Internet Engineering Task Force (IETF)
Request for Comments: 8933
Updates: 5652
Category: Standards Track
ISSN: 2070-1721

R. Housley
Vigil Security
October 2020

Update to the Cryptographic Message Syntax (CMS) for Algorithm Identifier Protection

Abstract

This document updates the Cryptographic Message Syntax (CMS) specified in RFC 5652 to ensure that algorithm identifiers in signed-data and authenticated-data content types are adequately protected.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document updates the Cryptographic Message Syntax (CMS) [RFC5652] to ensure that algorithm identifiers in signed-data and authenticated-data content types 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 associated 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 called a cross-algorithm 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 the Elliptic Curve Digital Signature Algorithm (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 randomized hash functions are employed, 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 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. Required Use of 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 signedAttrs 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 the signedAttrs field is present in the SignerInfo, then the
|  same digest algorithm MUST be used to compute both the digest of
|  the SignedData encapContentInfo eContent, which is carried in the
|  message-digest attribute, and the digest of the DER-encoded
|  signedAttrs, 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 encapContentInfo
|  eContent OCTET STRING, which is carried in the message-digest
|  attribute and the digest of the collection of attributes that are
|  signed.
```

3.3. RFC 5652, Section 5.6

Change the paragraph discussing the signed attributes 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 the
|  signedAttrs field, 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 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 signedAttrs field of the SignedData signerInfo.

3.4. Backward Compatibility Considerations

The new requirement introduced above might lead to incompatibility 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.

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 Stamping 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. The TSA then uses the originator-provided digest in the MessageImprint.

When producing the TimeStampToken, the TSA MUST use the same digest algorithm to compute the digest of the encapContentInfo eContent, which is an OCTET STRING that contains the TSTInfo, and the message-digest attribute within the SignerInfo.

To ensure that TimeStampToken values that were generated before this update remain valid, no requirement is placed on a TSA to ensure that the digest algorithm for the TimeStampToken matches the digest algorithm for the MessageImprint embedded within the TSTInfo.

4. Recommended 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 there are no known algorithm substitution attacks today, 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 has no IANA actions.

6. Security Considerations

The security properties of the CMS [RFC5652] signed-data and authenticated-data content types are updated to offer protection for algorithm identifiers, which makes algorithm substitution attacks significantly more difficult.

For the signed-data content type, the improvements specified in this document force an attacker to mount a hash algorithm substitution attack on the overall signature, not just on the message digest of the encapContentInfo eContent.

Some digital signature algorithms have prevented hash function substitutions by including a digest algorithm identifier as an input to the signature algorithm. As discussed in [HASHID], such a "firewall" may not be effective or even possible with newer signature algorithms. For example, RSASSA-PKCS1-v1_5 [RFC8017] protects the digest algorithm identifier, but RSASSA-PSS [RFC8017] does not. Therefore, it remains important that a signer have a way to signal to a recipient which digest algorithms are allowed to be used in conjunction with the verification of an overall signature. This signaling can be done as part of the specification of the signature algorithm in an X.509v3 certificate extension [RFC5280] or some other means. The Digital Signature Standard (DSS) [DSS] takes the first approach by requiring the use of an "approved" one-way hash algorithm.

For the authenticated-data content type, the improvements specified in this document force an attacker to mount a MAC algorithm substitution attack, which is difficult because the attacker does not know the authentication key.

The CMSAlgorithmProtection attribute [RFC6211] offers protection for the algorithm identifiers used in the signed-data and authenticated-data content types. However, no protection is provided for the algorithm identifiers in the enveloped-data, digested-data, or encrypted-data content types. Likewise, the CMSAlgorithmProtection attribute provides no protection for the algorithm identifiers used in the authenticated-enveloped-data content type defined in [RFC5083]. A mechanism for algorithm identifier protection for these content types is work for the future.

7. References

7.1. 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>.

[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>.

[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>.

7.2. Informative References

[DSS] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", FIPS 186-4, DOI 10.6028/NIST.FIPS.186-4, July 2013, <https://doi.org/10.6028/NIST.FIPS.186-4>.

[HASHID] Kaliski, B., "On Hash Function Firewalls in Signature Schemes", DOI 10.1007/3-540-45760-7_1, Lecture Notes in Computer Science, Volume 2271, February 2002, <https://doi.org/10.1007/3-540-45760-7_1>.

[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>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[SHS] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", FIPS 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

Acknowledgements

Many thanks to Jim Schaad and Peter Gutmann; without knowing it, they motivated me to write this document. Thanks to Roman Danyliw, Ben Kaduk, and Peter Yee for their careful review and editorial suggestions.

Author’s Address

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
United States of America

Email: housley@vigilsec.com