decapsulator behavior:

When a tunnel endpoint decapsulates a GRE packet which has an IPv4
packet as the payload, the destination address in the IPv4 payload
packet header MUST be used to forward the packet and the TTL of
the payload packet MUST be decremented.

Hence the inner IP packet header's TTL, as seen by the decapsulator,
can be set to an arbitrary value (in particular, 255). If the
decapsulator is also the protocol peer, it is possible to deliver the
protocol packet to it with a TTL of 255 (first case). On the other
hand, if the decapsulator needs to forward the protocol packet to a
directly connected protocol peer, the TTL will be decremented (second
case).

5.2.2. IP Tunneled over MPLS

Protocol packets may also be tunneled over MPLS Label Switched Paths
(LSPs) to a protocol peer. The following diagram depicts the
topology.

```
Peer router -------- LSP Termination router and peer
 TTL=255    MPLS LSP   [TTL=x at ingress]
```

MPLS LSPs can operate in Uniform or Pipe tunneling models. The TTL
handling for these models is described in RFC 3443 [RFC3443] that
updates RFC 3032 [RFC3032] in regards to TTL processing in MPLS
networks. RFC 3443 specifies the TTL processing in both Uniform and
Pipe Models, which in turn can used with or without penultimate hop
popping (PHP). The TTL processing in these cases results in
different behaviors, and therefore are analyzed separately. Please
refer to Section 3.1 through Section 3.3 of RFC 3443.

The main difference from a TTL processing perspective between Uniform
and Pipe Models at the LSP termination node resides in how the
incoming TTL (iTTL) is determined. The tunneling model determines
the iTTL: For Uniform Model LSPs, the iTTL is the value of the TTL
field from the popped MPLS header (encapsulating header), whereas for
Pipe Model LSPs, the iTTL is the value of the TTL field from the
exposed header (encapsulated header).

   For Uniform Model LSPs, RFC 3443 states that at ingress:

      For each pushed Uniform Model label, the TTL is copied from the
      label/IP-packet immediately underneath it.

   From this point, the inner TTL (i.e., the TTL of the tunneled IP
   datagram) represents non-meaningful information, and at the egress
   node or during PHP, the ingress TTL (iTTL) is equal to the TTL of the
   popped MPLS header (see Section 3.1 of RFC 3443). In consequence,
   for Uniform Model LSPs of more than one hop, the TTL at ingress
   (iTTL) will be less than 255 (x <= 254), and as a result the check
   described in Section 3 of this document will fail.

   The TTL treatment is identical between Short Pipe Model LSPs without
   PHP and Pipe Model LSPs (without PHP only). For these cases, RFC
   3443 states that:

      For each pushed Pipe Model or Short Pipe Model label, the TTL
      field is set to a value configured by the network operator. In
      most implementations, this value is set to 255 by default.

   In these models, the forwarding treatment at egress is based on the
   tunneled packet as opposed to the encapsulation packet. The ingress
   TTL (iTTL) is the value of the TTL field of the header that is
   exposed, that is the tunneled IP datagram's TTL. The protocol
   packet's TTL as seen by the LSP termination can therefore be set to
   an arbitrary value (including 255). If the LSP termination router is
   also the protocol peer, it is possible to deliver the protocol packet
   with a TTL of 255 (x = 255).

   Finally, for Short Pipe Model LSPs with PHP, the TTL of the tunneled
   packet is unchanged after the PHP operation. Therefore, the same
   conclusions drawn regarding the Short Pipe Model LSPs without PHP and
   Pipe Model LSPs (without PHP only) apply to this case. For Short
   Pipe Model LSPs, the TTL at egress has the same value with or without
   PHP.

   In conclusion, GTSM checks are possible for IP tunneled over Pipe
   model LSPs, but not for IP tunneled over Uniform model LSPs.
   Additionally, for all tunneling modes, if the LSP termination router
   needs to forward the protocol packet to a directly connected protocol
   peer, it is not possible to deliver the protocol packet to the
   protocol peer with a TTL of 255. If the packet is further forwarded,
   the outgoing TTL (oTTL) is calculated by decrementing iTTL by one.

5.3. Onlink Attackers

   As described in Section 2, an attacker directly connected to one
   interface can disturb a GTSM-protected session on the same or another
   interface (by spoofing a GTSM peer's address) unless ingress
   filtering has been applied on the connecting interface. As a result,
   interfaces that do not include such protection need to be trusted not
   to originate attacks on the router.

5.4. Fragmentation Considerations

   As mentioned, fragmentation may restrict the amount of information
   available for classification. Since non-initial IP fragments do not
   contain Layer 4 information, it is highly likely that they cannot be
   associated with a registered GTSM-enabled session. Following the
   receiving protocol procedures described in Section 3, non-initial IP
   fragments would likely be classified with Unknown trustworthiness.
   And since the IP packet would need to be reassembled in order to be
   processed, the end result is that the initial-fragment of a GTSM-
   enabled session effectively receives the treatment of an Unknown-
   trustworthiness packet, and the complete reassembled packet receives
   the aggregate of the Unknowns.

   In principle, an implementation could remember the TTL of all
   received fragments. Then when reassembling the packet, verify that
   the TTL of all fragments match the required value for an associated
   GTSM-enabled session. In the likely common case that the
   implementation does not do this check on all fragments, then it is
   possible for a legitimate first fragment (which passes the GTSM
   check) to be combined with spoofed non-initial fragments, implying
   that the integrity of the received packet is unknown and unprotected.
   If this check is performed on all fragments at reassembly, and some
   fragment does not pass the GTSM check for a GTSM-enabled session, the
   reassembled packet is categorized as a Dangerous-trustworthiness
   packet and receives the corresponding treatment.

   Further, reassembly requires to wait for all the fragments and
   therefore likely invalidates or weakens the fifth assumption
   presented in Section 2: it may not be possible to classify non-
   initial fragments before going through a scarce resource in the
   system, when fragments need to be buffered for reassembly and later
   processed by a CPU. That is, when classification cannot be done with
   the required granularity, non-initial fragments of GTSM-enabled
   session packets would not use different resource pools.

   Consequently, to get practical protection from fragment attacks,
   operators may need to rate-limit or discard all received fragments.
   As such, it is highly RECOMMENDED for GTSM-protected protocols to

avoid fragmentation and reassembly by manual MTU tuning, using
adaptive measures such as Path MTU Discovery (PMTUD), or any other
available method [RFC1191], [RFC1981], or [RFC4821].

5.5. Multi-Hop Protocol Sessions

GTSM could possibly offer some small, though difficult to quantify,
degree of protection when used with multi-hop protocol sessions (see
Appendix A). In order to avoid having to quantify the degree of
protection and the resulting applicability of multi-hop, we only
describe the single-hop case because its security properties are
clearer.

6. Applicability Statement

GTSM is only applicable to environments with inherently limited
topologies (and is most effective in those cases where protocol peers
are directly connected). In particular, its application should be
limited to those cases in which protocol peers are directly
connected.

GTSM will not protect against attackers who are as close to the
protected station as its legitimate peer. For example, if the
legitimate peer is one hop away, GTSM will not protect from attacks
from directly connected devices on the same interface (see
Section 2.2 for more).

Experimentation on GTSM's applicability and security properties is
needed in multi-hop scenarios. The multi-hop scenarios where GTSM
might be applicable is expected to have the following
characteristics: the topology between peers is fairly static and
well-known, and in which the intervening network (between the peers)
is trusted.

6.1. Backwards Compatibility

RFC 3682 [RFC3682] did not specify how to handle "related messages"
(ICMP errors). This specification mandates setting and verifying
TTL=255 of those as well as the main protocol packets.

Setting TTL=255 in related messages does not cause issues for RFC
3682 implementations.

Requiring TTL=255 in related messages may have impact with RFC 3682
implementations, depending on which default TTL the implementation
uses for originated packets; some implementations are known to use
255, while 64 or other values are also used. Related messages from
the latter category of RFC 3682 implementations would be classified

as Dangerous and treated as described in Section 3. This is not
believed to be a significant problem because protocols do not depend
on related messages (e.g., typically having a protocol exchange for
closing the session instead of doing a TCP-RST), and indeed the
delivery of related messages is not reliable. As such, related
messages typically provide an optimization to shorten a protocol
keepalive timeout. Regardless of these issues, given that related
messages provide a significant attack vector to e.g., reset protocol
sessions, making this further restriction seems sensible.

7. References

7.1. Normative References

   [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
              September 1981.

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

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

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

   [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P.
              Traina, "Generic Routing Encapsulation (GRE)", RFC 2784,
              March 2000.

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

   [RFC3443] Agarwal, P. and B. Akyol, "Time To Live (TTL) Processing
              in Multi-Protocol Label Switching (MPLS) Networks",
              RFC 3443, January 2003.

   [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms
              for IPv6 Hosts and Routers", RFC 4213, October 2005.

   [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway
              Protocol 4 (BGP-4)", RFC 4271, January 2006.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, December 2005.

7.2. Informative References

   [BITW]   "Thread: 'IP-in-IP, TTL decrementing when forwarding and
               BITW' on int-area list, Message-ID:
               <Pine.LNX.4.64.0606020830220.12705@netcore.fi>",
               June 2006, <http://www1.ietf.org/mail-archive/web/
               int-area/current/msg00267.html>.

   [RFC1191]   Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
               November 1990.

   [RFC1981]   McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery
               for IP version 6", RFC 1981, August 1996.

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

   [RFC3682]   Gill, V., Heasley, J., and D. Meyer, "The Generalized TTL
               Security Mechanism (GTSM)", RFC 3682, February 2004.

   [RFC3704]   Baker, F. and P. Savola, "Ingress Filtering for Multihomed
               Networks", BCP 84, RFC 3704, March 2004.

   [RFC4272]   Murphy, S., "BGP Security Vulnerabilities Analysis",
               RFC 4272, January 2006.

   [RFC4821]   Mathis, M. and J. Heffner, "Packetization Layer Path MTU
               Discovery", RFC 4821, March 2007.

Appendix A. Multi-Hop GTSM

   NOTE: This is a non-normative part of the specification.

   The main applicability of GTSM is for directly connected peers. GTSM
   could be used for non-directly connected sessions as well, where the
   recipient would check that the TTL is within a configured number of
   hops from 255 (e.g., check that packets have 254 or 255). As such
   deployment is expected to have a more limited applicability and
   different security implications, it is not specified in this
   document.

Appendix B. Changes Since RFC 3682

   o Bring the work on the Standards Track (RFC 3682 was Experimental).

   o New text on GTSM applicability and use in new and existing
      protocols.

   o Restrict the scope to not specify multi-hop scenarios.

   o Explicitly require that related messages (ICMP errors) must also
      be sent and checked to have TTL=255. See Section 6.1 for
      discussion on backwards compatibility.

   o Clarifications relating to fragmentation, security with tunneling,
      and implications of ingress filtering.

   o A significant number of editorial improvements and clarifications.

Authors' Addresses

   Vijay Gill
   EMail: vijay@umbc.edu

   John Heasley
   EMail: heas@shrubbery.net

   David Meyer
   EMail: dmm@1-4-5.net

   Pekka Savola (editor)
   Espoo
   Finland
   EMail: psavola@funet.fi

   Carlos Pignataro
   EMail: cpignata@cisco.com