                  Security Concerns with IP Tunneling

Abstract

   A number of security concerns with IP tunnels are documented in this
   memo. The intended audience of this document includes network
   administrators and future protocol developers. The primary intent of
   this document is to raise the awareness level regarding the security
   issues with IP tunnels as deployed and propose strategies for the
   mitigation of those issues.

Table of Contents

1. Introduction

   With NAT devices becoming increasingly more prevalent, there have
   recently been many tunneling protocols developed that go through NAT
   devices or firewalls by tunneling over UDP or TCP. For example,
   Teredo [RFC4380], Layer Two Tunneling Protocol Version 2 (L2TPv2)
   [RFC2661], and Layer Two Tunneling Protocol Version 3 (L2TPv3)
   [RFC3931] all tunnel IP packets over UDP. Similarly, many Secure
   Socket Layer (SSL) VPN solutions that tunnel IP packets over HTTP
   (and hence over TCP) are deployed today.

   This document discusses security concerns with tunneling IP packets
   and includes recommendations where relevant.

   The primary intent of this document is to help improve security
   deployments using tunnel protocols. In addition, the document aims
   to provide information that can be used in any new or updated tunnel
   protocol specification. The intended audience of this document
   includes network administrators and future protocol developers.

2. Tunnels May Bypass Security

2.1. Network Security Bypass

2.1.1. Problem

   Tunneled IP traffic may not receive the intended level of inspection
   or policy application by network-based security devices unless such
   devices are specifically tunnel aware. This reduces defense in depth
   and may cause security gaps. This applies to all network-located
   devices and to any end-host-based firewalls whose existing hooking
   mechanism(s) would not show them the IP packet stream after the
   tunnel client does decapsulation or before it does encapsulation.

2.1.2. Discussion

   Evasion by tunneling is often a problem for network-based security
   devices such as network firewalls, intrusion detection and prevention
   systems, and router controls. To provide such functionality in the
   presence of tunnels, the developer of such devices must add support
   for parsing each new protocol. There is typically a significant lag
   between when the security developer recognizes that a tunnel will be
   used (or will be remotely usable) to a significant degree and when
   the parsing can be implemented in a product update, the update can be
   tested and released, and customers can begin using the update. Late
   changes in the protocol specification or in the way it is implemented
   can cause additional delays. This becomes a significant security
   concern when a delay in applied coverage is occurring frequently.

One way to cut down on this lag is for security developers to follow
the progress of new IETF protocols, but this will still not account
for any new proprietary protocols.

For example, for L2TP or Teredo, an unaware network security device
would inspect or regulate the outer IP and the IP-based UDP layer as
normal, but it would not recognize that there is an additional IP
layer contained inside the UDP payload to which it needs to apply the
same controls as it would to a native packet. (Of course, if the
device discards the packet due to something in the IP or UDP header,
such as referring to an unknown protocol, the embedded packet is no
longer a concern.) In addition, if the tunnel does encryption, the
network-based security device may not be able to do much, just as if
IPsec end-to-end encryption were used without tunneling.

Network security controls not being applied must be a concern to
those that set them up, since those controls are supposed to provide
an additional layer of defense against external attackers. If
network controls are being bypassed due to the use of tunneling, the
strength of the defense (i.e., the number of layers of defense) is
reduced. Since security administrators may have a significantly
reduced level of confidence without this layer, this becomes a
concern to them.

One implication of the security control bypass is that defense in
depth has been reduced, perhaps down to zero unless a local firewall
is in use as recommended in [RFC4380]. However, even if there are
host-based security controls that recognize tunnels and all controls
that were maintained by the network are available on the host,
security administrators may not have configured them with full
security control parity. Thus, there may be gaps in desired
coverage.

Compounding this is that, unlike what would be the case for native
IP, some network administrators will not even be aware that their
hosts are globally reachable if the tunnel provides connectivity
to/from the Internet; for example, they may not be expecting this for
hosts behind a stateful firewall. In addition, Section 3.2 discusses
how it may not be efficient to find all tunneled traffic for network
devices to examine.

2.1.3. Recommendations

Security administrators who do not consider tunneling an acceptable
risk should disable tunnel functionality either on the end nodes
(hosts) or on the network nodes at the perimeter of their network.
However, there may be an awareness gap. Thus, due to the possible
negative security consequences, tunneling functionality should be

easy to disable on the host and through a central management facility
if one is provided.

To minimize security exposure due to tunnels, we recommend that a
tunnel be an interface of last resort, independent of IP version.
Specifically, we suggest that when both native and tunneled access to
a remote host is available, the native access be used in preference
to tunneled access except when the tunnel endpoint is known to not
bypass security (e.g., an IPsec tunnel to a gateway provided by the
security administrator of the network). This should also promote
greater efficiency and reliability.

Note that although Rule 7 of [RFC3484], Section 6 will prefer native
connectivity over tunnels, this rule is only a tie-breaker when a
choice is not made by earlier rules; hence, tunneling mechanisms that
are tied to a particular range of IP address space will be decided
based on the prefix precedence. For example, using the prefix policy
mechanism of [RFC3484], Section 2.1, Teredo might have a precedence
of 5 so that native IPv4 is preferred over Teredo.

2.2. IP Ingress and Egress Filtering Bypass

2.2.1. Problem

IP addresses inside tunnels are not subject to ingress and egress
filtering in the network they tunnel over, unless extraordinary
measures are taken. Only the tunnel endpoints can do such filtering.

2.2.2. Discussion

Ingress filtering (sanity-checking incoming destination addresses)
and egress filtering (sanity-checking outgoing source addresses) are
done to mitigate attacks and to make it easier to identify the source
of a packet and are considered to be a good practice. For example,
ingress filtering at the network perimeter should not allow packets
with a source address that belongs to the network to enter the
network from outside the network. This function is most naturally
(and in the general case, by requirement) done at network boundaries.
Tunneled IP traffic bypassing this network control is a specific case
of Section 2.1, but is illustrative.

2.2.3. Recommendations

Tunnel servers can apply ingress and egress controls to tunneled IP
addresses passing through them to and from tunnel clients.

Tunnel clients could make an effort to conduct ingress and egress
filtering.

Implementations of protocols that embed an IPv4 address in a tunneled
IPv6 address directly between peers should perform filtering based on
checking the correspondence.

Implementations of protocols that accept tunneled packets directly
from a server, relay, or protocol peer do filtering the same way as
it would be done on a native link with traffic from a router.

Some protocols such as 6to4 [RFC3056], Teredo, and the Intra-Site
Automatic Tunnel Addressing Protocol (ISATAP) [RFC5214] allow both
other hosts and a router over a common tunnel. To perform host-based
filtering with such protocols, a host would need to know the outer IP
address of each router from which it could receive traffic, so that
packets from hosts beyond the router will be accepted even though the
source address would not embed the router's IP address. Router
addresses might be learned via SEcure Neighbor Discovery (SEND)
[RFC3971] or some other mechanism (e.g., [RFC5214], Section 8.3.2).

2.3. Source Routing after the Tunnel Client

2.3.1. Problem

If the encapsulated IP packet specifies source routing beyond the
recipient tunnel client, the host may forward the IP packet to the
specified next hop. This may be unexpected and contrary to
administrator wishes and may have bypassed network-based source-
routing controls.

2.3.2. Discussion

A detailed discussion of issues related to source routing can be
found in [RFC5095] and [SECA-IP].

2.3.3. Recommendations

Tunnel clients should by default discard tunneled IP packets that
specify additional routing, as recommended in [RFC5095] and
[SECA-IP], though they may also allow the user to configure what
source-routing types are allowed. All pre-existing source-routing
controls should be upgraded to apply these controls to tunneled IP
packets as well.

3. Challenges in Inspecting and Filtering Content of Tunneled Data
    Packets

3.1. Inefficiency of Selective Network Filtering of All Tunneled
      Packets

3.1.1. Problem

   There is no mechanism that both efficiently and immediately filters
   all tunneled packets (other than the obviously faulty method of
   filtering all packets). This limits the ability to prevent tunnel
   use on a network.

3.1.2. Discussion

   Given concerns about tunnel security or a network's lack of
   preparedness for tunnels, a network administrator may wish to simply
   block all use of tunnels that bypass security policies. He or she
   may wish to do so using network controls; this could be either due to
   not having the capability to disable tunneling on all hosts attached
   to the network or due to wanting an extra layer of prevention.

   One simple method of doing this easily for many tunnel protocols is
   to block outbound packets to the UDP or TCP port used (e.g.,
   destination UDP port is 3544 for Teredo, UDP port 1701 for L2TP,
   etc.). This prevents a tunnel client from establishing a new tunnel.
   However, existing tunnels will not necessarily be affected if the
   blocked port is used only for initial setup. In addition, if the
   blocking is applied on the outside of the client's NAT device, the
   NAT device will retain the port mapping for the client. In some
   cases, however, blocking all traffic to a given outbound port (e.g.,
   port 80) may interfere with non-tunneled traffic so this should be
   used with caution.

   Another simple alternative, if the tunnel server addresses are well-
   known, is to filter out all traffic to/from such addresses.

   The other approach is to find all packets to block in the same way as
   would be done for inspecting all packets (Section 3.2). However,
   this presents difficulties in terms of efficiency of filtering, as is
   discussed in Section 3.2.

3.1.3. Recommendations

   Developers of protocols that tunnel over UDP or TCP (including HTTP)
   to reach the Internet should disable their protocols in networks that
   wish to enforce security policies on the user traffic. (Windows, for
   example, disables Teredo by default if it detects that it is within
   an enterprise network that contains a Windows domain controller.)

   Administrators of such networks may wish to filter all tunneled
   traffic at the boundaries of their networks. It is sufficient to
   filter out the tunneled connection requests (if they can be
   identified) to stop further tunneled traffic. The easiest mechanism
   for this would be to filter out outgoing traffic sent to the
   destination port defined by the tunneling protocol and incoming
   traffic with that source port. Similarly, in certain cases, it is
   also possible to use the IP protocol field to identify and filter
   tunneled packets. For example, 6to4 [RFC3056] is a tunneling
   mechanism that uses IPv4 packets to carry encapsulated IPv6 packets
   and can be identified by the IPv4 protocol type 41.

3.2. Problems with Deep Packet Inspection of Tunneled Data Packets

3.2.1. Problem

   There is no efficient mechanism for network-based devices, which are
   not the tunnel endpoint, to inspect the contents of all tunneled data
   packets the way they can for native packets. This makes it difficult
   to apply the same controls as they do to native IP.

3.2.2. Discussion

   Some tunnel protocols are easy to identify, such as if all data
   packets are encapsulated using a well-known UDP or TCP port that is
   unique to the protocol.

   Other protocols, however, either use dynamic ports for data traffic
   or else share ports with other protocols (e.g., tunnels over HTTP).

   The implication of this is that network-based devices that wish to
   passively inspect (and perhaps selectively apply policy to) all
   encapsulated traffic must inspect all TCP or UDP packets (or at least
   all packets not part of a session that is known not to be a tunnel).
   This is imperfect since a heuristic must then be applied to determine
   if a packet is indeed part of a tunnel. This may be too slow to make
   use of in practice, especially if it means that all TCP or UDP
   packets must be taken off of the device's "fast path".

One heuristic that can be used on packets to determine if they are
tunnel-related or not is as follows. For each known tunnel protocol,
attempt parsing the packet as if it were a packet of that protocol
destined to the local host (i.e., where the local host has the
destination address in the inner IP header, if any). If all syntax
checks pass, up to and including the inner IP header (if the tunnel
does not use encryption), then treat the packet as if it were a
tunneled packet of that protocol.

It is possible that non-tunneled packets will be treated as if they
were tunneled packets using this heuristic, but tunneled packets (of
the known types of tunnels) should not escape inspection, absent
implementation bugs.

For some protocols, it may be possible to monitor setup exchanges to
know to expect that data will be exchanged on certain ports later.
(Note that this does not necessarily apply to Teredo, for example,
since communicating with another Teredo client behind a cone NAT
[RFC5389] device does not require such signaling. In such cases this
control will not work. However, deprecation of the cone bit as
discussed in [RFC5991] means this technique may indeed work with
updated Teredo implementations.)

3.2.3. Recommendations

As illustrated above, it should be clear that inspecting the contents
of tunneled data packets is highly complex and often impractical.
For this reason, if a network wishes to monitor IP traffic, tunneling
across, as opposed to tunneling to, the security boundary is not
recommended. For example, to provide an IPv6 transition solution,
the network should provide native IPv6 connectivity or a tunnel
solution (e.g., ISATAP or 6over4 [RFC2529]) that encapsulates data
packets between hosts and a router within the network.

4. Increased Exposure Due to Tunneling

4.1. NAT Holes Increase Attack Surface

4.1.1. Problem

If the tunnel allows inbound access from the public Internet, the
opening created in a NAT device due to a tunnel client increases its
Internet attack surface area. If vulnerabilities are present, this
increased exposure can be used by attackers and their programs.

If the tunnel allows inbound access only from a private network
(e.g., a remote network to which one has VPNed), the opening created
in the NAT device still increases its attack surface area, although
not as much as in the public Internet case.

4.1.2. Discussion

When a tunnel is active, a mapped port is maintained on the NAT
device through which remote hosts can send packets and perhaps
establish connections. The following sequence is intended to sketch
out the processing on the tunnel client host that can be reached
through this mapped port; the actual processing for a given host may
be somewhat different.

1. Link-layer protocol processing

2. (Outer) IP host firewall processing

3. (Outer) IP processing by stack

4. UDP/TCP processing by stack

5. Tunnel client processing

6. (Inner) IP host firewall processing

7. (Inner) IP processing by stack

8. Various upper layer processing may follow

The inner firewall (and other security) processing may or may not be
present, but if it is, some of the inner IP processing may be
filtered. (For example, [RFC4380], Section 7.1 recommends that an
IPv6 host firewall be used on all Teredo clients.)

(By the virtue of the tunnel being active, we can infer that the
inner host firewall is unlikely to do any filtering based on the
outer IP.) Any of this processing may expose vulnerabilities an
attacker can exploit; similarly, these may expose information to an
attacker. Thus, even if firewall filtering is in place (as is
prudent) and filters all incoming packets, the exposed area is larger
than if a native IP Internet connection were in place, due to the
processing that takes place before the inner IP is reached
(specifically, the UDP/TCP processing, the tunnel client processing,
and additional IP processing, especially if one is IPv4 and the other
is IPv6).

One possibility is that a layer 3 (L3) targeted worm makes use of a
vulnerability in the exposed processing. The main benefit tunneling
provides to worms is enabling L3 reachability to the end host. Even
a thoroughly firewalled host could be subject to a worm that spreads
with a single UDP packet if the right remote code vulnerability is
present.

4.1.3. Recommendation

This problem seems inherent in tunneling being active on a host, so
the solution seems to be to minimize tunneling use.

For example, tunneling can be active only when it is really needed
and only for as long as needed. So, the tunnel interface can be
initially not configured and only used when it is entirely the last
resort. The interface should then be deactivated (ideally,
automatically) again as soon as possible. Note, however, that the
hole will remain in the NAT device for some amount of time after
this, so some processing of incoming packets is inevitable unless the
client's native IP address behind the NAT device is changed.

4.2. Exposure of a NAT Hole

4.2.1. Problem

Attackers are more likely to know about a tunnel client's NAT hole
than a typical hole in the NAT device. If they know about the hole,
they could try to use it.

4.2.2. Discussion

There are at least three reasons why an attacker may be more likely
to learn of the tunnel client's exposed port than a typical NAT
exposed port:

1. The NAT mapping for a tunnel is typically held open for a
    significant period of time and kept stable. This increases the
    chance of it being discovered.

2. In some protocols (e.g., Teredo), the external IP address and
    port are contained in the client's address that is used end-to-
    end and possibly even advertised in a name resolution system.
    While the tunnel protocol itself might only distribute this
    address in IP headers, peers, routers, and other on-path nodes
    still see the client's IP address. Although this point does not
    apply directly to protocols that do not construct the inner IP
    address based on the outer IP address (e.g., L2TP), the inner IP

      address is still known to peers, routers, etc., and can still be
      reached by attackers without their knowing the external IP
      address or port.

   3. Sending packets over a tunnel often results in more message
       exchanges due to the tunneling protocol, as well as messages
       being seen by more parties (e.g., due to a longer path length),
       than sending packets natively, offering more chances for
       visibility into the port and address in use.

4.2.3. Recommendation

   The recommendation from Section 4.1 seems to apply here as well:
   minimize tunnel use.

4.3. Public Tunnels Widen Holes in Restricted NATs

4.3.1. Problem

   Tunnels that allow inbound connectivity from the Internet (e.g.,
   Teredo, tunnel brokers, etc.) essentially disable the filtering
   behavior of the NAT for all tunnel client ports. This eliminates NAT
   devices filtering for such ports and may eliminate the need for an
   attacker to spoof an address.

4.3.2. Discussion

   NATs that implement Address-Dependent or Address and Port-Dependent
   Filtering [RFC4787] limit the source of incoming packets to just
   those that are a previous destination. This poses a problem for
   tunnels that intend to allow inbound connectivity from the Internet.

   Various protocols (e.g., Teredo, Session Traversal Utilities for NAT
   (STUN) [RFC5389], etc.) provide a facility for peers, upon request,
   to become a previous destination. This works by sending a "bubble"
   packet via a server, which is passed to the client and then sent by
   the client (through the NAT) to the originator.

   This removes any NAT-based barrier to attackers sending packets in
   through the client's service port. In particular, an attacker would
   no longer need to either be an actual previous destination or forge
   its addresses as a previous destination. When forging, the attacker
   would have had to learn of a previous destination and then would face
   more challenges in seeing any returned traffic.

4.3.3. Recommendations

   If the tunnel can provide connectivity to the Internet, the tunnel
   client should run a host firewall on the tunnel interface. Also,
   minimizing public tunnel use (see Section 4.1.3) would lower the
   attack opportunity related to this exposure.

5. Tunnel Address Concerns

5.1. Feasibility of Guessing Tunnel Addresses

5.1.1. Problem

   For some types of tunneling protocols, it may be feasible to guess IP
   addresses assigned to tunnels, either when looking for a specific
   client or when looking for an arbitrary client. This is in contrast
   to native IPv6 addresses in general but is no worse than for native
   IPv4 addresses today.

   For example, some protocols (e.g., 6to4 and Teredo) use well-defined
   address ranges. As another example, using well-known public servers
   for Teredo or tunnel brokers also implies using a well-known address
   range.

5.1.2. Discussion

   Several tunnel protocols use endpoint addresses that can be
   algorithmically derived from some known values. These addresses are
   structured, and the fields contained in them can be fairly
   predictable. This reduces the search space for an attacker and
   reduces the resistance of the address to scanning attacks.

5.1.3. Recommendations

   It is recommended that tunnel protocol developers use tunnel endpoint
   addresses that are not easily guessable. When the tunnel endpoint
   addresses are structured and fairly guessable, it is recommended that
   the implementation use any unused fields in the address to provide
   additional entropy to the address in order to reduce the address-
   scanning risks. For example, this could be done by setting these
   unused fields to some random values.

5.2. **Profiling Targets Based on Tunnel Address**

5.2.1. **Problem**

   An attacker encountering an address associated with a particular
   tunneling protocol or well-known tunnel server has the opportunity to
   infer certain relevant pieces of information that can be used to
   profile the host before sending any packets. This can reduce the
   attacker's footprint and increase the attacker's efficiency.

5.2.2. **Discussion**

   The tunnel address reveals some information about the nature of the
   client:

   o That a host has a tunnel address associated with a given protocol
      means that the client is running on some platform for which there
      exists a tunnel client implementation of that protocol. In
      addition, if some platforms have that protocol installed by
      default and if the host's default rules for using it make it
      susceptible to being in use, then the protocol is more likely to
      be running on such a platform than on one where it is not used by
      default. For example, as of this writing, seeing a Teredo address
      suggests that the host it is on is probably running Windows.

   o Similarly, the use of an address associated with a particular
      tunnel server also suggests some information. Tunnel client
      software is often deployed, installed, and/or configured using
      some degree of automation. It seems likely that the majority of
      the time, the tunnel server that results from the initial
      configuration will go unchanged from the initial setting.
      Moreover, the server that is configured for use may be associated
      with a particular means of installation, which often suggests the
      platform. For example, if the server field in a Teredo address is
      one of the IPv4 addresses to which teredo.ipv6.microsoft.com
      resolves, the host is likely running Windows.

   o The external IPv4 address of a NAT device can, of course, be
      readily associated with a particular organization or at least an
      ISP; hence, putting this address in an IPv6 address reveals this
      information. However, this is no different than using a native IP
      address and is therefore not new with tunneling.

   o It is also possible that external client port numbers may be more
      often associated with particular client software or the platform
      on which it is running. The usefulness of this for platform
      determination is, however, reduced by the different NAT port

number assignment behaviors. In addition, the same observations
would apply to use of UDP or TCP over native IP as well; hence,
this is not new with tunneling.

The platform, tunnel client software, or organization information can
be used by an attacker to target attacks more carefully. For
example, an attacker may decide to attack an address only if it is
likely to be associated with a particular platform or tunnel client
software with a known vulnerability. (This is similar to the ability
to guess some platforms based on the Organizationally Unique
Identifier (OUI) in the Extended Unique Identifier (EUI)-64 portion
of an IPv6 address generated from a Media Access Control (MAC)
address, since some platforms are commonly used with interface cards
from particular vendors.)

5.2.3. Recommendations

If installation programs randomize the server setting, they would
reduce the extent to which they can be profiled. Similarly,
administrators can choose to change the default setting to reduce the
degree to which they can be profiled ahead of time.

Randomizing the tunnel client port in use would mitigate any
profiling that can be done based on the external port, especially if
multiple tunnel clients did this. Further discussion on randomizing
ports can be found at [RFC6056].

It is recommended that tunnel protocols minimize the propagation of
knowledge about whether the NAT is a cone NAT.

6. Additional Security Concerns

6.1. Attacks Facilitated by Changing Tunnel Server Setting

6.1.1. Problem

If an attacker could change either a tunnel client's server setting
or the IP addresses to which a configured host name resolves (e.g.,
by intercepting DNS queries) AND if the tunnel is not authenticated,
the attacker would become a man in the middle. This would allow them
to at least monitor peer communication and at worst to impersonate
the remote peer.

6.1.2. Discussion

A client's server has good visibility into the client's communication
with IP peers. If the server were switched to one that records this
information and makes it available to third parties (e.g.,

advertisers, competitors, spouses, etc.), then sensitive information
would be disclosed, especially if the client's host prefers the
tunnel over native IP. Assuming the server provides good service,
the user would not have reason to suspect the change.

Full interception of IP traffic could also be arranged (including
pharming), which would allow any number of deception or monitoring
attacks, including phishing. We illustrate this with an example
phishing attack scenario.

It is often assumed that the tunnel server is a trusted entity. It
may be possible for malware or a malicious user to quietly change the
client's tunnel server setting and have the user be unaware that
their trust has been misplaced for an indefinite period of time.
However, malware or a malicious user can do much worse than this, so
this is not a significant concern. Hence, it is only important that
an attacker on the network cannot change the client's server setting.

1. A phisher sets up a malicious tunnel server (or tampers with a
    legitimate one). This server, for the most part, provides
    correct service.

2. An attacker, by some means, switches the host's tunnel server
    setting or spoofs a DNS reply to point to the above server. If
    neither DNS nor the tunnel setup is secured (i.e., if the client
    does not authenticate the information), then the attacker's
    tunnel server is seen as legitimate.

3. A user on the victim host types their bank's URL into his/her
    browser.

4. The bank's hostname resolves to one or more IP addresses, and the
    tunnel is selected for socket connection for whatever reason
    (e.g., the tunnel provides IPv6 connectivity, and the bank has an
    IPv6 address).

5. The tunnel client uses the server for help in connecting to the
    bank's IP address. Some tunneling protocols use a separate
    channel for signaling versus data, but this still allows the
    server to place itself in the data path by an appropriate signal
    to the client. For example, in Teredo, the client sends a ping
    request through a server, which is expected to come back through
    a data relay, and a malicious server can simply send it back
    itself to indicate that is a data relay for the communication.

   6. The rest works pretty much like any normal phishing transaction,
       except that the attacker acts as a tunnel server (or data relay,
       for protocols such as Teredo) and a host with the bank's IP
       address.

   This pharming-type attack is not unique to tunneling. Switching DNS
   server settings to a malicious DNS server or DNS cache poisoning in a
   recursive DNS resolver could have a similar effect.

6.1.3. Recommendations

   In general, anti-phishing and anti-fraud provisions should help with
   aspects of this, as well as software that specifically monitors for
   tunnel server changes.

   Perhaps the best way to mitigate tunnel-specific attacks is to have
   the client authenticate either the tunnel server or at least the
   means by which the tunnel server's IP address is determined. For
   example, SSL VPNs use https URLs and hence authenticate the server as
   being the expected one. When IPv6 Router Advertisements are sent
   over the tunnel, another mechanism is to use SEcure Neighbor
   Discovery (SEND) [RFC3971] to verify that the client trusts the
   server.

   On the host, it should require an appropriate level of privilege in
   order to change the tunnel server setting (as well as other non-
   tunnel-specific settings such as the DNS server setting, etc.).
   Making it easy to see the current tunnel server setting (e.g., not
   requiring privilege for this) should help detection of changes.

   The scope of the attack can also be reduced by limiting tunneling use
   in general but especially in preferring native IPv4 to tunneled IPv6
   when connecting to peers who are accessible over IPv4, as doing so
   helps mitigate attacks that are facilitated by changing the tunnel
   server setting. Please refer to Section 3 of [TUNNEL-LOOPS] for a
   detailed description and mitigation measures for a class of attacks
   based on IPv6 automatic tunnels.

7. Mechanisms to Secure the Use of Tunnels

   This document described several security issues with tunnels. This
   does not mean that tunnels need to be avoided at any cost. On the
   contrary, tunnels can be very useful if deployed, operated, and used
   properly. The threats against IP tunnels are documented here. If
   the threats can be mitigated, network administrators can efficiently
   and securely use tunnels in their network. Several measures can be
   taken in order to secure the operation of IPv6 tunnels:

   o Operating on-premise tunnel servers/relays so that the tunneled
      traffic does not cross border routers.

   o Setting up internal routing to steer traffic to these servers/
      relays

   o Setting up of firewalls [RFC2979] to allow known and controllable
      tunneling mechanisms and disallow unknown tunnels.

8. Acknowledgments

   The authors would like to thank Remi Denis-Courmont, Fred Templin,
   Jordi Palet Martinez, James Woodyatt, Christian Huitema, Brian
   Carpenter, Nathan Ward, Kurt Zeilenga, Joel Halpern, Erik Kline,
   Alfred Hoenes, and Fernando Gont for reviewing earlier versions of
   the document and providing comments to make this document better.

9. Security Considerations

   This entire document discusses security concerns with tunnels.

10. Informative References

   [RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4
              Domains without Explicit Tunnels", RFC 2529, March 1999.

   [RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn,
              G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"",
              RFC 2661, August 1999.

   [RFC2979] Freed, N., "Behavior of and Requirements for Internet
              Firewalls", RFC 2979, October 2000.

   [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains
              via IPv4 Clouds", RFC 3056, February 2001.

   [RFC3484] Draves, R., "Default Address Selection for Internet
              Protocol version 6 (IPv6)", RFC 3484, February 2003.

   [RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed.,
              "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC
              3931, March 2005.

   [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander,
              "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

   [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through
              Network Address Translations (NATs)", RFC 4380, February
              2006.

   [RFC4787] Audet, F., Ed., and C. Jennings, "Network Address
              Translation (NAT) Behavioral Requirements for Unicast
              UDP", BCP 127, RFC 4787, January 2007.

   [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation
              of Type 0 Routing Headers in IPv6", RFC 5095, December
              2007.

   [RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site
              Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214,
              March 2008.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              October 2008.

   [RFC5991] Thaler, D., Krishnan, S., and J. Hoagland, "Teredo
              Security Updates", RFC 5991, September 2010.

   [RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-
              Protocol Port Randomization", BCP 156, RFC 6056, January
              2011.

   [SECA-IP] Gont, F., "Security Assessment of the Internet Protocol
              version 4", Work in Progress, April 2011.

   [TUNNEL-LOOPS]
              Nakibly, G. and F. Templin, "Routing Loop Attack using
              IPv6 Automatic Tunnels: Problem Statement and Proposed
              Mitigations", Work in Progress, March 2011.

Authors' Addresses

   Suresh Krishnan
   Ericsson
   8400 Decarie Blvd.
   Town of Mount Royal, QC
   Canada

   Phone: +1 514 345 7900 x42871
   EMail: suresh.krishnan@ericsson.com

   Dave Thaler
   Microsoft Corporation
   One Microsoft Way
   Redmond, WA 98052
   USA

   Phone: +1 425 703 8835
   EMail: dthaler@microsoft.com

   James Hoagland
   Symantec Corporation
   350 Ellis St.
   Mountain View, CA 94043
   USA

   EMail: Jim_Hoagland@symantec.com
   URI: http://symantec.com/