Network Working Group
Request for Comments: 3562
Category:Informational

M. Leech
Nortel Networks

July 2003

Key Management Considerations for the TCP MD5 Signature Option

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 Internet Society (2003). All Rights Reserved.

Abstract

The TCP MD5 Signature Option (RFC 2385), used predominantly by BGP, has seen significant deployment in critical areas of Internet infrastructure. The security of this option relies heavily on the quality of the keying material used to compute the MD5 signature. This document addresses the security requirements of that keying material.

1. Introduction

The security of various cryptographic functions lies both in the strength of the functions themselves against various forms of attack, and also, perhaps more importantly, in the keying material that is used with them. While theoretical attacks against the simple MAC construction used in RFC 2385 are possible [MDXMAC], the number of text-MAC pairs required to mount a forgery make it vastly more probable that key-guessing is the main threat against RFC 2385.

We show a quantitative approach to determining the security requirements of keys used with [RFC2385], which tends to suggest the following:

- Key lengths SHOULD be between 12 and 24 bytes, with larger keys having effectively zero additional computational costs when compared to shorter keys.

- Key sharing SHOULD be limited so that keys aren't shared among multiple BGP peering arrangements.

- Keys SHOULD be changed at least every 90 days.

1.1. Requirements Keywords

The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", and "MAY" that appear in this document are to be interpreted as described in [RFC2119].

2. Performance assumptions

The most recent performance study of MD5 that this author was able to find was undertaken by J. Touch at ISI. The results of this study were documented in [RFC1810]. The assumption is that Moores Law applies to the data in the study, which at the time showed a best-possible *software* performance for MD5 of 87Mbits/second. Projecting this number forward to the ca 2002 timeframe of this document, would suggest a number near 2.1Gbits/second.

For purposes of simplification, we will assume that our key-guessing attacker will attack short packets only. A likely minimal packet is an ACK, with no data. This leads to having to compute the MD5 over about 40 bytes of data, along with some reasonable maximum number of key bytes. MD5 effectively pads its input to 512-bit boundaries (64 bytes) (it's actually more complicated than that, but this simplifying assumption will suffice for this analysis). That means that a minimum MD5 "block" is 64 bytes, so for a ca 2002-scaled software performance of 2.1Gbits/second, we get a single-CPU software MD5 performance near 4.1e6 single-block MD5 operations per second.

These numbers are, of course, assuming that any key-guessing attacker is resource-constrained to a single CPU. In reality, distributed cryptographic key-guessing attacks have been remarkably successful in the recent past.

It may be instructive to look at recent Internet worm infections, to determine what the probable maximum number of hosts that could be surreptitiously marshalled for a key-guessing attack against MD5. CAIDA [CAIDA2001] has reported that the Code Red worm infected over 350,000 Internet hosts in the first 14 hours of operation. It seems reasonable to assume that a worm whose "payload" is a mechanism for quietly performing a key-guessing attack (perhaps using idle CPU cycles of the infected host) could be at least as effective as Code Red was. If one assumes that such a worm were engineered to be maximally stealthy, then steady-state infection could conceivably reach 1 million hosts or more. That changes our single-CPU

performance from 4.1e6 operations per second, to somewhere between 1.0e11 and 1.0e13 MD5 operations per second.

In 1997, John Gilmore, and the Electronic Frontier Foundation [EFF98] developed a special-purpose machine, for an investment of approximately USD$250,000. This machine was able to mount a key-guessing attack against DES, and compute a key in under 1 week. Given Moores Law, the same investment today would yield a machine that could do the same work approximately 8 times faster. It seems reasonable to assume that a similar hardware approach could be brought to bear on key-guessing attacks against MD5, for similar key lengths to DES, with somewhat-reduced performance (MD5 performance in hardware may be as much as 2-3 times slower than DES).

3. Key Lifetimes

Operational experience with RFC 2385 would suggest that keys used with this option may have lifetimes on the order of months. It would seem prudent, then, to choose a minimum key length that guarantees that key-guessing runtimes are some small multiple of the key-change interval under best-case (for the attacker) practical attack performance assumptions.

The keys used with RFC 2385 are intended only to provide authentication, and not confidentiality. Consequently, the ability of an attacker to determine the key used for old traffic (traffic emitted before a key-change event) is not considered a threat.

3. Key Entropy

If we make an assumption that key-change intervals are 90 days, and that the reasonable upper-bound for software-based attack performance is 1.0e13 MD5 operations per second, then the minimum required key entropy is approximately 68 bits. It is reasonable to round this number up to at least 80 bits, or 10 bytes. If one assumes that hardware-based attacks are likely, using an EFF-like development process, but with small-country-sized budgets, then the minimum key size steps up considerably to around 83 bits, or 11 bytes. Since 11 is such an ugly number, rounding up to 12 bytes is reasonable.

In order to achieve this much entropy with an English-language key, one needs to remember that English has an entropy of approximately 1.3 bits per character. Other human languages are similar. This means that a key derived from a human language would need to be approximately 61 bytes long to produce 80 bits of entropy, and 73 bytes to produce 96 bits of entropy.

A more reasonable approach would be to use the techniques described in [RFC1750] to produce a high quality random key of 96 bits or more.

It has previously been noted that an attacker will tend to choose short packets to mount an attack on, since that increases the key-guessing performance for the attacker. It has also been noted that MD5 operations are effectively computed in blocks of 64 bytes. Given that the shortest packet an attacker could reasonably use would consist of 40 bytes of IP+TCP header data, with no payload, the remaining 24 bytes of the MD5 block can reasonably be used for keying material without added CPU cost for routers, but substantially increase the burden on the attacker. While this practice will tend to increase the CPU burden for ordinary short BGP packets, since it will tend to cause the MD5 calculations to overflow into a second MD5 block, it isn't currently seen to be a significant extra burden to BGP routing machinery.

The most reasonable practice, then, would be to choose the largest possible key length smaller than 25 bytes that is operationally reasonable, but at least 12 bytes.

Some implementations restrict the key to a string of ASCII characters, much like simple passwords, usually of 8 bytes or less. The very real risk is that such keys are quite vulnerable to key-guessing attacks, as outlined above. The worst-case scenario would occur when the ASCII key/password is a human-language word, or pseudo-word. Such keys/passwords contain, at most, 12 bits of entropy. In such cases, dictionary driven attacks can yield results in a fraction of the time that a brute-force approach would take. Such implementations SHOULD permit users to enter a direct binary key using the command line interface. One possible implementation would be to establish a convention that an ASCII key beginning with the prefix "0x" be interpreted as a string of bytes represented in hexadecimal. Ideally, such byte strings will have been derived from a random source, as outlined in [RFC1750]. Implementations SHOULD NOT limit the length of the key unnecessarily, and SHOULD allow keys of at least 16 bytes, to allow for the inevitable threat from Moores Law.

4. Key management practices

In current operational use, TCP MD5 Signature keys [RFC2385] may be shared among significant numbers of systems. Conventional wisdom in cryptography and security is that such sharing increases the probability of accidental or deliberate exposure of keys. The more frequently such keying material is handled, the more likely it is to be accidentally exposed to unauthorized parties.

Since it is possible for anyone in possession of a key to forge packets as if they originated with any of the other keyholders, the most reasonable security practice would be to limit keys to use between exactly two parties. Current implementations may make this difficult, but it is the most secure approach when key lifetimes are long. Reducing key lifetimes can partially mitigate widescale key-sharing, by limiting the window of opportunity for a "rogue" keyholder.

Keying material is extremely sensitive data, and as such, should be handled with reasonable caution. When keys are transported electronically, including when configuring network elements like routers, secure handling techniques MUST be used. Use of protocols such as S/MIME [RFC2633], TLS [RFC2246], Secure Shell (SSH) SHOULD be used where appropriate, to protect the transport of the key.

5. Security Considerations

This document is entirely about security requirements for keying material used with RFC 2385.

No new security exposures are created by this document.

6. Acknowledgements

Steve Bellovin, Ran Atkinson, and Randy Bush provided valuable commentary in the development of this document.

7. References

[RFC1771] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.

[RFC1810] Touch, J., "Report on MD5 Performance", RFC 1810, June 1995.

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

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

[MDXMAC] Van Oorschot, P. and B. Preneel, "MDx-MAC and Building Fast MACs from Hash Functions". Proceedings Crypto ’95, Springer-Verlag LNCS, August 1995.

[RFC1750] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[EFF98] "Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design". Electronic Frontier Foundation, 1998.

[RFC2633] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[CAIDA2001] "CAIDA Analysis of Code Red" http://www.caida.org/analysis/security/code-red/

8. Author's Address

Marcus D. Leech
Nortel Networks
P.O. Box 3511, Station C
Ottawa, ON
Canada, K1Y 4H7

Phone: +1 613-763-9145
EMail: mleech@nortelnetworks.com

9. Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS 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.

Acknowledgement

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