Network Working Group R. Housley
Internet-Draft Vigil Security
Updates: 5652 (if approved) October 03, 2019
Intended status: Standards Track
Expires: April 5, 2020

Update to the Cryptographic Message Syntax (CMS) for Algorithm Identifier Protection
draft-housley-lamps-cms-update-alg-id-protect-00

Abstract

This document updates the Cryptographic Message Syntax (CMS) specified in RFC 5652 to ensure that algorithm identifiers are adequately protected.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. Require use the same hash algorithm 3
3.1. RFC 5652, Section 5.3 3
3.2. RFC 5652, Section 5.4 4
3.3. RFC 5652, Section 5.6 4
3.4. Backward Compatibility Considerations 5
3.5. Timestamp Compatibility Considerations 5
4. Recommend inclusion of the CMSAlgorithmProtection attribute . 5
4.1. RFC 5652, Section 14 6
5. IANA Considerations 6
6. Security Considerations 6
7. Acknowledgements 6
8. References 6
8.1. Normative References 6
8.2. Informative References 7
Author's Address 8

1. Introduction

This document updates the Cryptographic Message Syntax (CMS) [RFC5652] to ensure that algorithm identifiers are adequately protected.

The CMS Signed-data Content Type [RFC5652], unlike X.509 certificates [RFC5280], can be vulnerable to algorithm substitution attacks. In an algorithm substitution attack, the attacker changes either the algorithm identifier or the parameters associated with the algorithm identifier to change the verification process used by the recipient. The X.509 certificate structure protects the algorithm identifier and the associate parameters by signing them.

In an algorithm substitution attack, the attacker looks for a different algorithm that produces the same result as the algorithm used by the originator. As an example, if the signer of a message used SHA-256 [SHS] as the digest algorithm to hash the message content, then the attacker looks for a weaker hash algorithm that produces a result that is of the same length. The attacker's goal is to find a different message that results in the same hash value, which is commonly called a collision. Today, there are many hash functions that produce 256-bit results. One of them may be found to be weak in the future.

Further, when a digest algorithm produces a larger result than is needed by a digital signature algorithm, the digest value is reduced to the size needed by the signature algorithm. This can be done both

by truncation and modulo operations, with the simplest being straightforward truncation. In this situation, the attacker needs to find a collision with the reduced digest value. As an example, if the message signer uses SHA-512 [SHS] as the digest algorithm and ECDSA with the P-256 curve [DSS] as the signature algorithm, then the attacker needs to find a collision with the first half of the digest.

Similar attacks can be mounted against parameterized algorithm identifiers. When looking at randomized hash functions, such as the example in [RFC6210], the algorithm identifier parameter includes a random value that can be manipulated by an attacker looking for collisions. Some other algorithm identifiers include complex parameter structures, and each value provides another opportunity for manipulation by an attacker.

This document makes two updates to CMS to provide similar protection for the algorithm identifier. First, it mandates a convention followed by many implementations by requiring the originator to use the same hash algorithm to compute the digest of the message content and the digest of signed attributes. Second, it recommends that the originator include the CMSAlgorithmProtection attribute [RFC6211].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Require use the same hash algorithm

This section updates [RFC5652] to require the originator to use the same hash algorithm to compute the digest of the message content and the digest of signed attributes.

3.1. RFC 5652, Section 5.3

Change the paragraph describing the digestAlgorithm as follows:

OLD:

> digestAlgorithm identifies the message digest algorithm, and any associated parameters, used by the signer. The message digest is computed on either the content being signed or the content together with the signed attributes using the process described in Section 5.4. The message digest algorithm SHOULD be among those listed in the digestAlgorithms field of the associated SignerData.

Implementations MAY fail to validate signatures that use a digest algorithm that is not included in the SignedData digestAlgorithms set.

NEW:

digestAlgorithm identifies the message digest algorithm, and any associated parameters, used by the signer. The message digest is computed on either the content being signed or the content together with the signed attributes using the process described in Section 5.4. The message digest algorithm SHOULD be among those listed in the digestAlgorithms field of the associated SignerData. If signedAttrs are present in the SignerInfo, then the same digest algorithm MUST be used to compute the digest of the SignedData encapContentInfo eContent, which is carried in the message-digest attribute, and to compute the digest of the DER-encoded SET OF signed attributes, which is passed to the signature algorithm. Implementations MAY fail to validate signatures that use a digest algorithm that is not included in the SignedData digestAlgorithms set.

3.2. RFC 5652, Section 5.4

Add the following paragraph as the second paragraph in Section 5.4:

ADD:

When the signedAttrs field is present, the same digest algorithm MUST be used to compute the digest of the the encapContentInfo eContent OCTET STRING, which is carried in the message-digest attribute, and the collection of attributes that are signed.

3.3. RFC 5652, Section 5.6

Change the paragraph discussing the signedAttributes as follows:

OLD:

The recipient MUST NOT rely on any message digest values computed by the originator. If the SignedData signerInfo includes signedAttributes, then the content message digest MUST be calculated as described in Section 5.4. For the signature to be valid, the message digest value calculated by the recipient MUST be the same as the value of the messageDigest attribute included in the signedAttributes of the SignedData signerInfo.

NEW:

The recipient MUST NOT rely on any message digest values computed by the originator. If the SignedData signerInfo includes signedAttributes, then the content message digest MUST be calculated as described in Section 5.4, using the same digest algorithm to compute the digest of the the encapContentInfo eContent OCTET STRING and the message-digest attribute. For the signature to be valid, the message digest value calculated by the recipient MUST be the same as the value of the messageDigest attribute included in the signedAttributes of the SignedData signerInfo.

3.4. Backward Compatibility Considerations

The new requirement introduced above might lead to compatibility with an implementation that allowed different digest algorithms to be used to compute the digest of the message content and the digest of signed attributes. The signatures produced by such an implementation when two different digest algorithms are used will be considered invalid by an implementation that follows this specification. However, most, if not all, implementations already require the originator to use the same digest algorithm for both operations.

READER:

If you have an implementation that allows different digest algorithms to be used to compute the digest of the message content and the digest of signed attributes, please tell us on the spasm@ietf.org mail list.

3.5. Timestamp Compatibility Considerations

The new requirement introduced above might lead to compatibility issues for timestamping systems when the originator does not wish to share the message content with the Time Stamp Authority (TSA) [RFC3161]. In this situation, the originator sends a TimeStampReq to the TSA that includes a MessageImprint, which consists of a digest algorithm identifier and a digest value, then the TSA uses the digest in the MessageImprint. As a result, the signature algorithm used by the TSA needs to be compatible with the digest algorithm selected by the originator for the MessageImprint.

4. Recommend inclusion of the CMSAlgorithmProtection attribute

This section updates [RFC5652] to recommend that the originator include the CMSAlgorithmProtection attribute [RFC6211] whenever signed attributes or authenticated attributes are present.

4.1. RFC 5652, Section 14

Add the following paragraph as the eighth paragraph in Section 14:

ADD:

While no known algorithm substitution attacks are known at this time, the inclusion of the algorithm identifiers used by the originator as a signed attribute or an authenticated attribute makes such an attack significantly more difficult. Therefore, the originator of a Signed-data content type that includes signed attributes SHOULD include the CMSAlgorithmProtection attribute [RFC6211] as one of the signed attributes. Likewise, the originator of an Authenticated-data content type that includes authenticated attributes SHOULD include the CMSAlgorithmProtection attribute [RFC6211] as one of the authenticated attributes.

5. IANA Considerations

This document makes no requests of the IANA.

6. Security Considerations

The security considerations of [RFC5652] are updated ensure that algorithm identifiers are adequately protected, which makes algorithm substitution attacks significantly more difficult.

The CMSAlgorithmProtection attribute [RFC6211] offers protection the algorithm identifiers used in the signed-data and authenticated-data content types. There is not currently protection mechanism for the algorithm identifiers used in the enveloped-data, digested-data, or encrypted-data content types. Likewise there us not currently protection mechanism for the algorithm identifiers used in the authenticated-enveloped-data content type defined in [RFC5083].

7. Acknowledgements

Many thanks to Jim Schaad and Peter Gutmann; without knowing it, they motivated me to write this document.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC6211] Schaad, J., "Cryptographic Message Syntax (CMS) Algorithm Identifier Protection Attribute", RFC 6211, DOI 10.17487/RFC6211, April 2011, <https://www.rfc-editor.org/info/rfc6211>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[DSS] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS Publication 186-3, June 2009.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, DOI 10.17487/RFC5083, November 2007, <https://www.rfc-editor.org/info/rfc5083>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6210] Schaad, J., "Experiment: Hash Functions with Parameters in the Cryptographic Message Syntax (CMS) and S/MIME", RFC 6210, DOI 10.17487/RFC6210, April 2011, <https://www.rfc-editor.org/info/rfc6210>.

[SHS] National Institute of Standards and Technology (NIST), "Secure Hash Standard", FIPS Publication 180-3, October 2008.

Author's Address

Russ Housley
Vigil Security
516 Dranesville Road
Herndon, VA 20170
US

Email: housley@vigilsec.com