                 Defending TCP Against Spoofing Attacks
                   draft-ietf-tcpm-tcp-antispoof-06.txt

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 August 23, 2007.

Copyright Notice

Abstract

   Recent analysis of potential attacks on core Internet infrastructure
   indicates an increased vulnerability of TCP connections to spurious
   resets (RSTs), sent with forged IP source addresses (spoofing). TCP
   has always been susceptible to such RST spoofing attacks, which were
   indirectly protected by checking that the RST sequence number was
   inside the current receive window, as well as via the obfuscation of

TCP endpoint and port numbers. For pairs of well-known endpoints
often over predictable port pairs, such as BGP or between web servers
and well-known large-scale caches, increases in the path bandwidth-
delay product of a connection have sufficiently increased the receive
window space that off-path third parties can brute-force generate a
viable RST sequence number. The susceptibility to attack increases
with the square of the bandwidth, thus presents a significant
vulnerability for recent high-speed networks. This document
addresses this vulnerability, discussing proposed solutions at the
transport level and their inherent challenges, as well as existing
network level solutions and the feasibility of their deployment.
This document focuses on vulnerabilities due to spoofed TCP segments,
and includes a discussion of related ICMP spoofing attacks on TCP
connections.

Table of Contents

1. Introduction

   Analysis of the Internet infrastructure has recently demonstrated a
   new version of a vulnerability in BGP connections between core
   routers using an attack based on RST spoofing from off-path attackers
   [9][10][47]. This attack has been known for nearly six years [20].
   Such connections, typically using TCP, can be susceptible to off-path
   third-party reset (RST) segments with forged source addresses
   (spoofed), which terminate the TCP connection. BGP routers react to
   a terminated TCP connection in various ways which can amplify the
   impact of an attack, ranging from restarting the connection to
   deciding that the other router is unreachable and thus flushing the
   BGP routes [37]. This sort of attack affects other protocols besides
   BGP, involving any long-lived connection between well-known
   endpoints. The impact on the Internet infrastructure can be
   substantial (esp. for the BGP case), and warrants immediate
   attention.

   TCP, like many other protocols, can be susceptible to these off-path
   third-party spoofing attacks. Such attacks rely on the increase of
   commodity platforms supporting public access to previously privileged
   resources, such as system-level (i.e., root) access. Given such
   access, it is trivial for anyone to generate a packet with any header
   desired.

   This, coupled with the lack of sufficient address filtering to drop
   such spoofed traffic, can increase the potential for off-path third-
   party spoofing attacks [9][10][47]. Proposed solutions include the
   deployment of existing Internet network and transport security as
   well as modifications to transport protocols that reduce its
   vulnerability to generated attacks [13][15][20][36][46].

   One way to defeat spoofing is to validate the segments of a
   connection, either at the transport level or the network level. TCP
   with MD5 extensions provides this authentication at the transport
   level, and IPsec provides authentication at the network level
   [20][24][27]. In both cases their deployment overhead may be
   prohibitive, e.g., it may not be feasible for public services, such
   as web servers, to be configured with the appropriate certificate
   authorities of large numbers of peers (for IPsec using IKE), or
   shared secrets (for IPsec in shared-secret mode, or TCP/MD5), because

   many clients may need to be configured rapidly without external
   assistance. Services located on public web servers connecting to
   large-scale caches to BGP with larger numbers of peers can fall into
   this category.

   The remainder of this document outlines the recent attack scenario in
   detail and describes and compares a variety of solutions, including
   existing solutions based on TCP/MD5 and IPsec, as well as recently
   proposed solutions, including modifications to TCP's RST processing
   [36], modifications to TCP's timestamp processing [34], and
   modifications to IPsec and TCP/MD5 keying [45]. This document
   focuses on spoofing of TCP segments, although a discussion of related
   spoofing of ICMP packets based on spoofed TCP contents is also
   discussed.

   Note that the description of these attacks is not new; attacks using
   RSTs on BGP have been known since 1998, and were the reason for the
   development of TCP/MD5 [20]. The recent attack scenario was first
   documented by Convery at a NANOG meeting in 2003, but that analysis
   assumed the entire sequence space (2^32 packets) needed to be covered
   for an attack to succeed [10]. Watson's more detailed analysis
   discovered that a single packet anywhere in the current window could
   succeed at an attack [47]. This document adds the observation that
   susceptibility to attack goes as the square of bandwidth, due to the
   coupling between the linear increase in receive window size and
   linear increase in rate of a potential attack, as well as comparing
   the variety of more recent proposals, including modifications to TCP,
   use of IPsec, and use of TCP/MD5 to resist such attacks.

2. Background

   The recent analysis of potential attacks on BGP has again raised the
   issue of TCP's vulnerability to off-path third-party spoofing attacks
   [9][10][47]. A variety of such attacks have been known for several
   years, including sending RSTs, SYNs, and even ACKs in an attempt to
   affect an existing connection or to load down servers. These attacks
   often combine external knowledge (e.g., to indicate the IP addresses
   to attack, the destination port number, and sometimes the ISN) with
   brute-force capabilities enabled by modern computers and network
   bandwidths (e.g., to scan all source ports or an entire window
   space). Overall, such attacks are countered by the use of some form
   of authentication at the network (e.g., IPsec), transport (e.g., SYN
   cookies, TCP/MD5), or other layers. TCP already includes a weak form
   of such authentication in its check of segment sequence numbers
   against the current receiver window. Increases in the bandwidth-
   delay product for certain long connections have sufficiently weakened
   this type of weak authentication to make reliance on it inadvisable.

2.1. **Review of TCP Windows**

   Before proceeding, it is useful to review the terminology and
   components of TCP's windowing algorithm. TCP connections have three
   kinds of windows [1][35]:

   o Send window (SND.WND): the latest send window size.

   o Receive window (RCV.WND): the latest advertised receive window
      size.

   o Congestion window (CWND): the window determined by congestion
      feedback that limits how much of RCV.WND can be in-flight in a
      round trip time.

   For most modern TCP connections, SND.WND and RCV.WND are the size of
   the corresponding send and receive socket buffers, and are
   configurable using socket buffer resizing commands.

   CWND determines how much data can be in transit in a round trip time,
   SND.WND determines how much data the sender is willing to store on
   its side for possible retransmission due to loss, and RCV.WND
   determines the ability of the receiver to accommodate that loss and
   reorder received packets. CWND never grows beyond RCV.WND.

   High bandwidth-delay product networks need CWND to be sufficiently
   large to accommodate as much data as can be in transit in a round
   trip time, otherwise their performance will suffer. As a result, it
   is recommended that users and various automatic programs increase
   RCV.WND to at least the size of bandwidth*delay (the bandwidth-delay
   product) [23][38].

   As the bandwidth-delay product of the network increases, however,
   such increases in the advertised receive window can cause increased
   susceptibility to spoofing attacks, as the remainder of this document
   shows. This assumes, however, that the receive window size (e.g.,
   via increased receive socket buffer configuration) is increased with
   the increased bandwidth-delay product; if not, then connection
   performance will degrade, but susceptibility to spoofing attacks will
   increase only linearly (with the rate at which the attacker can send
   spoofed packets), not as the square of the bandwidth. Note that
   either increase depends on the receive window itself, and is
   independent of the congestion state or amount of data transmitted.

2.2. Recent BGP Attacks Using TCP RSTs

BGP represents a particular vulnerability to spoofing attacks because
it uses TCP connectivity to infer routability, so losing a TCP
connection with a BGP peer can result in the flushing of routes to
that peer [37].

Until six years ago, such connections were assumed difficult to
attack because they were described by a few comparatively obscure
parameters [20]. Most TCP connections are protected by multiple
levels of obfuscation except at the endpoints of the connection:

o Both endpoint addresses are usually not well-known; although
   server addresses are advertised, clients are somewhat anonymous.

o Both port numbers are usually not well-known; the server's usually
   is advertised (representing the service), but the client's is
   typically sufficiently unpredictable to an off-path third-party.

o Valid sequence number space is not well-known.

o Connections are relatively short-lived and valid sequence space
   changes, so any attempt to guess (e.g., by external knowledge or
   brute force) the above information is unlikely to be useful.

BGP represents an exception to the above criteria (though not the
only case). Both endpoints can be well-known, or guessed using hints
from part of an AS path. The destination port is typically fixed to
indicate the BGP service. The source port used by a BGP router is
sometimes fixed and advertised to enable firewall configuration; even
when not fixed, there are only approximately 65,000 valid source
ports which may be exhaustively attacked. Connections are long-
lived, and as noted before some BGP implementations interpret
successive TCP connection failures as routing failures, discarding
the corresponding routing information. In addition, the valid
sequence number space once thought to provide some protection has
been significantly weakened by increasing advertised receive window
sizes.

2.3. TCP RST Vulnerability

TCP has a known vulnerability to third-party spoofed segments. SYN
flooding consumes server resources in half-open connections,
affecting the server's ability to open new connections [4][11]. ACK
spoofing can cause connections to transmit too much data too quickly,
creating network congestion and segment loss, causing connections to
slow to a crawl. In the most recent attacks on BGP, RSTs cause

connections to be dropped. As noted earlier, some BGP
implementations interpret TCP connection termination, or a series of
such failures, as a network failure [37]. This causes routers to
drop the BGP routing information already exchanged, in addition to
inhibiting their ongoing exchanges, thus amplifying the impact of the
attack. The result can affect routing paths throughout the Internet.

The dangerous effects of RSTs on TCP have been known for many years,
even when used by the legitimate endpoints of a connection. TCP RSTs
cause the receiver to drop all connection state; because the source
is not required to maintain a TIME_WAIT state, such a RST can cause
premature reuse of address/port pairs, potentially allowing segments
from a previous connection to contaminate the data of a new
connection, known as TIME_WAIT assassination [8]. In this case,
assassination occurs inadvertently as the result of duplicate
segments from a legitimate source, and can be avoided by blocking RST
processing while in TIME_WAIT. However, assassination can be useful
to deliberately reduce the state held at servers; this requires that
the source of the RSTs go into TIME_WAIT state to avoid such hazards,
and that RSTs are not blocked in the TIME_WAIT state [12].

Firewalls and load balancers, so-called 'middleboxes', sometimes emit
RSTs on behalf of transited connections to optimize server
performance, as noted in RFC 3360 [14]. This is effectively an on-
path RST attack in which the RSTs are sent for benign or beneficial
intent. There are numerous hazards with such use of RSTs, outlined
in that RFC.

2.4. What Changed - the Ever Opening Advertised Receive Window

RSTs represent a hazard to TCP, especially when completely
unvalidated. Fortunately, there are a number of obfuscation
mechanisms that make it difficult for off-path third parties to forge
(spoof) valid RSTs, as noted earlier. We have already shown it is
easy to learn both endpoint addresses and ports for some protocols,
notably BGP. The final obfuscation is the segment sequence number.

TCP segments include a sequence number which enables out-of-order
receiver processing as well as duplicate detection. The sequence
number space is also used to manage congestion, and indicates the
index of the next byte to be transmitted or received. For RSTs, this
is relevant because legitimate RSTs use the next sequence number in
the transmitter window, and the receiver checks that incoming RSTs
have a sequence number in the expected receive window. Such
processing is intended to eliminate duplicate segments (somewhat moot
for RSTs, though), and to drop RSTs which were part of previous
connections.

TCP uses two window mechanisms, a primary mechanism which uses a
space of 32 bits, and a secondary mechanism which scales this window
[23][35]. The valid advertised receive window is a fraction, not to
exceed approximately half, of this space, or ~2 billion (2 * 10^9,
i.e., 2E9 or 2 U.S. billion). Under typical configurations, the
majority of TCP connections open to a very small fraction of this
space, e.g., 10,000-60,000(approximately 5-100 segments). This is
because the advertised receive window typically matches the receive
socket buffer size. It is recommended that this buffer be tuned to
match the needs of the connection, either manually or by automatic
external means [38].

On a low-loss path, the advertised receive window should be
configured to match the path bandwidth-delay product, including
buffering delays (assume 1 packet/hop) [38]. Many paths in the
Internet have end-to-end bandwidths of under 1 Mbps, latencies under
100ms, and are under 15 hops, resulting in fairly small advertised
receive windows as above (under 35,000 bytes). Under these
conditions, and further assuming that the initial sequence number is
suitably (pseudo-randomly) chosen, a valid guessed sequence number
would have odds of 1 in 57,000 of falling within the advertised
receive window. Put differently, a blind (i.e., off-path) attacker
would need to send 57,000 RSTs with suitably spaced sequence number
guesses to successfully reset a connection. At 1 Mbps, 57,000 (40
byte) RSTs would take only 20 seconds to transmit, but this presumes
that both IP addresses and both ports are known. Absent knowledge of
the source port, an off-path spoofer would need to try at least the
entire range of 49152-65535, or 16,384 different ports, resulting in
an attack that would take over 91 hours. Because most TCP
connections are comparatively short-lived, even this moderate
variation in the source port is sufficient for such environments,
although further port randomization may be recommended [29].

Recent use of high bandwidth paths of 10 Gbps and higher results in
bandwidth-delay products over 125 MB - approximately 1/10 of TCP's
overall maximum advertised receive window size (i.e., assuming the
receive socket buffers are increased as much as possible) excluding
scale, assuming the receiver allocates sufficient buffering (as
discussed in Sec. 2). Even under networks that are ten times slower
(1 Gbps), the active advertised receive window covers 1/100th of the
overall window size. At these speeds, it takes only 10-100 packets,
or less than 32 microseconds, to correctly guess a valid sequence
number and kill a connection. A table of corresponding exposure to
various amounts of RSTs is shown below, for various line rates,
assuming the more conventional 100ms latencies (though even 100ms is
large for BGP cases):

BW	BW*delay		RSTs needed	Time needed
10 Gbps	125	MB	35	1 us (microsecond)
1 Gbps	12.5	MB	344	110 us
100 Mbps	1.25	MB	3,436	10 ms (millisecond)
10 Mbps	0.125	MB	34,360	1 second
1 Mbps	0.0125	MB	343,598	2 minutes
100 Kbps	0.00125	MB	3,435,974	3 hours

Figure 1 Time needed to kill a connection

This table demonstrates that the effect of bandwidth on the
vulnerability is squared; for every increase in bandwidth, there is a
linear decrease in the number of sequence number guesses needed, as
well as a linear decrease in the time needed to send a set of
guesses. Notably, as inter-router link bandwidths approach 1 Mbps,
an 'exhaustive' attack becomes practical. Checking that the RST
sequence number is somewhere in the advertised receive window out of
the overall maximum receive window (2^32) is an insufficient
obfuscation.

Note that this table makes a number of assumptions:

1. the overall bandwidth-delay product is relatively fixed

2. traffic losses are negligible (insufficient to affect the
   congestion window over the duration of most of the connection)

3. the advertised receive window is a large fraction of the overall
   maximum receive window size, e.g., because the receive socket
   buffers are set to match a large bandwidth-delay product

4. the attack bandwidth is similar to the end-to-end path bandwidth

Of these assumptions, the last two are more notable. The issue of
receive socket buffers was discussed in Sec. 2. Figure 1 summarized
the time to an successful attack based on large advertised receive
windows, but many current commercial routers have limits of 128KB for
large devices, 32KB for medium, and as little as 4KB for modest ones.
Figure 2 shows the time and bandwidths needed to accomplish an attack
on BGP sessions in the time shown for 100ms latencies; for even
short-range network latencies (10ms), these sessions can be still be
attacked over short timescales (minutes to hours).

```
        BW        BW*delay   RSTs needed     Time needed
     --------------------------------------------------------------
     10 Mbps     0.128 MB        33,555     1 second
      3 Mbps     0.032 MB       134,218    40 seconds
    300 Kbps     0.004 MB     1,073,742     1 hour
```

        Figure 2 Time needed to kill a connection with limited buffers

   The issue of the attack bandwidth is considered reasonable as
   follows:

   1. RSTs are substantially easier to send than data; they can be
      precomputed and they are smaller than data packets (40 bytes)

   2. although susceptible connections use somewhat less ubiquitous
      high-bandwidth paths, the attack may be distributed, at which
      point only the ingress link of the attack is the primary
      limitation

   3. for the purposes of the above table, we assume that the ingress at
      the attack has the same bandwidth as the path, as an approximation

   The previous sections discussed the nature of the recent attacks on
   BGP due to the vulnerability of TCP to RST spoofing attacks, due
   largely to recent increases in the fraction of the TCP advertised
   receive window space in use for a single, long-lived connection.

3. Proposed Solutions and Mitigations

   TCP currently authenticates received RSTs using the address and port
   pair numbers, and checks that the sequence number is inside the valid
   receiver window. The previous section demonstrated how TCP has
   become more vulnerable to RST spoofing attacks due to the increases
   in the receive window size. There are a number of current and
   proposed solutions to this vulnerability, all attempting to provide
   evidence that a received RST is legitimate.

3.1. Transport Layer Solutions

   The transport layer represents the last place that segments can be
   authenticated before they affect connection management. TCP has a
   variety of current and proposed mechanisms to increase the
   authentication of segments, protecting against both off-path and on-
   path third-party spoofing attacks. Other transport protocols, such
   as SCTP and DCCP, also have limited antispoofing mechanisms.

3.1.1. TCP MD5 Authentication

An extension to TCP supporting MD5 authentication was developed in
1998 specifically to authenticate BGP connections (although it can be
used for any TCP connection) [20]. The extension relies on a pre-
shared secret key to authenticate the entire TCP segment, including
the data, TCP header, and TCP pseudo-header (certain fields of the IP
header). All segments are protected, including RSTs, to be accepted
only when their signature matches. This option, although widely
deployed in Internet routers, is considered undeployable for
widespread use because the need for pre-shared keys [3][30]. It
further is considered computationally expensive for either hosts or
routers due to the overhead of MD5 [43][44].

There are also concerns about the use of MD5 due to recent collision-
based attacks [22]. Similar concerns exist for SHA-1, and the IETF
is currently evaluating how these attacks impact the recommendation
for using these hashes, both in TCP/MD5 and in the IPsec suite. For
the purposes of this discussion, the particular algorithm used in
either protocol suite is not the focus, and there is ongoing work to
allow TCP/MD5 to evolve to a more general TCP security option [6].

3.1.2. TCP RST Window Attenuation

A recent proposal extends TCP to further constrain received RST to
match the expected next sequence number [36]. This restores TCP's
resistance to spurious RSTs, effectively limiting the receive window
for RSTs to a single number. As a result, an attacker would need to
send 2^32 different packets to brute-force guess the sequence number
(worst case, average would be half that); this makes TCP's
vulnerability to attack independent of the size of the receive window
(RCV.WND). The extension further modifies the RST receiver to react
to incorrectly-numbered RSTs, by sending a zero-length ACK. If the
RST source is legitimate, upon receipt of an ACK the closed source
would presumably emit a RST with the sequence number matching the
ACK, correctly resetting the intended recipient. This modification
changes TCP's control processing, adding to its complexity and thus
potentially affecting its correctness (in contrast to adding MD5
signatures, which is orthogonal to TCP control processing
altogether). For example, there may be complications between RSTs of
different connections between the same pair of endpoints because RSTs
flush the TIME-WAIT (as mentioned earlier). Further, this proposal
modifies TCP so that under some circumstances a RST causes a reply
(an ACK), in violation of generally accepted practice, if not gentle
recommendation - although this can be omitted, allowing timeouts to
suffice. The advantage to this proposal is that it can be deployed
incrementally and has benefit to the endpoint on which it is

deployed. The other advantage to this proposal is that the window attenuation described here makes the vulnerability to spoofed RST packets independent of the size of the receive window.

A variant of this proposal uses a different value to attenuate the window of viable RSTs. It requires RSTs to carry the initial sequence number rather than the next expected sequence number, i.e., the value negotiated on connection establishment [42][48]. This proposal has the advantage of using an explicitly negotiated value, but at the cost of changing the behavior of an unmodified endpoint to a currently valid RST. It would thus be more difficult, without additional mechanism, to deploy incrementally.

Another variant of this proposal involves increasing TCP's window space, rather than decreasing the valid range for RSTs, i.e., increasing the sequence space from 32 bits to 64 bits. This has the equivalent effect - the ratio of the valid sequence numbers for any segment to the overall sequence number space is significantly reduced. The use of the larger space, as with current schemes to establish weak authentication using initial sequence numbers (ISNs), is contingent on using suitably random values for the ISN. Such randomness adds additional complexity to TCP both in specification and implementation, and provides only very weak authentication. Such a modification is not obviously backward compatible, and would be thus difficult to deploy.

A converse variant of increasing TCP's window space is to decrease the receive window (RCV.WND) explicitly, which would further reduce the effectiveness of spoofed RSTs with random sequence numbers. This alternative may reduce the throughput of the connection, if the advertised receive window is smaller than the bandwidth-delay product of the connection.

3.1.3. TCP Timestamp Authentication

Another way to authenticate TCP segments is via its timestamp option, using the value as a sort of authentication [34]. This requires that the receiver TCP discard segments whose timestamp is outside the accepted window, which is derived from the timestamps of other packets from the same connection. This technique uses an existing TCP option, but also requires modified TCP control processing (with the same caveats) and may be difficult to deploy incrementally without further modifications. Additionally, the timestamp value may be easier to guess because it can be derived predictably, either assuming it represents actual time at the host, or by probing the host using unrelated benign traffic.

3.1.4. Other TCP Cookies

   All of the above techniques are variants of cookies, otherwise
   meaningless data whose value is used to validate the packet. In the
   case of MD5 checksums, the cookie is computed based on a shared
   secret. Note that even a signature can be guessed, and presents a 1
   in 2^(signature length) probability of attack. The primary
   difference is that MD5 signatures are effectively one-time cookies,
   not predictable based on on-path snooping, because they are dependent
   on packet data and thus do not repeat. Window attenuation sequence
   numbers can be guessed by snooping the sequence number of current
   packets of an existing connection, and timestamps can be guessed even
   less directly, either by separate benign connections or by assuming
   reasonably correlation to local time. These variants of cookies are
   similar in spirit to TCP SYN cookies, again patching a vulnerability
   to off-path third-party spoofing attacks based on a (fairly weak,
   excepting MD5) form of authentication. Another form of cookie is the
   source port itself, which can be randomized but provides only 16 bits
   of protection (65,000 combinations), which may be exhaustively
   attacked. This can be combined with destination port randomization
   as well, but that would require a separate coordination mechanism (so
   both parties know which ports to use), which is equivalent to (and as
   infeasible for large-scale deployments as) exchanging a shared secret
   [39].

3.1.5. Other TCP Considerations

   The analysis of the potential for RST spoofing above assumes that the
   advertised receive window is opened to the maximum extent suggested
   by the bandwidth-delay product of the end-to-end path, and that the
   window is opened to an appreciable fraction of the overall sequence
   number space. As noted earlier, for most common cases, connections
   are too brief or over bandwidths too low for such a large window to
   be useful. Expanding TCP's sequence number space is a direct way to
   further avoid such vulnerability, even for long connections over
   emerging bandwidths. If either manual tuning or automatic tuning of
   the advertised receive window (via receive buffer tuning) is not
   provided, this is not an issue (although connection performance will
   suffer) [38].

   It may be sufficient for the endpoint to limit the advertised receive
   window by deliberately leaving it small. If the receive socket
   buffer is limited, e.g., to the ubiquitous default of 64KB, the
   advertised receive window will not be as vulnerable even for very
   long connections over very high bandwidths. The vulnerability will
   grow linearly with the increased network speed, but not as the
   square. The consequence is lower sustained throughput, where only

one window's worth of data per round trip time (RTT) is exchanged.
This will keep the connection open longer; for long-lived connections
with continuous sourced data, this may continue to present an attack
opportunity, albeit a sparse and slow-moving target. For the most
recent case where BGP data is being exchanged between Internet
routers, the data is bursty and the aggregate traffic may be small
(i.e., unlikely to cover a substantial portion of the sequence space,
even if long-lived), so smaller advertised receive windows (via small
receiver buffers) may, in some cases, sufficiently address the
immediate problem. This assumes that the routing tables can be
exchanged quickly enough with bandwidth reduced due to the smaller
buffers, or perhaps that the advertised receive window is opened only
during a large burst exchange (e.g., via some other signal between
the two routers).

3.1.6. Other Transport Protocol Solutions

Segment authentication has been addressed at the transport layer in
other protocols. Both SCTP and DCCP include cookies for connection
establishment and use them to authenticate a variety of other control
messages [28][41]. The inclusion of such mechanism at the transport
protocol, although emerging as standard practice, complicates the
design and implementation of new protocols [32]. As new attacks are
discovered (SYN floods, RSTs, etc.), each protocol must be modified
individually to compensate. A network solution may be more
appropriate and efficient.

It should be noted that RST attacks which rely on brute-force are
relatively easy for intrusion detection software to detect at the TCP
layer. Any connection that receives a large number of invalid -
outside-window - RSTs might have subsequent RSTs blocked, to defeat
such attacks. This would have the side-effect of blocking legitimate
RSTs to that connection, which might then interfere with cleaning up
the transport state between the endpoint peers. This side-effect,
coupled with the increased monitoring load, might render such
solutions undesirable in the general case, but they might usefully be
applied to special cases, e.g., for BGP for routers.

3.2. Network Layer (IP) Solutions

There are two primary variants of network layer solutions to
spoofing: address filtering and IPsec. Address filtering is an
indirect system which relies on other parties to filter packets sent
upstream of an attack, but does not necessarily require participation
of the packet source. IPsec requires cooperation between the
endpoints wanting to avoid attack on their connection, which

   currently involves pre-existing shared knowledge of either a shared
   key or shared certificate authority.

3.2.1. Address filtering

   Address filtering is often proposed as an alternative to protocol
   mechanisms to defeat IP source address spoofing [2][13]. Address
   filtering restricts traffic from downstream sources across transit
   networks based on the IP source address. A kind of filtering already
   occurs at the endpoints of a connection, because attack messages must
   match the socket pair to succeed; again, note that such attacks
   require knowing the entire socket pair, and are unlikely except in
   particular cases. This section discusses filtering based on address
   only, typically done at the borders of an AS.

   It can also restrict core-to-edge paths to reject traffic that should
   have originated further toward the edge. It cannot restrict traffic
   from edges lacking filtering through the core to a particular edge.
   As a result, each border router must perform the appropriate
   filtering for overall protection to result; failure of any border
   router to filter defeats the protection of all participants inside
   the border, and potentially those outside as well. Address filtering
   at the border can protect those inside the border from some kinds of
   spoofing, i.e., connections among those inside a border, because only
   interior addresses should originate inside the border. It cannot,
   however, protect connections including connections outside the border
   except to restrict where the traffic enters from, e.g., if it
   expected from one AS and not another.

   As a result, address filtering is not a local solution that can be
   deployed to protect communicating pairs, but rather relies on a
   distributed infrastructure of trusted gateways filtering forged
   traffic where it enters the network. It is not feasible for local,
   incremental deployment, but may be applicable to connections among
   those inside the protected border in some scenarios. Applying
   filtering can also be useful to reduce the network load of spoofed
   traffic [31].

   A more recent variant of address filtering checks the IP TTL field,
   relying on the TTL set by the other end of the connection [15]. This
   technique has been used to provide filtering for BGP. It assumes the
   connection source TTL is set to 255; packets at the receiver are
   checked for TTL=255, and others are dropped. This restricts traffic
   to one hop upstream of the receiver (i.e., a BGP router), but those
   hops could include other user programs at those nodes (e.g., the BGP
   router's peer) or any traffic those nodes accept via tunnels -
   because tunnels need not decrement TTLs, notably for "bump in the

wire" (BITW) or BITW-equivalent scenarios [33] (see also Sec. 5.1 of
[15] and [16]). TTL filtering works only where all traffic from the
other end of the tunnel is trusted, i.e., where it does not originate
or transit spoofed traffic. The use of TTL rather than link or
network security also assumes an untampered point-to-point link,
where no other traffic can be spoofed onto a link.

This method of filtering works best where traffic originates one hop
away, so that the address filtering is based on the trust of only
directly-connected (tunneled or otherwise) nodes. Like conventional
address filtering, this reduces spoofing traffic in general, but is
not considered a reliable security mechanism because it relies on
distributed filtering (e.g., the fact that upstream nodes do not
terminate tunnels arbitrarily).

3.2.2. IPsec

TCP is susceptible to RSTs, but also to other off-path and on-path
spoofing attacks, including SYN attacks. Other transport protocols,
such as UDP and RTP are equally susceptible. Although emerging
transport protocols attempt to defeat such attacks at the transport
layer, such attacks take advantage of network layer identity
spoofing. The packet is coming from an endpoint who is spoofing
another endpoint, either upstream or somewhere else in the Internet.
IPsec was designed specifically to establish and enforce
authentication of a packet's source and contents, to most directly
and explicitly address this security vulnerability.

The larger problem with IPsec is that of key distribution and use.
IPsec is often cumbersome, and has only recently been supported in
many end-system operating systems. More importantly, it relies on
preshared keys, signed X.509 certificates, or a third-party (e.g.,
Kerberos) public key infrastructure to establish and exchange keying
information (e.g., via IKE). Each of these issues presents
challenges when using IPsec to secure traffic to a well-known server,
whose clients may not support IPsec or may not have registered with a
previously-known certificate authority (CA).

These keying challenges are being addressed in the IETF in ways that
will enable servers secure associations with other parties without
advance coordination [45][46]. This can be especially useful for
publicly-available servers, or for protecting connections to servers
that - for whatever reason - have not, or will not deploy
conventional IPsec certificates (i.e., core Internet BGP routers).

4. **ICMP**

   Just as spoofed TCP packets can terminate a connection, so too can
   spoofed ICMP packets. ICMP can be used to launch a variety of
   attacks on TCP including connection resets, path-MTU attacks, and can
   also be used to attack the host with non-TCP 'ping of death' and
   'smurf attacks', etc. [40]. ICMP thus represents a substantial
   threat to TCP, but this is not the focus of this document, although a
   number of protections are discussed below because some are comparable
   to TCP anti-spoofing techniques. Note also that ICMP attacks on TCP
   assume that the socket pair is known by the attacker, which is
   unlikely except for a subset of services between pairs of widely-
   known endpoints.

   TCP headers can be included inside certain ICMP messages [7]. There
   have been recent suggestions to validate the sequence number of TCP
   headers when they occur inside ICMP messages [18]. This sequence
   checking is similar to checks that would occur for conventional data
   packets in TCP, but is being proposed in the spirit of the RST window
   attenuation described in Section 3.1.2.

   Some such checks may be reasonable, especially where they parallel
   the validations already performed by TCP processing, notably where
   they emulate the semantics of such processing. For example, the TCP
   checksum should be validated (if the entire TCP segment is contained
   in the ICMP message) before any fields of the TCP header are
   examined, to avoid reacting to corrupted packets. Similarly, if the
   TCP MD5 option is present, its signature should probably be validated
   before considering the contents of the message. Such validation can
   ensure that the packet was not corrupted prior to the ICMP generation
   (checksum), that the packet was one sent by the source (IPsec or
   TCP/MD5 authenticated), or that the packet was not in the network for
   an excess of 2*MSL (valid sequence number).

   ICMP presents a particular challenge because some messages can reset
   a connection more easily - with less validation - than even some
   spoofed TCP segments. One other proposed alternative is to change
   TCP's reaction to ICMPs after a connection is established; that may
   leave TCP susceptible during connection establishment and modifies
   TCP's reaction to certain valid network events [19]. This considers
   the context-sensitivity of ICMP messages, as does IPsec in some
   tunneled configurations, but the recommendations are ambiguous
   regarding such filtering [27].

   Ultimately, requiring TCP ICMP messages to be 'in window' may be
   insufficient protection, as this document shows for spoofed data.
   ICMP packets can be authenticated when originating at known, trusted

endpoints, such as endpoints of connections or routers in known
domains with pre-existing IPsec associations. Unfortunately, they
also can originate at other places in the network. In addition, some
networks filter all ICMP packets because validation may not be
possible, especially because they can be injected from anywhere in a
network, and so cannot be easily and locally address filtered [27].
As a result, they are not addressed separately in the issues or
security considerations of this document further.

5. Issues

There are a number of existing and proposed solutions addressing the
vulnerability of transport protocols in general (and TCP in specific)
to off-path third-party spoofing attacks. As shown, these operate at
the transport or network layer. Transport solutions require separate
modification of each transport protocol, addressing network identity
spoofing separately in the context of each transport association.
Network solutions require distributed coordination (filtering) or can
be computationally intensive and require pervasive registration of
certificate authorities with every possible endpoint
(authentication). This section explains these observations further.

5.1. Transport Layer (e.g., TCP)

Transport solutions rely on shared cookies to authenticate segments,
including data, transport header, and even pseudo-header (e.g., fixed
portions of the outer IP header in TCP). Because the Internet relies
on stateless network protocols, it makes sense to rely on state
establishment and maintenance available in some transport layers not
only for the connection but for authentication state. Three-way
handshakes and heartbeats can be used to negotiate authentication
state in conjunction with connection parameters, which can be stored
with connection state easily.

As noted earlier, transport layer solutions require separate
modification of all transport protocols to include authentication.
Not all transport protocols support negotiated endpoint state (e.g.,
UDP), and legacy protocols have been notoriously difficult to safely
augment. Not all authentication solutions are created equal either,
and relying on a variety of transport solutions exposes end-systems
to increased potential for incorrectly specified or implemented
solutions. Transport authentication has often been developed piece-
wise, in response to specific attacks, e.g., SYN cookies and RST
window attenuation [4][36].

Transport layer solutions are not only per-protocol, but often per-
connection. This has both advantages and drawbacks. One advantage

   to transport layer solutions is that they can protect the transport
   protocol when lower layers have failed, e.g., due to bugs in
   implementation. TCP already includes a variety of packet validation
   mechanisms to protect in these cases, e.g., checking that RSTs are
   in-window. More strict checks can increase the protections provided,
   e.g., to protect against misaddressed RSTs that end up in-window (via
   TCPsecure) or to protect against connection interruption due to RSTs,
   SYNs, or data injection from misaddressed packets (TCP/MD5) [36].

   Another advantage is that transport layer protections can be more
   specifically limited to a particular connection. Because each
   connection negotiates its state separately, that state can be more
   specifically tied to that connection. This is both an advantage and
   a drawback. It can make it easier to tie security to an individual
   connection, although in practice a shared secret or certificate will
   generally be shared across multiple connections.

   As a drawback, each transport connection needs to negotiate and
   maintain authentication state separately. Some overhead is not
   amortized over multiple connections, e.g., overheads in packet
   exchanges, whereas other overheads are not amortized over different
   transport protocols, e.g., design and implementation complexity -
   both as would be the case in a network layer solution. Because the
   authentication happens later in packet processing than is required,
   additional endpoint resources may be needlessly consumed, e.g., in
   demultiplexing received packets, indexing connection identifiers, and
   continuing to buffer spoofed packets, etc., only to be dropped later
   at the transport layer.

5.2. Network Layer (IP)

   A network layer solution avoids the hazards of multiple transport
   variants, using a single shared endpoint authentication mechanism
   early in receiver packet processing to discard unauthenticated
   packets at the network layer instead. This defeats spoofing entirely
   because spoofing involves masquerading as another endpoint, and
   network layer security validates the endpoint as the source of the
   packets it emits. Such a network level solution protects all
   transport protocols as a result, including both legacy and emerging
   protocols, and reduces the complexity of these protocols as well. A
   shared solution also reduces protocol overhead, and decouples the
   management (and refreshing) of authentication state from that of
   individual transport connections. Finally, a network layer solution
   protects not only the transport layer but the network layer as well,
   e.g., from IGMP, and some kinds of ICMP (Sec. 4), spoofing attacks.

The IETF Proposed Standard protocol for network layer authentication
is IPsec [27]. IPsec specifies the overall architecture, including
header authentication (AH) [25] and encapsulation (ESP) modes [26].
AH authenticates both the IP header and IP data, whereas ESP
authenticates only the IP data (e.g., transport header and payload).
AH is being phased out since ESP is more efficient and the Security
Parameters Index (SPI) includes sufficient information to verify the
IP header anyway [27]. These two modes describe the security applied
to individual packets within the IPsec system; key exchange and
management is performed either out-of-band (via pre-shared keys) or
by an automated key exchange protocol IKE [24].

IPsec already provides authentication of an IP header and its data
contents sufficient to defeat both on-path and off-path third-party
spoofing attacks. IKE can configure authentication between two
endpoints on a per-endpoint, per-protocol, or per-connection basis,
as desired. IKE also can perform automatic periodic re-keying,
further defeating crypto-analysis based on snooping (clandestine data
collection). The use of IPsec is already commonly strongly
recommended for protected infrastructure.

Existing IPsec is not appropriate for many deployments. It is
computationally intensive both in key management and individual
packet authentication [43]. This computational overhead can be
prohibitive, and so often requires additional hardware, especially in
commercial routers. As importantly, IKE is not anonymous; keys can
be exchanged between parties only if they trust each others' X.509
certificates, trust some other third-party to help with key
generation (e.g., Kerberos), or pre-share a key. These certificates
provide identification (the other party knows who you are) only where
the certificates themselves are signed by certificate authorities
(CAs) that both parties already trust. To a large extent, the CAs
themselves are the pre-shared keys which help IKE establish security
association keys, which are then used in the authentication
algorithms.

Alternative mechanisms are under development to address this
limitation, to allow publicly-accessible servers to secure
connections to clients not known in advance, or to allow unilateral
relaxation of identity validation so that the remaining protections
of IPsec can be made available [45][46]. In particular, these
mechanisms can prevent a client (but without knowing who that client
is) from being affected by spoofing from other clients, even when the
attackers are on the same communications path.

IPsec, although widely available both in commercial routers and
commodity end-systems, is not often used except between parties that

already have a preexisting relationship (employee/employer, between
two ISPs, etc.). Servers to anonymous clients (e.g., customer/
business) or more open services (e.g., BGP, where routers may have
large numbers of peers) are unmanageable, due to the breadth and flux
of CAs. New endpoints cannot establish IPsec associations with such
servers unless their own certificate is signed by a CA already
trusted by the server. Different servers - even within the same
overall system (e.g., BGP) - often cannot or will not trust
overlapping subsets of CAs in general.

5.3. Application Layer

There are a number of application layer authentication mechanisms,
often implicit within end-to-end encryption. Application-layer
security (e.g., TLS, SSH, or MD5 checksums within a BGP stream)
provides the ultimate protection of application data from all
intermediaries, including network routers as well as exposure at
other layers in the end-systems. This is the only way to ultimately
protect the application data.

Application authentication cannot protect either the network or
transport protocols from spoofing attacks, however. Spoofed packets
interfere with network processing or reset transport connections
before the application checks the data. Authentication needs to
winnow these packets and drop them before they interfere at these
lower layers.

An alternate application layer solution would involve resilience to
reset connections. If the application can recover from such
connection interruptions, then such attacks have less impact.
Unfortunately, attackers still affect the application, e.g., in the
cost of restarting connections, delays until connections are
restarted, or increased connection establishment messages on the
network. Some applications - notably BGP - even interpret TCP
connection reliability as an indicator of route path stability, which
is why attacks on BGP have such substantial consequences.

5.4. Link Layer

Link layer security operates separately on each hop of an Internet.
Such security can be critical in protecting link resources, such as
bandwidth and link management protocols. Protection at this layer
cannot suffice for network or transport layers, because it cannot
authenticate the endpoint source of a packet. Link authentication
ensures only the source of the current link hop where it is examined.

5.5. Issues Discussion

   The issues raised in this section suggest that there are challenges
   with all solutions to transport protection from spoofing attacks.
   This raises the potential need for alternate security levels. While
   it is already widely recognized that security needs to occur
   simultaneously at many protocol layers, there also may be utility in
   supporting a variety of strengths at a single layer. For example,
   IPsec already supports a variety of algorithms (MD5, SHA1, etc., for
   authentication), but always assumes that:

   1. the entire body of the packet is secured

   2. security associations are established only where identity is
      authenticated by a know certificate authority or other pre-shared
      key

   3. both on-path and off-path third-party spoofing attacks must be
      defeated

   These assumptions are prohibitive, especially in many cases of
   spoofing attacks. For spoofing, the primary issue is whether packets
   are coming from the same party the server can reach. Only the IP
   header is fundamentally in question, so securing the entire packet
   (1) is computational overkill. It is sufficient to authenticate the
   other party as "a party you have exchanged packets with", rather than
   establishing their trusted identity ("Bill" vs. "Bob") as in (2).
   Finally, many cookie systems use clear-text (unencrypted), fixed
   cookie values, providing reasonable (1 in 2^{cookie-size}) protection
   against off-path third-party spoof attacks, but not addressing on-
   path attacks at all. Such potential solutions are discussed in the
   BTNS documents [5][45][46]. Note also that NULL Encryption in IPsec
   applies a variant of this cookie, where the SPI is the cookie, and no
   further encryption is applied [17].

6. Security Considerations

   This entire document focuses on increasing the security of transport
   protocols and their resistance to spoofing attacks. Security is
   addressed throughout.

   This document describes a number of techniques for defeating spoofing
   attacks. Those relying on clear-text cookies, either explicit or
   implicit (e.g., window sequence attenuation) do not protect from on-
   path spoofing attacks, since valid values can be learned from prior
   traffic. Those relying on true authentication algorithms are
   stronger, protecting even from on-path attacks, because the

authentication hash in a single packet approaches the behavior of
"one time" cookies.

The security of various levels of the protocol stack is addressed.
Spoofing attacks are fundamentally identity masquerading, so we
believe the most appropriate solutions defeat these at the network
layer, where end-to-end identity lies. Some transport protocols
subsume endpoint identity information from the network layer (e.g.,
TCP pseudo-headers), whereas others establish per-connection identity
based on exchanged nonces (e.g., SCTP). It is reasonable, if not
recommended, to address security at all layers of the protocol stack.

Note that NATs and other middleboxes complicate the design and
deployment of techniques to defeat spoofing attacks. Devices such as
these, that modify IP and/or TCP headers in-transit, generate traffic
equivalent to a spoofing attack, and thus should be inhibited by
antispoofing mechanisms. Details of these middlebox-related problems
are out of scope for this document, but issues thereof are addressed
in RFCs and emerging Internet Drafts that discuss the interactions
between such devices and the Internet architecture, e.g., [21].
Fortunately, many of the most critical TCP-based connections, in
particular supporting routing protocols like BGP, do not traverse
such middleboxes, and are not affected by this limitation.

7. IANA Considerations

There are no IANA considerations in this document.

This section should be removed by the RFC-Editor upon publication as
an RFC.

8. Conclusions

This document describes the details of the recent BGP spoofing
attacks involving spurious RSTs which could be used to shutdown TCP
connections. It summarizes and discusses a variety of current and
proposed solutions at various protocol layers.

9. Acknowledgments

This document was inspired by discussions in the TCPM WG
<http://www.ietf.org/html.charters/tcpm-charter.html> about the
recent spoofed RST attacks on BGP routers, including R. Stewart's
draft (whose author list has since evolved) [36][42]. The analysis
of the attack issues, alternate solutions, and the anonymous security
proposed solutions were the result of discussions on that list as
well as with USC/ISI's T. Faber, A. Falk, G. Finn, and Y. Wang. R.

Atkinson suggested the UDP variant of TCP/MD5, P. Goyette suggested using the ISN to seed TCP/MD5, and L. Wood suggested using the ISN to validate RSTs. Other improvements are due to the input of various members of the IETF's TCPM WG, notably detailed feedback from F. Gont, P. Savola, and A. Hoenes.

This document was prepared using 2-Word-v2.0.template.dot.

10. References

10.1. Normative References

None.

10.2. Informative References

[1]   Allman, M., V. Paxson, and W. Stephens, "TCP Congestion Control", RFC 2581 (Proposed Standard), Apr. 1999.

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

[3]   Bellovin, S., and A. Zinin, "Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification", RFC 4278 (Informational), Jan. 2006.

[4]   Bernstein, D., "SYN cookies", http://cr.yp.to/syncookies.html, 1997.

[5]   Better Than Nothing Security [BTNS] WG web pages, http://www.postel.org/anonsec

[6]   Bonica, R., B. Weis, S. Viswanathan, A. Lange, and O. Wheeler, "Authentication for TCP-based Routing and Management Protocols", draft-bonica-tcp-auth-06 (work in progress), Feb. 2007.

[7]   Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC 1122 / STD 3, Oct. 1989.

[8]   Braden, R., "TIME-WAIT Assassination Hazards in TCP", RFC 1337 (Informational), May 1992.

[9]   CERT alert: "Technical Cyber Security Alert TA04-111A: Vulnerabilities in TCP", http://www.us-cert.gov/cas/techalerts/TA04-111A.html, April 20 2004.

[10]   Convery, S., and M. Franz, "BGP Vulnerability Testing:
       Separating Fact from FUD", 2003,
       http://www.nanog.org/mtg-0306/pdf/franz.pdf

[11]   Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations",
       draft-ietf-tcpm-syn-flood-01.txt (work in progress), Dec. 2006.

[12]   Faber, T., J. Touch, and W. Yue, "The TIME-WAIT state in TCP
       and Its Effect on Busy Servers", Proc. Infocom 1999, pp. 1573-
       1583, Mar. 1999.

[13]   Ferguson, P., and D. Senie, "Network Ingress Filtering:
       Defeating Denial of Service Attacks which employ IP Address
       Spoofing", RFC 2827 / BCP 38, May 2000.

[14]   Floyd, S., "Inappropriate TCP Resets Considered Harmful", BCP
       60, RFC 3360 / BCP 60, Aug. 2002.

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

[16]   Gill, V., J. Heasley, D. Meyer, P. Savola, Ed., and C.
       Pignataro, "The Generalized TTL Security Mechanism (GTSM)"
       draft-ietf-rtgwg-rfc3682bis-09.txt (work in progress), Jan.
       2007.

[17]   Glenn, R., and S. Kent, "The NULL Encryption Algorithm and Its
       Use With IPsec", RFC 2410 (Proposed Standard), Nov. 1998.

[18]   Gont, F., "ICMP attacks against TCP", draft-ietf-tcpm-icmp-
       attacks-01.txt (work in progress), Oct. 2006.

[19]   Gont, F., "TCP's Reaction to Soft Errors", draft-ietf-tcpm-tcp-
       soft-errors-03.txt (work in progress), Jan. 2007.

[20]   Heffernan, A., "Protection of BGP Sessions via the TCP MD5
       Signature Option", RFC 2385 (Proposed Standard), Aug. 1998.

[21]   Holdrege, M., and P. Srisuresh, "Protocol Complications with
       the IP Network Address Translator", RFC 3027 (Informational),
       Jan. 2001.

[22]   Housley, R., Post to IETF Discussion mailing list regarding his
       IETF 64 Security Area presentation,
       ID=7.0.0.10.2.20051124135914.00f50558@vigilsec.com, Nov. 24,
       2005, http://www1.ietf.org/mail-
       archive/ietf/Current/maillist.html

[23] Jacobson, V., B. Braden, and D. Borman, "TCP Extensions for
      High Performance", RFC 1323 (Proposed Standard), May 1992.

[24] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306
      (Proposed Standard), Dec. 2005.

[25] Kent, S., "IP Authentication Header", RFC 4302 (Proposed
      Standard), Dec. 2005.

[26] Kent, S, "IP Encapsulating Security Payload (ESP)", RFC 4303
      (Proposed Standard), Dec. 2005.

[27] Kent, S., and K. Seo, "Security Architecture for the Internet
      Protocol", RFC 4301 (Proposed Standard), Dec. 2005.

[28] Kohler, E., M. Handley, and S. Floyd, "Datagram Congestion
      Control Protocol (DCCP)", RFC 4340 (Proposed Standard), Dec.
      2005.

[29] Larsen, M., and F. Gont, "Port Randomization", draft-larsen-
      tsvwg-port-randomization-01.txt (work in progress), Feb. 2007.

[30] Leech, M., "Key Management Considerations for the TCP MD5
      Signature Option", RFC 3562 (Informational), July 2003.

[31] Moore, D., G. Voelker, and S. Savage, "Inferring Internet
      Denial-of-Service Activity", Proc. Usenix Security Symposium,
      Aug. 2001.

[32] O'Malley, S., and L. Peterson, "TCP Extensions Considered
      Harmful", RFC 1263 (Informational), October 1991.

[33] Perkins, C., "IP Encapsulation within IP", RFC 2003 (Proposed
      Standard), Oct. 1996.

[34] Poon, K., "Use of TCP timestamp option to defend against blind
      spoofing attack", draft-poon-tcp-tstamp-mod-01.txt (expired
      work in progress), Oct. 2004.

[35] Postel, J., "Transmission Control Protocol", RFC 793 / STD 7,
      Sep. 1981.

[36] Ramaiah, A., R. Stewart, and M. Dalal, "Improving TCP's
      Robustness to Blind In-Window Attacks", draft-ietf-tcpm-
      tcpsecure-06.txt (work in progress), Nov. 2006.

[37] Rekhter, Y., T. Li, and S. Hares (eds.), "A Border Gateway
      Protocol 4 (BGP-4)", RFC 4271 (Draft Standard), Jan. 2006.

[38] Semke, J., J. Mahdavi, and M. Mathis, "Automatic TCP Buffer
      Tuning", ACM SIGCOMM '98/ Computer Communication Review, volume
      28, number 4, Oct. 1998.

[39] Shepard, T., "Reassign Port Number option for TCP", draft-
      shepard-tcp-reassign-port-number-00.txt (expired work in
      progress), Jul. 2004.

[40] Shirey, R., "Internet Security Glossary, Version 2", draft-
      shirey-secgloss-v2-08.txt (work in progress), Nov. 2006.

[41] Stewart, R., Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer,
      T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson,
      "Stream Control Transmission Protocol", RFC 2960 (Proposed
      Standard), Oct. 2000.

[42] TCPM: IETF TCPM Working Group and mailing list,
      http://www.ietf.org/html.charters/tcpm-charter.html.

[43] Touch, J., "Report on MD5 Performance", RFC 1810
      (Informational), Jun. 1995.

[44] Touch, J., "Performance Analysis of MD5", Proc. Sigcomm 1995,
      pp. 77-86, Mar. 1999.

[45] Touch, J., "ANONsec: Anonymous Security to Defend Against
      Spoofing Attacks", draft-touch-anonsec-00.txt (expired work in
      progress), May 2004.

[46] Touch, J., D. Black, and Y. Wang, "Problem and Applicability
      Statement for Better Than Nothing Security (BTNS)",
      draft-ietf-btns-prob-and-applic-05.txt (work in progress), Feb.
      2007.

[47] Watson, P., "Slipping in the Window: TCP Reset attacks",
      Presentation at 2004 CanSecWest.
      http://www.cansecwest.com/archives.html

[48] Wood, L., Post to TCPM mailing list regarding use of ISN in
      RSTs, ID=Pine.GSO.4.50.0404232249570.5889-
      100000@argos.ee.surrey.ac.uk, Apr. 23, 2004.
      http://www1.ietf.org/mail-
      archive/web/tcpm/current/msg00213.html

Author's Addresses

   Joe Touch
   USC/ISI
   4676 Admiralty Way
   Marina del Rey, CA 90292-6695
   U.S.A.

   Phone: +1 (310) 448-9151
   Fax: +1 (310) 448-9300
   Email: touch@isi.edu
   URI: http://www.isi.edu/touch