Network Working Group
Internet-Draft
Intended status: Experimental
Expires: February 1, 2015

S. Hartman
M. Wasserman
Painless Security
D. Zhang
Huawei
M. Bhatia
A-L
D. He
Huawei

July 31, 2014

Authenticating version 3 of the Simple Network Management Protocol (SNMPv3) using HMAC-SHA-2 procedures
draft-hartman-snmp-sha2-02

Abstract

This document describes the mechanism to authenticate SNMPv3 protocol packets using Hashed Message Authentication Mode (HMAC) with the SHA-256, SHA-384, and SHA-512 algorithms.

Requirements Language

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 RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on February 1, 2015.

Table of Contents

1. Introduction

The cryptographic authentication mechanism proposed in [RFC3414] specifies the support of MD5 [RFC1321] and Secure Hash Algorithm (SHA-1) algorithms for authenticating SNMPv3 packets. The recent escalating series of attacks on MD5 and SHA-1 [SHA-1-attack1] [SHA-1-attack2] raise concerns about their remaining useful lifetime [RFC6151] [RFC6194].

These attacks may not necessarily result in direct vulnerabilities for Keyed-MD5 and Keyed-SHA-1 digests as message authentication codes

because the colliding message may not correspond to a syntactically correct SNMP protocol packet. Regardless, there is a need felt to deprecate MD5 and SHA-1 as the basis for the HMAC algorithm in favor of stronger digest algorithms.

This document adds support for Secure Hash Algorithms (SHA) defined in the US NIST Secure Hash Standard (SHS), which is defined by NIST FIPS 180-2 [FIPS-180-2]. [FIPS-180-2] includes SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. The HMAC authentication mode defined in NIST FIPS 198 is used [FIPS-198].

2. Cryptographic Aspects

In the algorithm description below, the following nomenclature, which is consistent with [FIPS-198], is used:

H is the specific hashing algorithm (e.g. SHA-256).

K is the secret key for authentication.

Ko is the cryptographic key used with the hash algorithm.

B is the block size of H, measured in octets rather than bits. Note that B is the internal block size, not the hash size.

For SHA-1 and SHA-256: B == 64

For SHA-384 and SHA-512: B == 128

L is the length of the hash, measured in octets rather than bits.

XOR is the exclusive-or operation.

Opad is the hexadecimal value 0x5c repeated B times.

Ipad is the hexadecimal value 0x36 repeated B times.

Apad is the hexadecimal value of source IPv4 address repeated (L/4) times (repeated (L/16) for IPV6 addresses).

(1) Preparation of the Key

In this application, Ko is always L octets long.

If the Authentication Key (K) is L octets long, then Ko is equal to K. If the Authentication Key (K) is more than L octets long, then Ko is set to H(K). If the Authentication Key (K) is less than L octets long, then Ko is set to the Authentication Key (K) with zeros

appended to the end of the Authentication Key (K) such that Ko is L octets long.

(2) First Hash

First, the AuthenticationParameters field is filled with the value of Apad and is set to the serialization, according to the rules in [RFC3417].

Then, a first hash, also known as the inner hash, is computed as follows:

First-Hash = H(Ko XOR Ipad || (SNMP Packet))

(3) Second Hash T

Then a second hash, also known as the outer hash, is computed as follows:

Second-Hash = H(Ko XOR Opad || First-Hash)

(4) Result

The resultant Second-Hash becomes the Authentication Data that is sent in the AuthenticationParameters field. The length of the AuthenticationParameters field is always identical to the message digest size of the specific hash function H that is being used.

3. Authentication Protocols using SHA-2

This section introduces how the authentication protocols using SHA-2 work. The protocols are identical to the authentication protocols proposed in the [RFC3414] except the authentication algorithms used in generating digests.

3.1. Elements of Authentication Protocols

3.1.1. Users

Authentication using the authentication protocols makes use of a defined set of userNames. For any user on whose behalf a message must be authenticated at a particular SNMP engine, that SNMP engine must have knowledge of that user. An SNMP engine that wishes to communicate with another SNMP engine must also have knowledge of a user known to that engine, including knowledge of the applicable attributes of that user.

A user and its attributes are defined as follows:

<userName> A string representing the name of the user.

<authKey> A user's secret key to be used when calculating a digest. Specifically, the value SHOULD be 32 octets for HMAC-SHA-256, 48 octets for HMAC-SHA-384, and 64 octets for HMAC- SHA-512.

3.1.2. SNMP Messages Using this Authentication Protocol

Messages using this authentication protocol carry a msgAuthenticationParameters field as part of the msgSecurityParameters. For this protocol, the msgAuthenticationParameters field is the serialized OCTET STRING representing the SHA-2 output done over the wholeMsg. Specifically, the length of this field is 32 octets for HMAC-SHA-256, 48 octets for HMAC-SHA-384, and 64 octets for HMAC- SHA-512. The digest is calculated over the wholeMsg so if a message is authenticated, which also means that all the fields in the message are intact and have not been tampered with.

3.2. Services Provided by the Authentication Module

3.2.1. Services for Generating an Outgoing SNMP Message

The authentication protocol assumes that the selection of the authKey is done by the caller and that the caller passes the secret key to be used. Upon completion the authentication module returns statusInformation and, if the message digest was correctly calculated, the wholeMsg with the digest inserted at the proper place. The abstract service primitive is:

```
 statusInformation =              -- success or failure
   authenticateOutgoingMsg(
   IN   authKey                   -- secret key for authentication
   IN   wholeMsg                  -- unauthenticated complete message
   OUT  authenticatedWholeMsg     -- complete authenticated message
        )
```

The abstract data elements are:

statusInformation: An indication of whether the authentication process was successful. If not it is an indication of the problem.

authKey: The secret key to be used by the authentication algorithm.

wholeMsg: The message to be authenticated.

authenticatedWholeMsg: The authenticated message (including inserted digest) on output.

3.2.2. Services for Processing an Incoming SNMP Message

The authentication protocol assumes that the selection of the authKey is done by the caller and that the caller passes the secret key to be used. Upon completion the authentication module returns statusInformation and, if the message digest was correctly calculated, the wholeMsg as it was processed. The abstract service primitive is:

```
statusInformation =              -- success or failure
  authenticateIncomingMsg(
  IN   authKey                   -- secret key for authentication
  IN   authParameters            -- as received on the wire
  IN   wholeMsg                  -- as received on the wire
  OUT  authenticatedWholeMsg     -- complete authenticated message
       )
```

The abstract data elements are:

statusInformation: An indication of whether the authentication process was successful. If not it is an indication of the problem.

authKey: The secret key to be used by the authentication algorithm.

authParameters: The authParameters from the incoming message.

wholeMsg: The message to be authenticated on input and the authenticated message on output.

authenticatedWholeMsg: The whole message after the authentication check is complete.

4. Elements of Procedure

4.1. Procedures at the Sending Side

Before a SNMPv3 device sends an SNMP packet out, the device needs to select an appropriate key for authentication if a keyed digest for the packet is required. If no appropriate key is available, the SNMP packet MUST be discarded.

If an appropriate key for authentication is available, the device then finds the authentication algorithm (HMAC-SHA-256, HMAC-SHA-384 or HMAC-SHA-512) associated with the key.

Then, the operations illustrated in Section 2 are performed.

The authenticatedWholeMsg is then returned to the caller together with statusInformation indicating success.

4.2. Procedure at the Receiving Side

Upon receiving an SNMP packet with a msgAuthenticationParameters field appended, a device needs to locate an appropriate key for authentication to verify the packet.

If there is no key found or the length of the digest received in the msgAuthenticationParameters field does not match the length associated with authentication algorithm, the received packet MUST be discarded.

An authentication algorithm dependent process then needs to be performed by using the algorithm specified by the appropriate key for the received packet.

Before the device performs any processing, it needs to save the content of the AuthenticationParameters field and set the AuthenticationParameters field with Apad.

Then, the operations illustrated in Section 2 are performed. The calculated data is compared with the received authentication data in the packet.

The packet MUST be discarded if the calculated and the received authentication data do not match. In this case, a failure and an errorIndication (authenticationFailure) are returned to the calling module. Otherwise, the authenticatedWholeMsg and statusInformation indicating success are then returned to the caller.

5. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

7. Acknowledgements

This work makes use of signifitent texts from that RFC3414.

8. References

8.1. Normative References

[FIPS-180-2]
National Institute of Standards and Technology, FIPS PUB 180-2, "The Keyed-Hash Message Authentication Code (HMAC)", August 2002.

[FIPS-198]
National Institute of Standards and Technology, FIPS PUB 198, "The Keyed-Hash Message Authentication Code (HMAC)", March 2002.

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

[RFC3417] Presuhn, R., "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.

[RFC6039] Manral, V., Bhatia, M., Jaeggli, J., and R. White, "Issues with Existing Cryptographic Protection Methods for Routing Protocols", RFC 6039, October 2010.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, March 2011.

8.2. Informative References

[Dobb96a] Dobbertin, H., "Cryptanalysis of MD5 Compress", May 1996.

[Dobb96b] Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes", 1996.

[I-D.ietf-karp-design-guide]
Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Design Guidelines", draft-ietf-karp-design-guide-10 (work in progress), December 2011.

[MD5-attack]
Wang, X., Feng, D., Lai, X., and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", August 2004.

[NIST-HMAC-SHA]
National Institute of Standards and Technology, Available online at http://csrc.nist.gov/groups/ST/hash/policy.html, "NIST's Policy on Hash Functions", 2006.

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

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4822] Atkinson, R. and M. Fanto, "RIPv2 Cryptographic Authentication", RFC 4822, February 2007.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, February 2009.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[SHA-1-attack1]
Wang, X., Yin, Y., and H. Yu, "Finding Collisions in the Full SHA-1", 2005.

[SHA-1-attack2]
Wang, X., Yao, A., and F. Yao, "New Collision Search for SHA-1", 2005.

Authors' Addresses

Sam Hartman
Painless Security
356 Abbott Street
North Andover, MA 01845
USA

Email: hartmans@painless-security.com
URI: http://www.painless-security.com

Margaret Wasserman
Painless Security
356 Abbott Street
North Andover, MA 01845
USA

Phone: +1 781 405 7464
Email: mrw@painless-security.com
URI: http://www.painless-security.com

Dacheng Zhang
Huawei
Beijing
China

Email: zhangdacheng@huawei.com

Manav Bhatia
A-L
India

Email: manav.bhatia@alcatel-lucent.com

Danping He
Huawei
Beijing
China

Email: ana.hedanping@huawei.com