

## Protection of BGP Sessions via the TCP MD5 Signature Option

### Status of this Memo

This document is an Internet Draft. 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.

This document is an Internet-Draft and is subject to all provisions of [Section 10 of RFC2026](#).

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any Internet Draft.

### 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

INTERNET-DRAFT

TCP MD5 Signature Option

March 2002

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
2. the TCP header, excluding options (but with a correct Data Offset field), 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

For IPv4, the pseudo-header is as described in [RFC 793](#), namely the 32-bit source IP address, 32-bit destination IP address, zero-extended protocol number (to form 16 bits), and 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 as described in [RFC 2460](#), namely the 128-bit source IPv6 address, 128-bit destination IPv6 address, zero-extended next header value (to form 32 bits), and 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.

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. (Note that other padding schemes are possible which would reduce the aggregate size.)

#### [4.4](#) Key Configuration

Heffernan

Expires September 1, 2002

[Page 4]

---

INTERNET-DRAFT

TCP MD5 Signature Option

March 2002

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.

#### [5.1](#) MD5 as a hashing algorithm

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 document specifies the MD5 algorithm, however, since the option has already been deployed operationally, and there is no "algorithm type" field defined to allow an upgrade using the same option number.

The original draft 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 draft, however.

## [5.2](#) Signature Coverage

A further weakness exists due to the exclusion of option data from the signature. This decision was made to simplify the protocol definition and implementation, but might possibly leave a connection vulnerable since option data can be rewritten without detection.

## [6.0](#) Changes from [RFC 2385](#)

The previous version of this specification was published as [RFC 2385](#).

The changes in this document are primarily to be more explicit about the data over which the MD5 signature is taken. In particular, [Section 2.0](#) describes the differences between IPv4 and IPv6 and their respective pseudo-header definitions.

Additionally, [section 4.3](#) adds a note about option padding.

The former [section 4.4](#) ("MD5 has a Hashing Algorithm") has moved to the Security Considerations section of the document as [section 5.1](#) and the former [section 4.5](#) ("Key configuration") is now [section 4.4](#).

Finally, [section 5.2](#) has been added to mention that the signature does not cover option data, which might be a vulnerability.

## 7.0 References

[RFC793] Postel, J., "Transmission Control Protocol," [RFC 793](#), September 1981.

[RFC2460] Deering, S., and Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification," [RFC 2460](#), December 1998.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm," [RFC 1321](#), MIT Laboratory for Computer Science, April 1992.

[RFC1323] Jacobson, V., Braden, R., and Borman, D., "TCP Extensions for High Performance", [RFC 1323](#), LBL, USC/Information Sciences Institute, Cray Research, May 1992.

[Dobb] Dobbertin, H., "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  
Juniper Networks  
1194 N. Mathilda Avenue  
Sunnyvale, CA 94089 USA

Phone: +1 408 745-2037  
Email: ahh@juniper.net