Network Working Group
Request for Comments: 4808
Category: Informational

S. Bellovin
Columbia University
March 2007

Key Change Strategies for TCP-MD5

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

The TCP-MD5 option is most commonly used to secure BGP sessions between routers. However, changing the long-term key is difficult, since the change needs to be synchronized between different organizations. We describe single-ended strategies that will permit (mostly) unsynchronized key changes.

1. Introduction

The TCP-MD5 option [RFC2385] is most commonly used to secure BGP sessions between routers. However, changing the long-term key is difficult, since the change needs to be synchronized between different organizations. Worse yet, if the keys are out of sync, it may break the connection between the two routers, rendering repair attempts difficult.

The proper solution involves some sort of key management protocol. Apart from the complexity of such things, RFC 2385 was not written with key changes in mind. In particular, there is no KeyID field in the option, which means that even a key management protocol would run into the same problem.

Fortunately, a heuristic permits key change despite this protocol deficiency. The change can be installed unilaterally at one end of a connection; it is fully compatible with the existing protocol.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. The Algorithm

Separate algorithms are necessary for transmission and reception. Reception is easier; we explain it first.

2.1. Reception

A receiver has a list of valid keys. Each key has a (conceptual) timestamp associated with it. When a segment arrives, each key is tried in turn. The segment is discarded if and only if it cannot be validated by any key in the list.

In principle, there is no need to test keys in any particular order. For performance reasons, though, a simple most-recently-used (MRU) strategy -- try the last valid key first -- should work well. More complex mechanisms, such as examining the TCP sequence number of an arriving segment to see whether it fits in a hole, are almost certainly unnecessary. On the other hand, validating that a received segment is putatively legal, by checking its sequence number against the advertised window, can help avoid denial of service attacks.

The newest key that has successfully validated a segment is marked as the "preferred" key; see below.

Implicit in this scheme is the assumption that older keys will eventually be unneeded and can be removed. Accordingly, implementations SHOULD provide an indication of when a key was last used successfully.

2.2. Transmission

Transmission is more complex, because the sender does not know which keys can be accepted at the far end. Accordingly, the conservative strategy is to delay using any new keys for a considerable amount of time, probably measured in days. This time interval is the amount of asynchronicity the parties wish to permit; it is agreed upon out of band and configured manually.

Some automation is possible, however. If a key has been used successfully to validate an incoming segment, clearly the other side knows it. Accordingly, any key marked as "preferred" by the receiving part of a stack SHOULD be used for transmissions.

A sophisticated implementation could try alternate keys if the TCP retransmission counter gets too high. (This is analogous to dead gateway detection.) In particular, if a key change has just been attempted but such segments are not acknowledged, it is reasonable to fall back to the previous key and issue an alert of some sort. Similarly, an implementation with a new but unused key could occasionally try to use it, much in the way that TCP implementations probe closed windows. Doing this avoids the "silent host" problem discussed in Section 3.1. This should be done at a moderately slow rate.

Note that there is an ambiguity when an acknowledgment is received for a segment transmitted with two different keys. The TCP Timestamp option [RFC1323] can be used for disambiguation.

3. Operations

3.1. Single-Ended Operations

Suppose only one end of the connection has this algorithm implemented. The new key is provisioned on that system, with a start time far in the future -- sufficiently far, in fact, that it will not be used spontaneously. After the key is ready, the other end is notified, out-of-band, that a key change can commence.

At some point, the other end is upgraded. Because it does not have multiple keys available, it will start using the new key immediately for its transmission, and will drop all segments that use the old key. As soon as it tries to transmit, the upgraded side will designate the new key as preferred, and will use it for all of its transmissions. Note specifically that this will include retransmissions of any segments rejected because they used the old key.

There is a problem if the unchanged machine is a "silent host" -- a host that has nothing to say, and hence does not transmit. The best way to avoid this is for an upgraded machine to try a variety of keys in the event of repeated unacknowledged packets, and to probe for new unused keys during silent periods, as discussed in Section 2.2. Alternatively, application-level KeepAlive messages may be used to ensure that neither end of the connection is completely silent. See, for example, Section 4.4 of [RFC4271] or Section 3.5.4 of [RFC3036].

3.2. Double-Ended Operations

Double-ended operations are similar, save that both sides deploy the new key at about the same time. One should be configured to start using the new key at a point where it is reasonably certain that the other side would have it installed, too. Assuming that has in fact happened, the new key will be marked "preferred" on both sides.

3.3. Monitoring

As noted, implementations should monitor when a key was last used for transmission or reception. Any monitoring mechanism can be used; most likely, it will be one or both of a MIB object or objects and the vendor's usual command-line mechanism for displaying data of this type. Regardless, the network operations center should keep track of this. When a new key has been used successfully for both transmission and reception for a reasonable amount of time -- the exact value isn't crucial, but it should probably be longer than twice the maximum segment lifetime -- the old key can be marked for deletion. There is an implicit assumption here that there will not be substantial overlap in the usage period of such keys; monitoring systems should look for any such anomalies, of course.

4. Moving Forward

As implied in Section 1, this is an interim strategy, intended to make TCP-MD5 operationally usable today. We do not suggest or recommend it as a long-term solution. In this section, we make some suggestions about the design of a future TCP authentication option.

The first and most obvious change is to replace keyed MD5 with a stronger MAC [RFC4278]. Today, HMAC-SHA1 [RFC4634] is the preferred choice, though others such as UMAC [RFC4418] should be considered as well.

A new authentication option should contain some form of a Key ID field. Such an option would permit unambiguous identification of which key was used to create the MAC for a given segment, sparing the receiver the need to engage in the sort of heuristics described here. A Key ID is useful with both manual and automatic key management. (Note carefully that we do not prescribe any particular Key ID mechanism here. Rather, we are stating a requirement: there must be a simple, low-cost way to select a particular key, and it must be possible to rekey without tearing down long-lived connections.)

Finally, an automated key management mechanism should be defined. The general reasoning for that is set forth in [RFC4107]; specific issues pertaining to BGP and TCP are given in [RFC3562].

5. Security Considerations

In theory, accepting multiple keys simultaneously makes life easier for an attacker. In practice, if the recommendations in [RFC3562] are followed, this should not be a problem.

New keys must be communicated securely. Specifically, new key messages must be kept confidential and must be properly authenticated.

Having multiple keys makes CPU denial-of-service attacks easier. This suggests that keeping the overlap period reasonably short is a good idea. In addition, the Generalized TTL Security Mechanism [RFC3682], if applicable to the local topology, can help. Note that most of the time, only one key will exist; virtually all of the remaining time there will be only two keys in existence.

6. IANA Considerations

There are no IANA actions required. The TCP-MD5 option number is defined in [RFC2385], and is currently listed by IANA.

7. Acknowledgments

I’d like to thank Ron Bonica, Randy Bush, Ross Callon, Rob Evans, Eric Rescorla, and Sam Weiler for their comments and inspiration.

8. References

8.1. Normative References

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

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

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.

8.2. Informative References

[RFC3036] Andersson, L., Doolan, P., Feldman, N., Fredette, A., and B. Thomas, "LDP Specification", RFC 3036, January 2001.

[RFC3562] Leech, M., "Key Management Considerations for the TCP MD5 Signature Option", RFC 3562, July 2003.

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

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

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

[RFC4278] Bellovin, S. and A. Zinin, "Standards Maturity Variance Regarding the TCP MD5 Signature Option (RFC 2385) and the BGP-4 Specification", RFC 4278, January 2006.

[RFC4418] Krovetz, T., "UMAC: Message Authentication Code using Universal Hashing", RFC 4418, March 2006.

[RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, August 2006.

Author's Address

Steven M. Bellovin
Columbia University
1214 Amsterdam Avenue
MC 0401
New York, NY 10027
US

Phone: +1 212 939 7149
EMail: bellovin@acm.org

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat 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 implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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