        Protection of BGP Sessions via the TCP MD5 Signature Option

Status of this Memo

Copyright Notice

IESG Note

   This document describes currrent existing practice for securing BGP
   against certain simple attacks. It is understood to have security
   weaknesses against concerted attacks.

Abstract

   This memo describes a TCP extension to enhance security for BGP. It
   defines a new TCP option for carrying an MD5 [RFC1321] digest in a
   TCP segment. This digest acts like a signature for that segment,
   incorporating information known only to the connection end points.
   Since BGP uses TCP as its transport, using this option in the way
   described in this paper significantly reduces the danger from certain
   security attacks on BGP.

1.0 Introduction

   The primary motivation for this option is to allow BGP to protect
   itself against the introduction of spoofed TCP segments into the
   connection stream. Of particular concern are TCP resets.

   To spoof a connection using the scheme described in this paper, an
   attacker would not only have to guess TCP sequence numbers, but would
   also have had to obtain the password included in the MD5 digest.
   This password never appears in the connection stream, and the actual
   form of the password is up to the application. It could even change

during the lifetime of a particular connection so long as this change
was synchronized on both ends (although retransmission can become
problematical in some TCP implementations with changing passwords).

Finally, there is no negotiation for the use of this option in a
connection, rather it is purely a matter of site policy whether or
not its connections use the option.

2.0 Proposal

Every segment sent on a TCP connection to be protected against
spoofing will contain the 16-byte MD5 digest produced by applying the
MD5 algorithm to these items in the following order:

    1. the TCP pseudo-header (in the order: source IP address,
       destination IP address, zero-padded protocol number, and
       segment length)
    2. the TCP header, excluding options, and assuming a checksum of
       zero
    3. the TCP segment data (if any)
    4. an independently-specified key or password, known to both TCPs
       and presumably connection-specific

The header and pseudo-header are in network byte order. The nature
of the key is deliberately left unspecified, but it must be known by
both ends of the connection. A particular TCP implementation will
determine what the application may specify as the key.

Upon receiving a signed segment, the receiver must validate it by
calculating its own digest from the same data (using its own key) and
comparing the two digest. A failing comparison must result in the
segment being dropped and must not produce any response back to the
sender. Logging the failure is probably advisable.

Unlike other TCP extensions (e.g., the Window Scale option
[RFC1323]), the absence of the option in the SYN,ACK segment must not
cause the sender to disable its sending of signatures. This
negotiation is typically done to prevent some TCP implementations
from misbehaving upon receiving options in non-SYN segments. This is
not a problem for this option, since the SYN,ACK sent during
connection negotiation will not be signed and will thus be ignored.
The connection will never be made, and non-SYN segments with options
will never be sent. More importantly, the sending of signatures must
be under the complete control of the application, not at the mercy of
the remote host not understanding the option.

3.0 Syntax

   The proposed option has the following format:

```
            +---------+---------+-------------------+
            | Kind=19 |Length=18|   MD5 digest...   |
            +---------+---------+-------------------+
            |                                       |
            +---------------------------------------+
            |                                       |
            +---------------------------------------+
            |                                       |
            +------------------+-------------------+
            |                  |
            +------------------+
```

   The MD5 digest is always 16 bytes in length, and the option would
   appear in every segment of a connection.

4.0 Some Implications

4.1 Connectionless Resets

   A connectionless reset will be ignored by the receiver of the reset,
   since the originator of that reset does not know the key, and so
   cannot generate the proper signature for the segment. This means,
   for example, that connection attempts by a TCP which is generating
   signatures to a port with no listener will time out instead of being
   refused. Similarly, resets generated by a TCP in response to
   segments sent on a stale connection will also be ignored.
   Operationally this can be a problem since resets help BGP recover
   quickly from peer crashes.

4.2 Performance

   The performance hit in calculating digests may inhibit the use of
   this option. Some measurements of a sample implementation showed
   that on a 100 MHz R4600, generating a signature for simple ACK
   segment took an average of 0.0268 ms, while generating a signature
   for a data segment carrying 4096 bytes of data took 0.8776 ms on
   average. These times would be applied to both the input and output
   paths, with the input path also bearing the cost of a 16-byte
   compare.

4.3 TCP Header Size

   As with other options that are added to every segment, the size of
   the MD5 option must be factored into the MSS offered to the other
   side during connection negotiation. Specifically, the size of the
   header to subtract from the MTU (whether it is the MTU of the
   outgoing interface or IP's minimal MTU of 576 bytes) is now at least
   18 bytes larger.

   The total header size is also an issue. The TCP header specifies
   where segment data starts with a 4-bit field which gives the total
   size of the header (including options) in 32-byte words. This means
   that the total size of the header plus option must be less than or
   equal to 60 bytes -- this leaves 40 bytes for options.

   As a concrete example, 4.4BSD defaults to sending window-scaling and
   timestamp information for connections it initiates. The most loaded
   segment will be the initial SYN packet to start the connection. With
   MD5 signatures, the SYN packet will contain the following:

      -- 4 bytes MSS option
      -- 4 bytes window scale option (3 bytes padded to 4 in 4.4BSD)
      -- 12 bytes for timestamp (4.4BSD pads the option as recommended
         in RFC 1323 Appendix A)
      -- 18 bytes for MD5 digest
      -- 2 bytes for end-of-option-list, to pad to a 32-bit boundary.

      This sums to 40 bytes, which just makes it.

4.4 MD5 as a Hashing Algorithm

   Since this memo was first issued (under a different title), the MD5
   algorithm has been found to be vulnerable to collision search attacks
   [Dobb], and is considered by some to be insufficiently strong for
   this type of application.

   This memo still specifies the MD5 algorithm, however, since the
   option has already been deployed operationally, and there was no
   "algorithm type" field defined to allow an upgrade using the same
   option number. The original document did not specify a type field
   since this would require at least one more byte, and it was felt at
   the time that taking 19 bytes for the complete option (which would
   probably be padded to 20 bytes in TCP implementations) would be too
   much of a waste of the already limited option space.

This does not prevent the deployment of another similar option which
uses another hashing algorithm (like SHA-1). Also, if most
implementations pad the 18 byte option as defined to 20 bytes anyway,
it would be just as well to define a new option which contains an
algorithm type field.

This would need to be addressed in another document, however.

4.5 Key configuration

It should be noted that the key configuration mechanism of routers
may restrict the possible keys that may be used between peers. It is
strongly recommended that an implementation be able to support at
minimum a key composed of a string of printable ASCII of 80 bytes or
less, as this is current practice.

5.0 Security Considerations

This document defines a weak but currently practiced security
mechanism for BGP. It is anticipated that future work will provide
different stronger mechanisms for dealing with these issues.

6.0 References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm," RFC 1321,
          April 1992.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions
          for High Performance", RFC 1323, May 1992.

[Dobb] H. Dobbertin, "The Status of MD5 After a Recent Attack", RSA
          Labs' CryptoBytes, Vol. 2 No. 2, Summer 1996.
          http://www.rsa.com/rsalabs/pubs/cryptobytes.html

Author's Address

Andy Heffernan
cisco Systems
170 West Tasman Drive
San Jose, CA 95134 USA

Phone: +1 408 526-8115
EMail: ahh@cisco.com