Network Working Group
Internet-Draft
Expires: November 3, 2004

J. Touch
USC/ISI

May 5, 2004

ANONsec: Anonymous IPsec to Defend Against Spoofing Attacks
draft-touch-anonsec-00

Status of this Memo

Copyright Notice

Abstract

Recent attacks on core Internet infrastructure indicate an increased vulnerability of TCP connections to spurious resets (RSTs). TCP has always been susceptible to such RST spoof 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, increases in the path bandwidth-delay product of a connection have sufficiently increased the receive window space that off-path third parties can guess a viable RST sequence number. 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. Finally, it proposes an extension to IPsec configuration called ANONsec that intends to efficiently and scalably secure any transport protocol from such off-path third-party spoofing attacks.

Table of Contents

1. Introduction

The Internet infrastructure has recently seen a flurry of attacks on BGP connections between core routers using an attack known for nearly six years [1][2]. These connections, typically using TCP, can be susceptible to off-path (non man-in-the-middle) third-party reset (RST) segments, which terminate the TCP connection. BGP routers react to a terminated TCP connection in various ways, ranging from restarting the connection to deciding that the other router is unreachable and thus flushing the BGP routes. The impact on Internet infrastructure has been substantial, and warrants immediate attention.

TCP, like many other protocols, has been susceptible to off-path third-party attacks. Such attacks rely on the increase of commodity platforms supporting public access to previously privileged resources, such as root-level access. Given such access, it is trivial for anyone to generate a packet with any header desired. This, coupled with the lack of sufficient ingress filtering to drop such spoofed traffic, has resulted in an increase in off-path third-party spoofing attacks. As a result, a number of proposed solutions have been developed in collaboration among the Internet research and commercial router communities. The foremost of these modifies TCP processing to defeat off-path third-party spoofs by further limiting viable sequence numbers in the RST segment.

Such modifications are, at best, temporary patches to the ubiquitous vulnerability to spoofing attacks. The obvious solution to spoofing is to validate the segments of a connection, either at the transport level (which the patch provides, weakly) or the network level. IPsec already intends to provide authentication of a network level packet and its contents, but its deployment overhead can be prohibitive. E.g., it is not feasible for BGP routers to be configured with the appropriate certificate authorities of hundreds of thousands of peers [AUTH - this is from posts on the IPsec mailing list; does anyone have a confirmation?], many of whom need to be configured rapidly without external assistance.

The remainder of this document outlines the attack in detail, and describes a new solution -- ANONsec: Anonymous IPsec. ANONsec allows IPsec association establishment, e.g., via IKE, without needing pre-shared keys (e.g., for new certificate authorities). ANONsec establishes a shared cookie between endpoints, sufficient to assert, "the other end is reachable and I know it from the cookie we share." Such security, together with the use of a range of authentication, including null authentication, provides efficient and scalable protection from off-path (and, depending on configuration, man-in-the-middle) third-party attacks for all protocols from the

network layer up.

2. Background

The recent attacks on BGP have raised the issue of TCP's vulnerability to off-path third-party spoofing attacks [1]. A number 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. Overall, such attacks are countered by the use of some form of authentication at the network (IPsec), transport (SYN cookies), or other layers. TCP already includes a weak form of such authentication in its check of segment sequence numbers. Increases in the bandwidth-delay product for certain long connections has made sufficiently weakened this authentication in recent weeks, rendering it moot.

2.1 Recent BGP Attacks Using TCP RSTs

BGP represents a particular vulnerability to spoofing attacks. Most TCP connections are protected by multiple levels of obfuscation except at the endpoints of the connection:

- Both endpoint addresses are usually not well-known; although server addresses are advertised, clients are somewhat anonymous.
- 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.
- Valid sequence number space is not well-known.
- Connections are relatively short-lived and valid sequence space changes, so any guess of the above information is unlikely to be useful.

BGP (somewhat) uniquely represents an exception to the above criteria. Both endpoints are well-known, notably as 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 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. As importantly and as will be shown below, the valid sequence number space once thought to provide some protection has been rendered useless by increasing congestion window sizes.

2.2 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. 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. This causes routers to drop the BGP routing information already exchanged, in addition to inhibiting their ongoing exchanges. 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 [3]. 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 [4].

Firewalls and load balancers, so-called 'middleboxes', sometimes emit RSTs on behalf of transited connections to optimize server performance [5]. This is a 'man in the middle' RST attack, where the RSTs are sent for benign or beneficial intent. There are numerous hazards with such use of RSTs, outlined in that RFC.

2.3 What Changed -- the Ever Opening Window

RSTs represent a hazard to TCP, especially when completely unchecked. 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 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 [6][7]. The valid receive window is a fraction, not to exceed approximately half, of this space, or ~2,000,000,000. Under typical use, the majority of TCP connections open to a very small fraction of this space, e.g., 10,000-60,000 (approximately 5-100 segments). On a low-loss path, the window should open to around the path bandwidth-delay product, including buffering delays (assume 1 packet/ hop). 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 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. Put differently, a blind (non man-in-the-middle) 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 over 50 minutes to transmit, and, as noted earlier, most current connections are fairly brief by comparison.

Recent use of high bandwidth paths of 10 Gbps and result in bandwidth-delay products over 125 MB - approximately 1/10 of TCP’s overall window size excluding scale, assuming the receiver allocates sufficient buffering (to be discussed later). Even under networks that are ten times slower (1 Gbps), the active receiver window covers 1/100th of the overall window size. At these speeds, it takes only 10-100 packets, or under 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:

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 valid window (bw*delay) out of the overall 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 most of the connection)

3. the receive socket buffers do not limiting the receive window

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 will be addressed later. 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 window space in use for a single, long-lived connection.

3. Proposed solutions

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 centering on increasing the authentication of received RSTs.

3.1 Transport Layer

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 third-party spoofs and man-in-the-middle attacks. SCTP also has mechanisms to authenticate segments.

3.1.1 TCP MD5 Authentication

An extension to TCP supporting MD5 authentication was developed around six years ago specifically to authenticate BGP connections [2]. 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, which are 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. It further is considered computationally expensive for either hosts or routers due to the overhead of MD5 [8][9].

3.1.2 TCP RST Window Attenuation

A recent proposal extends TCP to further constrain received RST to match the expected next sequence number [20]. 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 correctly guess the sequence number. 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. There are a number of concerns with this proposal, including the platitude "think twice before modifying TCP, then don't" [10], notably because this modification adds arcs to the TCP state diagram (in contrast to

adding MD5 signatures, which is orthogonal to the state machine 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 modifies TCP so that under some circumstances a RST causes a reply, in violation of generally accepted practice, if not gentle recommendation. The advantage to this proposal is that it can be deployed incrementally and has benefit to the endpoint on which it is deployed.

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 [11]. 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.

The most obvious other 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 at best. While there are many reasons to increase the TCP sequence number space, we believe authentication is not one of them. Finally, such a modification is not obviously backward compatible, and would be thus difficult to deploy.

3.1.3 TCP Timestamp Authentication

Another way to authenticate TCP segments is to utilize its timestamp option, using the value as a sort of authentication [11]. This requires that the receiver TCP discard values 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 RST processing and may be difficult to deploy incrementally without further modifications. Additionally, the timestamp value may be easier to guess because it is derived from a predictable value.

3.1.4 Other TCP Cookies

All of the above techniques are variants of cookies, otherwise nonsensical data whose value is used to validate the packet. In the case of MD5 checksums, the cookie is computed based on a shared secret. Even a signature can be guessed, and presents a 1 in 2^(cookie length) probability of attack anyway. The primary difference is that MD5 signatures are effectively one-time cookies, not predictable based on man-in-the-middle snooping. Window attenuation sequence numbers can be guessed by snooping the sequence number of current packets, and timestamps can may be guessed even more remotely. 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.

3.1.5 Other TCP Considerations

The analysis of the potential for RST spoofing above assumes that the receive window opens to the maximum extent suggested by the bandwidth-delay product of the end-to-end path, and that the window opens 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 to for such a large window to occur. Expanding TCP’s sequence number space is a direct way to further avoid such vulnerability, even for long connections over emerging bandwidths.

Finally, it is often sufficient for the endpoint to limit the receive window in other ways, notably using ’socket options’. If the receive socket buffer is limited, e.g., to the ubiquitous default of 65KB, the receive window cannot grow to vulnerable sizes even for very long connections over very high bandwidths. The consequence is lower sustained throughput, where only one window’s worth of data per round trip time (RTT) is exchanged. Although this will keep the connection open longer, it also reduces the receive window; 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 is small (i.e., unlikely to cover a substantial portion of the sequence space, even if long-lived), so is difficult to consider

where smaller receive buffers would not sufficiently address the immediate problem.

3.1.6 Other Protocols

Segment authentication has been addressed at the transport layer in other protocols. Both SCTP and DCCP* include cookies for connection establishment and uses them to authenticate a variety of other control messages [12][21]. The inclusion of such mechanism at the transport protocol, although emerging as standard practice, unnecessarily complicates the design and implementation of new protocols. As new attacks are discovered (SYN floods, RSTs, etc.), each protocol must be modified individually to compensate. We consider a network solution more appropriate and efficient.

*[AUTH - DCCP may be removing cookies from the spec for the redundancies discussed above, because the use of cookies at the transport layer primarily supports connection mobility (a design goal of SCTP, but not DCCP) rather than security.

3.2 Network Layer (IP)

TCP is susceptible to RSTs, but also to other spoofing and man-in-the-middle 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, it is clear that such attacks are fundamentally a network layer issue. 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, which most directly, explicitly, and completely addresses this security vulnerability.

The larger problem with IPsec is that of CA 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 signed X.509 certificates to establish and exchange keying information (e.g., via IKE). These present 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).

4. 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. These solutions are not a sufficient long-term strategy to dealing with such attacks, however. Transport solutions require pervasive modification of every transport protocol and address the problem of packet origin identification at the wrong layer. Current network solutions are computationally intensive and require pervasive registration of certificate authorities with every possible endpoint. Neither application-layer nor link-layer solutions suffice to protect either the network or transport layers. This section explains these observations in detail, in advance of presenting a modified network layer solution called ANONsec.

4.1 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 pervasive modification of all transport protocols to include authentication. Not all transport layers support negotiated endpoint state (e.g., UDP), and legacy protocols are notoriously hard to safely augment (e.g., TCP). 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 [13][20].

Transport layer solutions are not only per-protocol, but often per-connection. Each connection needs to negotiate and maintain authentication state separately. Overhead is not amortized over multiple connections - this includes overheads in packet exchanges, design complexity, and implementation complexity. Finally, because the authentication happens later in packet processing than is required, additional endpoint resources are consumed needlessly, e.g., in demultiplexing received packets, indexing connection identifiers, etc.

4.2 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 quickly. Network solutions protect all transport protocols, 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 ICMP, IGMP, etc. spoofing attacks.

The ubiquitous protocol for network layer authentication is IPsec [14][22]. IPsec specifies the overall architecture, including header authentication (AH) [15][23] and encapsulation (ESP) modes [16]. AH authenticates both the IP header and IP data, whereas ESP authenticates only the IP data (e.g., transport header and payload). ESP is somewhat deprecated, as a result, since AH is somewhat of a superset of ESP's capabilities. 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 [17][24].

IPsec already provides authentication of an IP header and its data contents sufficient to defeat both man-in-the-middle 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.

IPsec does not suffice for many uses of BGP, however. It is computationally intensive both in key management and individual packet authentication [8]. As importantly, IKE is not anonymous; keys can be exchanged between parties only if they trust each others' X.509 certificates. 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.

IPsec, although widely available both in commercial routers and commodity end-systems, is not often utilized 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 hundreds of thousands of peers) are unmanageable, due to the breadth and flux of CAs. New endpoints cannot establish IPsec associations with such servers unless their 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.

4.3 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. It is the only way to protect the application data, ultimately.

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 ever gets to check the data. Authentication needs to winnow these packets and drop them before they interfere at these lower layers.

4.4 Shim Transport/Application Layer

Security can also be provided over the transport layer but below the application layer, in a kind of 'shim' protocol, such as SSL or TLS. These protocols provide data protection for a variety of applications over a single, legacy transport protocol, such as SSL/TCP for HTTPS. Unfortunately, like application authentication, they do not protect the transport layer against spoofing attacks.

4.5 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 hop where it is examined.

4.6 Need for Alternate Security Levels

The issues raised in this section suggest the need for alternate security levels. While it is already widely recognized that security

needs to occur simultaneously at many protocol layers, the need for a variety of strengths at a single layer. IPsec already supports a variety of algorithms (MD5, SHA, 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 man-in-the-middle 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 spoofs, but not addressing man-in-the-middle at all.

5. Anonymous IPsec (ANONsec)

We believe it would be useful to allow IPsec where CAs are not pre-shared and to allow a lower-overhead authentication. The former is a kind of anonymous IKE, where a key is negotiated and exchanged but without requiring pre-agreed CAs. The latter can be supported using null-mode or header-only authentication modes. The two in combination can provide a high-performance, easily managed protection from spoofing attacks. Although this proposal, called "Anonymous IPsec (ANONsec)", requires extensions to both IKE and AH processing, we believe it is the most appropriate response to recent attacks.

5.1 Overview of ANONsec

Anonymous IPsec, ANONsec for short, augments IPsec to enable its use by cooperating but otherwise anonymous parties. It relies on no pre-shared information and operates at a variety of performance levels, providing a variety of levels of authentication and/or encryption. To avoid pre-shared information, it uses ANONike, a variant of IKE that avoids the use of CAs. To provide a variety of performance and strength levels, it uses variants of existing IPsec authentication modes: full (fullANON, equivalent to current AH), headerANON (header only AH), and nullANON (no authentication modes).

5.1.1 Anonymous IKE (ANONike)

IKE establishes a shared key between Internet endpoints, based on X.509 certificates and shared agreement on certificate authorities (CAs) who have signed those certificates.* We propose a new mode of IPsec (details to be provided in a separate document if recommended) which relaxes this requirement, called "Anonymous IKE" or ANONike for short. ANONike allows endpoints to specify a certificate or to leave the field blank. Upon receipt of an ANONike message, an endpoint decides whether to reply (e.g., if ANONike is enabled). If permitted, the remainder of IKE proceeds as currently specified, except that the value "0" (or a sufficiently padded nonce, or other fixed value, TBD) is used in place of the certificate throughout IKE processing.

*[AUTH - It appears this requirement is because IKE is based on ISAKMP, which itself requires CAs [18] - if this is not required, or if IKEv2 [24] relaxes this constraint, we would be glad to utilize that and remove this portion of the proposal]

5.1.2 Anonymous Authentication Modes

ANONsec uses three authentication modes together with ANONike. FullANON is just existing AH, which authenticates the fixed portions of the IP header as well as the entire IP payload. HeaderANON

protects only the IP (and possibly TCP) header. Both fullANON and headerANON protect from all third-party spoofing attacks; nullANON, which uses a fixed cookie rather than a hash algorithm, protects only against off-path third-party attacks, but has very low computational cost.

5.1.2.1 Full Authentication (fullANON)

Full anonymous authentication (fullANON) uses ANONike together with conventional AH processing. In this case, the fixed portions of the IP header together with the entire IP payload is authenticated using existing algorithms, e.g., MD5, SHA, etc. FullANON provides full protection of both all headers and data against all third-party spoofing, as well as tampering. Unfortunately, this alternative is as computationally expensive as other forms of authentication, e.g., TCP MD5, since similar algorithms are used over the bulk of the packet [8][9]. FullANON is already supported in IPsec, i.e., it is just AH in default mode, assuming ANONike configures the association.

5.1.2.2 Header-only Authentication (headerANON)

Header-only authentication (headerANON) uses ANONike together with truncated AH processing, where only the fixed portions of the IP header (possibly with some short prefix of the payload, for padding and to avoid small-block hash problems). This mode protects the IP header from all third-party spoofing and tampering. It does not protect the IP payload (outside that optionally used to pad the hash) from tampering. When using a short prefix of the payload, headerANON also usually protects the TCP header as well (when the IP header + TCP header is less than the hash block size, e.g., 64 bytes in MD5), which completely protects against RST attacks. HeaderANON does not protect from man-in-the-middle replay of signed headers with alternate payloads. For packets substantially larger than the hash block size (again, 64 bytes for MD5), headerANON can represent a significant computational savings over full-packet authentication.

HeaderANON would require an extension to AH to indicate that only the fixed portions of the IP header or a fixed length beginning at that header be authenticated. This extension would constitute a new mode for AH, to be negotiated during security association establishment.

5.1.2.3 Null Authentication (nullANON)

Null authentication ignores the context of individual packets completely when computing the authentication signature. The signature may be fixed or vary (if this is even possible). The signature is effectively a cookie; it may even be the case that the SPI itself is a sufficient cookie for this purpose, though we do not recommend that

variant (SPI values are not chosen pseudo-randomly). The cookie may be based on the nonces exchanged via ANONike. The benefit of nullANON is performance; only header modification is required, avoiding all computational overhead. NullAnon protects only from off-path third-party spoofing.

NullANON does not require an extension AH, which already supports null-mode [19]. This mode is currently restricted to use in combination with encryption, i.e., null authentication with non-null encryption. NullANON describes the need for null authentication alone to support IPsec cookies to protect against off-path third-party spoofing attacks, which are not addressed in RFC2401 [14].

*[AUTH- is that the same as with null encryption, or just null auth alone?]

5.2 Benefits of ANONsec

ANONsec provides a way to establish security associations using IKE without requiring endpoints to agree in advance on CAs. Its various authentication modes enable a spectrum of protection that trades various levels of vulnerability for performance and computational overhead. All variants of ANONsec require IPsec keying and key indexing, which itself can be computationally intensive. We believe such keying is not substantially harder than connection demultiplexing (e.g., based on ports), and since a single IKE session can secure all transport connections between two endpoints, there should be much fewer SAs than connection blocks.

ANONsec attempts to make IPsec sufficiently attractive, both computationally and managerially, to operators of ubiquitous Internet services. It provides sufficient protection against spoofing attacks at the appropriate level where masquerading occurs, and can be accomplished with very minor and orthogonal modifications to the IPsec protocol suite. While we appreciate that there is substantial (and understandable) inertia in IPsec, it has less inertia than TCP and we believe these modifications involve less of a change in protocol operation and are a better long-term solution to the recent spoofing attacks.

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 man-in-the-middle spoofing attacks, since valid values can be learned from prior traffic. Those relying on true authentication algorithms are stronger, protecting even from man-in-the-middle, because the authentication hash in a single packet approaches the behavior of "one time" cookies.

Security at various levels of the protocol stack are 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), while 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. We believe that the current attacks are most directly addressed at the network layer, thus the focus of the solutions in this document.

This document presents a security solution which weakens overall security compared to existing solutions. We note, however, that a stronger solution which is not deployed, due to its complexity or performance, is equivalent to no security at all. We believe providing a more easily deployed anonymous variant of IKE together with authentication that can be adjusted to trade strength for performance constitutes an overall benefit toward the increased deployment of security solutions.

7. Conclusions

This document describes the details of the recent BGP spoofing attacks involving spurious RSTs used to shutdown TCP connections. It summarizes and discusses a variety of current and proposed solutions at various protocol layers. Finally, it presents ANONsec, a modification of IPsec that enables anonymous IPsec using automatic (IKE) key exchange and using a variety of levels of authentication. We believe ANONsec is the most appropriate and direct response to the recent issue of spoofing attacks, both for the current case and for the long term benefit of the Internet as a whole.

8. Acknowledgments

This document was inspired by discussions on the <http://www.ietf.org/html.charters/tcpm-charter.html> about the recent spoofed RST attacks on BGP routers, including R. Stewart's draft [11][20]. The analysis of the attack issues, alternate solutions, and ANONsec proposed solution were the result of discussions on that list as well as with USC/ISI's T. Faber, A. Falk, G. Finn, and Y. Wang.

9. References

9.1 Normative References

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

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

[3] Braden, B., "TIME-WAIT Assassination Hazards in TCP", RFC 1337, May 1992.

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

[5] Floyd, S., "Inappropriate TCP Resets Considered Harmful", BCP 60, RFC 3360, August 2002.

[6] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[7] Jacobson, V., Braden, B. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[8] Touch, J., "Report on MD5 Performance", RFC 1810, June 1995.

[9] Touch, J., "Performance Analysis of MD5", Proc. Sigcomm 1995 77-86., March 1999.

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

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

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

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

[14] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[15] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[16] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[17] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[18] Maughan, D., Schneider, M. and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[19] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.

9.2 Informative References

[20] Stewart, R., "Transmission Control Protocol security considerations", draft-ietf-tcpm-tcpsecure-00 (work in progress), April 2004.

[21] Kohler, E., "Datagram Congestion Control Protocol (DCCP)", draft-ietf-dccp-spec-06 (work in progress), February 2004.

[22] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", draft-ietf-ipsec-rfc2401bis-02 (work in progress), April 2004.

[23] Kent, S., "IP Authentication Header", draft-ietf-ipsec-rfc2402bis-07 (work in progress), March 2004.

[24] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", draft-ietf-ipsec-ikev2-13 (work in progress), March 2004.

Author's Address

Joe Touch
USC/Information Sciences Institute
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

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.