Authentication for TCP-based Routing and Management Protocols
draft-bonica-tcp-auth-01

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 March 31, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This memo extends RFC 2385 to support time-based key rollover and multiple hashing algorithms. Operators can use the time-based key rollover feature to in order to periodically update the key that is used to create authentication data for each TCP segment. Operators may also wish to select the hashing algorithm used to create authentication data depending upon the perceived threat level and the computational capabilities of their hardware platforms.
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Conventions Used In This Document</td>
</tr>
<tr>
<td>2. Introduction</td>
</tr>
<tr>
<td>3. Proposal</td>
</tr>
<tr>
<td>4. Syntax</td>
</tr>
<tr>
<td>5. Implications</td>
</tr>
<tr>
<td>5.1. Clock Synchronization</td>
</tr>
<tr>
<td>5.2. Connectionless Resets</td>
</tr>
<tr>
<td>5.3. Performance</td>
</tr>
<tr>
<td>5.4. TCP Header Size</td>
</tr>
<tr>
<td>5.5. Key Configuration</td>
</tr>
<tr>
<td>5.6. Backwards Compatibility</td>
</tr>
<tr>
<td>6. Security Considerations</td>
</tr>
<tr>
<td>7. IANA Considerations</td>
</tr>
<tr>
<td>8. Acknowledgments</td>
</tr>
<tr>
<td>9. Normative References</td>
</tr>
<tr>
<td>Authors’ Addresses</td>
</tr>
<tr>
<td>Intellectual Property and Copyright Statements</td>
</tr>
</tbody>
</table>
1. Conventions Used In This Document

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 [1].

2. Introduction

RFC 2385 [2] proposes a mechanism that secures BGP [3] sessions using MD5 [4] authentication. Specifically, RFC 2385 proposes a TCP MD5 Signature Option that can be appended to each TCP header. The MD5 Signature Option contains a 16-byte MD5 digest that serves as authentication data for the TCP segment. The MD5 digest is calculated over the following fields:

- the TCP pseudo-header
- the TCP header, excluding options, and assuming a checksum of zero
- the TCP segment data (if any)
- an independently-specified key or password, known to both TCPs and presumably connection-specific

To spoof a connection using the scheme described above, 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 determined by the application.

RFC 3562 [5] addresses key management considerations regarding the TCP MD5 Signature Option. Specifically, based upon the strength of the MD5 hashing algorithm, RFC 3562 recommends that keys SHOULD be changed at least every 90 days.

Unfortunately, the strategy described in RFC 2385 permits keys to be changed during the lifetime of a TCP connection only so long as the change is synchronized at both ends. This limitation has proven to be a significant deterrent to the deployment of the TCP MD5 Signature Option for BGP.

This document addresses the above mentioned limitation. It also extends the strategy proposed in RFC 2385 to allow for other hashing algorithms besides MD5.
3. Proposal

This document proposes a new TCP Enhanced Authentication Option that is used as follows.

Operators configure a list of authentication elements for each protected TCP connection. Each authentication element includes the following data items:

- an authentication element identifier (integer [0..255])
- a key
- a hash algorithm
- a start time

Each authentication element in the list must include a unique element identifier and a unique start time.

Whenever TCP generates a segment, it selects an authentication element from the list. The selected element must have a start time that is greater than or equal to the current time. If multiple authentication elements meet that criteria, TCP will select one of them. Specifically, it will select the authentication element that specifies the earliest start time.

TCP then inserts the new option and calculates a message digest. It calculates a message digest by applying the hash algorithm from the selected authentication element to the following items in the order that they are listed:

- the TCP pseudo-header
- the TCP header, including options, but with hash value set to zero for the purpose of computation and assuming a checksum of zero
- the TCP segment data (if any)
- the key specified by the selected authentication element

For IPv4, the pseudo-header is described in RFC 793 [6]. It includes the 32-bit source IP address, the 32-bit destination IP address, the zero-extended protocol number (to form 16 bits), and the 16-bit segment length. Note that this includes use of IPv4 via IPv4-mapped IPv6 addresses, in which case the source and destination IP addresses are from the IPv4 portions of the IPv6 source and destination.
addresses, respectively.

For IPv6, the pseudo-header is described in RFC 2460 [7]. It includes the 128-bit source IPv6 address, the 128-bit destination IPv6 address, the zero-extended next header value (to form 32 bits), and the 32-bit segment length.

For any other network protocol, the pseudo-header is as described in the document that defines how upper-level protocols like TCP compute their checksums.

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.

Having calculated the message digest, TCP updates the new TCP option to include the message digest. TCP then calculates a checksum and forwards the segment to its TCP peer.

The TCP peer is also configured with a list of authentication elements for the connection. Having received a TCP segment, the TCP peer scans its list of authentication elements, searching for an element whose identifier matches that which was specified by the incoming TCP option. If such an authentication element is found, TCP uses the key from that authentication element to calculate a message digest. If the calculated message digest matches the message digest received in the incoming TCP segment, the segment is accepted. Otherwise, TCP declares an authentication failure and discards the datagram. An authentication failure MUST NOT produce any response back to the sender. Routers SHOULD log authentication failures.

Unlike other TCP extensions (e.g., the Window Scale option [8]), the absence of the option in the SYN,ACK segment must not cause the sender to disable its sending of authentication data. 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 authentication data must be under the complete control of the application, not at the mercy of the remote host not understanding the option.

4. Syntax

The proposed TCP Enhanced Authentication Option has the following
format:

```
+---+---+---+---+---+---+---+---+
|   Kind   |   Length   |   Auth ID   |   Reserved   |
+---+---+---+---+---+---+---+---+
               Message Digest
```

Figure 1: Option Syntax

Kind: 8 bits

The Kind field identifies the TCP Enhanced Authentication Option. This value will be assigned by IANA.

Length: 8 bits

The Length field specifies the length of the TCP Enhanced Authentication Option, in octets. This count includes two octets representing the Kind and Length fields.

Auth ID: 8 bits

The Auth ID field identifies the authentication element that was used to generate the message digest.

Reserved: 8 bits

Must be equal to zero.

Message Digest: Variable length

A Message Digest that serves as authentication data for the TCP segment. The length of the Message Digest, and therefore, the length of the entire option, is determined by the hash algorithm.

Table 1 maps hash algorithms to the size of the digests that they produce. Permissible hash algorithms are not restricted to those listed in the table.
5. Implications

5.1. Clock Synchronization

Because the TCP Enhanced Authentication Option includes an authentication element identifier, the strategy described herein is immune from most problems caused by poor clock synchronization. Clocks do not need to be synchronized between the sending and receiving systems. The only requirement is that the authentication element used to generate the hash value on the sending system is also configured on the receiving system.

Receipt of a segment whose authentication data was generated using a stale authentication element does not constitute an error. It may indicate only that clocks are not synchronized between the sending and receiving systems.

5.2. 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 therefore cannot generate the proper authentication data for the segment. This means, for example, that connection attempts by a TCP which is generating authentication data 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 some protocols recover quickly from peer crashes.

5.3. Performance

The performance hit in calculating digests may inhibit the use of this option. Performance will vary depending upon processor type,
hash algorithm, packet size and number of hash calculations per second.

5.4. TCP Header Size

As with other options that are added to every segment, the size of the TCP Enhanced Authentication 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 octets) is now increased by the size of the TCP Enhanced Authentication Option.

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 octets. This leaves 40 octets for options.

As a concrete example, assume that a TCP implementation defaults to sending window-scaling for connections it initiates. The most loaded segment will be the initial SYN packet to start the connection. With a TCP Enhanced Authentication object using SHA1 authentication, the SYN packet will contain the following:

-- 4 octets MSS option
-- 4 octets window scale option (3 octets padded to 4 in this implementation)
-- 24 octets for the TCP Enhanced Authentication Option
-- 2 octets for end-of-option-list, to pad to a 32-bit boundary.

This sums to exactly 34 octets. This leaves only 6 octets for additional TCP options. Some longer options (e.g. Timestamp) would not fit in that space.

5.5. 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 octets or less, as this is current practice.

During the lifetime of a TCP connection, network operators may add or delete any key. However, the network operator must ensure that the
active key is always configured on both TCP endpoints.

Network operators may choose to protect multiple connections with a single list of authentication elements. For example, a network operator may associate every TCP connection supporting iBGP with one authentication element list while associating a unique authentication element list with each TCP connection that supports eBGP.

In the future, a key exchange protocol may be specified to provision the authentication elements described herein.

5.6. Backwards Compatibility

On any particular TCP connection, use of the TCP Enhanced Authentication Option precludes use of the TCP MD5 Signature Option. However, use of the TCP Enhanced Authentication Option on one connection does not preclude the use of the TCP MD5 Signature Option on another connection by the same system.

6. Security Considerations

This document defines a weak but easily deployed security mechanism for TCP-based routing protocols. It is anticipated that future work will provide different stronger mechanisms for dealing with these issues.

7. IANA Considerations

IANA will assign a codepoint for the TCP Enhanced Authentication Option.

8. Acknowledgments

Thanks to Steve Bellovin, Ted Faber, Ross Callon and Ran Atkinson for their comments regarding this draft.

9. Normative References


Bonica & Heffernan Expires March 31, 2006 [Page 9]


Authors’ Addresses

Ronald P. Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, VA 20171
US
Phone: +1 571 203 1704
Email: rbonica@juniper.net

Andy Heffernan
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US
Phone: +1 408 745 2037
Email: ahh@juniper.net
Intellectual Property Statement

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.

Disclaimer of Validity

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 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.

Copyright Statement

Copyright (C) The Internet Society (2005). 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.

Acknowledgment

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