Limited Additional Mechanisms for PKIX and SMIME
Internet-Draft
Intended status: Standards Track
Expires: 19 September 2025

D. Van Geest
CryptoNext Security
F. Strenzke
MTG AG

18 March 2025

EUF-CMA for the Cryptographic Message Syntax (CMS) SignedData
draft-vangeest-lamps-cms-euf-cma-signeddata-01

Abstract

The Cryptographic Message Syntax (CMS) has different signature verification behaviour based on whether signed attributes are present or not. This results in a potential existential forgery vulnerability in CMS and protocols which use CMS. This document describes the vulnerability and lists a number of potential mitigations for LAMPS working group discussion.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://danvangeest.github.io/cms-euf-cma-signeddata/draft-vangeest-lamps-cms-euf-cma-signeddata.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-vangeest-lamps-cms-euf-cma-signeddata/.

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME Working Group mailing list (mailto:spasm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spasm/. Subscribe at https://www.ietf.org/mailman/listinfo/spasm/.

Source for this draft and an issue tracker can be found at https://github.com/danvangeest/cms-euf-cma-signeddata.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Cryptographic Message Syntax (CMS) [RFC5652] signed-data content type allows any number of signers in parallel to sign any type of content.

CMS gives a signer two options when generating a signature on some content:

* Generate a signature on the whole content; or

* Compute a hash over the content, place this hash in the message-digest attribute in the SignedAttributes type, and generate a signature on the SignedAttributes.

The resulting signature does not commit to the presence of the SignedAttributes type, allowing an attacker to influence verification behaviour. An attacker can perform two different types of attacks:

1. Take an arbitrary CMS signed message M which was originally signed with SignedAttributes present and rearrange the structure such that the SignedAttributes field is absent and the original DER-encoded SignedAttributes appears as an encapsulated or detached content of type id-data, thereby crafting a new structure M' that was never explicitly signed by the signer. M' has the DER-encoded SignedAttributes of the original message as its content and verifies correctly against the original signature of M.

2. Let the signer sign a message of the attacker's choice without SignedAttributes. The attacker chooses this message to be a valid DER-encoding of a SignedAttributes object. He can then add this encoded SignedAttributes object to the signed message and change the signed message to the one that was used to create the messageDigest attribute within the SignedAttributes. The signature created by the signer is valid for this arbitrary attacker-chosen message.

This vulnerability was presented by Falko Strenzke at IETF 121 [LAMPS121] and is detailed in [Str23].

Due to the limited flexibility of either the signed or the forged message in either attack variant, the fraction of vulnerable systems can be assumed to be small. But due to the wide deployment of the affected protocols, such instances cannot be excluded.

2. Conventions and Definitions

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. Potential Mitigations

Potential mitigations are described in the following sub-sections as input to the working group discussion. If this draft is adopted and the working group has taken a decision which measure(s) should be realized, we'll describe the chosen measures in detail.

The mitigations in this section make use of a context string which is passed to the signature algorithm's sign and verify functions.

ML-DSA [FIPS204], SLH-DSA [FIPS205], Composite ML-DSA [I-D.ietf-lamps-pq-composite-sigs], and Ed448 [RFC8032] take a context string during signing and verification. The context string may be up to 255 bytes long. By default the context string is the empty string.

```
Sign(sk, M, ctx="")
Verify(sk, M, ctx="")
```

RSA, ECDSA and Ed25519 signatures do not take a context string and would not be helped by these mitigations.

Ed448 can take a context string but does not currently in CMS [RFC8419].

Ed25519ctx [RFC8032] takes a context string but is not specified for use in CMS.

3.1. Immediate Forced Use of Specific Signature Context Strings

Immediately update [I-D.ietf-lamps-cms-ml-dsa], [I-D.ietf-lamps-cms-sphincs-plus], and [I-D.ietf-lamps-pq-composite-sigs] to require a context string, with a different value for use with and without signated attributes.

When signed attributes are present:

```
Sign(sk, M, "signed-attributes")
Verify(sk, M, "signed-attributes")
```

When signed attributes are absent:

```
Sign(sk, M, "no-signed-attributes")
Verify(sk, M, "no-signed-attributes")
```

Unlike the following mitigations, Ed448 cannot be addressed by this mitigation because it is already published and in use.

3.2. Attribute-Specified Use of Implicit Signature Context Strings

Like Section 3.1, but the use of the signature context string is indicated by a new, empty (or attribute value ignored), sign-with-context-implicit unsigned attribute.

[I-D.ietf-lamps-cms-ml-dsa], [I-D.ietf-lamps-cms-sphincs-plus], and [I-D.ietf-lamps-pq-composite-sigs] can be published using the default signature context string. ML-DSA, SLH-DSA, Composite-ML-DSA, and Ed448 only use the non-default context string when the new attribute is used.

3.2.1. Signing

When signed attributes are present:

```
unsigned-attributes.add(sign-with-context-implicit)
Sign(sk, M, "signed-attributes")
```

When signed attributes are absent:

```
unsigned-attributes.add(sign-with-context-implicit)
Sign(sk, M, "no-signed-attributes")
```

3.2.2. Verifying

When signed attributes are present:

```
IF unsigned-attributes.contains(sign-with-context-implicit)
THEN Verify(sk, M, "signed-attributes")
ELSE Verify(sk, M, "")
```

When signed attributes are absent:

```
IF unsigned-attributes.contains(sign-with-context-implicit)
THEN Verify(sk, M, "no-signed-attributes")
ELSE Verify(sk, M, "")
```

3.3. Attribute-Specified Use of Explicit Signature Context Strings

Like Section 3.2 but the new unsigned attribute (sign-with-context-explict) contains a semi-colon-delimited list of keyword (and optional value) strings. This addresses the possibility of future CMS features that require context parameters.

```
ctx = "<keyword_1>[=value1];...;<keyword_n>[=value]"
```

The list is ordered alphabetically by type string. This list is validated by the verifier and used as the signature context string. (alternative: the SHA-256 hash of the list is used as the signature context string to avoid it getting too long)

A proposed list of initial signature context string keywords follows:

keyword	value	comment
"IETF/CMS"		REQUIRED to be in the sign-with-context-implicit attribute, to differentiate a signature in CMS from a signature with the same private key over some other data.
"signed-attrs"		Present if signed attributes are used, not present if signed attributes are not used. Alternative: always present, value = 0/1, yes/no depending on whether signed attributes are present or not.
"app-ctx"	base64(SHA-256(protocol_context))	Allows the protocol using CMS to specify a context. SHA-256 is applied so that the length available to the protocol context isn't dependent on the other context values used in CMS. (alternative: no SHA-256 here, apply SHA-256 to the whole CMS context). base64-encoding is applied so the app context doesn't introduce semi-colons to mess up CMS' parsing of this string.

Table 1: Potential Context String Keywords

When a verifier processes a SignerInfo containing the sign-with-context-explicit attribute, it MUST perform the following consistency checks:

* If the "signed-attrs" keyword is present and SignedAttributes is not present in the SignerInfo, fail verification.

* If the "signed-attrs" keyword is not present and SignedAttributes is present in the SignerInfo, fail verification.

If the consistency checks pass, the signature is verified using the string in the sign-with-context-explicit attribute as the signature context (alternative: using SHA-256 of the string in the sign-with-context-explicit attribute).

When a verifier processes a SignerInfo without the sign-with-context-explicit attribute, they MUST verify the signature using the default signature context value ("").

[I-D.ietf-lamps-cms-ml-dsa], [I-D.ietf-lamps-cms-sphincs-plus], and [I-D.ietf-lamps-pq-composite-sigs] can be published using the default signature context string. ML-DSA, SLH-DSA, Composite-ML-DSA, and Ed448 only use the non-default context string when the new attribute is used.

4. Straw Mitigations

The following mitigations might not be good ideas but are included just in case there’s a seed of genius in them.

4.1. Attack Detection in CMS

If eContentType is id-data and SignedAttributes is not present, check if the encapsulated or detached content is a valid DER-encoded SignedAttributes structure and fail if it is. The mandatory contentType and messageDigest attributes, with their respective OIDs, should give a low probability of a legitimate message being flagged.

If an application protocol deliberately uses such a signed messages, verification would fail.

This mitigation does not address the inverse problem where a protocol doesn’t used SignedAttributes but for some reason often sends messages which happen to be formatted like valid SignedAttributes encodings, with attacker-controlled bytes where the message digest attribute would be.

4.2. Always/Never use SignedAttributes in Your Protocol

Individually update each protocol which use CMS to always require or forbid signed attributes. In particular, if an eContentType other than id-data is used, Section 5.3 of [RFC5652] requires that signed attributes are used. During verification, ensure that you are receiving the expected (non-id-data) eContentType and that signed attribues are present.

4.3. Attack Detection in Your Protocol

Section 4.1 but specified in the protocol that uses CMS rather than CMS itself.

4.4. Check Content Type Consistency

Check that the content type the EncapsulatedContentInfo value being verified is consistent with your application.

If the content type of the EncapsulatedContentInfo value being verified is not id-data and signed attributes are not present, verification MUST fail.

5. RFCs Using the id-data eContentType

The RFCs in the following subsections use the id-data eContentType. This table summarizes their usages of signed attributes.

RFC	Signed Attributes Usage
[RFC8894]	Requires the used of signed attributes
[RFC8572]	Says nothing about signed attributes
[RFC8551]	RECOMMENDS signed attributes
[RFC6257]	Forbids signed attributes
[RFC5751]	RECOMMENDS signed attributes
[RFC5655]	Says nothing about signed attributes
[RFC5636]	Forbids signed attributes
[RFC5126]	Requires signed attributes
[RFC5024]	Says nothing about signed attributes
[RFC3851]	RECOMMENDS signed attributes
[RFC3126]	Requires signed attributes
[RFC2633]	RECOMMENDS signed attributes

Table 2: RFCs using id-data

An RFC requiring or forbidding signed attributes does not necessarily mean that a verifier will enforce this requirement when verifying, their CMS implementation may simply process the message whether or not signed attributes are present. If one of the signed attributes is necessary for the verifier to successfully verify the signature or to successfully process the CMS data then the attack will not apply; at least not when assuming the signer is well-behaved and always signs with signed attributes present in accordance with the applicable specification.

5.1. RFC 8894 Simple Certificate Enrolment Protocol

Figure 6 in Section 3 of [RFC8894] specifies id-data as the eContentType, and shows the use of signedAttrs. The document itself never refers to signed attributes, but instead to authenticated attributes and an authenticatedAttributes type. Errata ID 8247 clarifies that it should be "signed attributes" and "signedAttrs".

Since SCEP requires the use of signedAttrs with the id-data eContentType, and the recipient must process at least some of the signed attributes, it is not affected by the attack.

5.2. RFC 8572 Secure Zero Touch Provisioning (SZTP)

Section 3.1 of [RFC8572] allows the use of the id-data eContentType, although it also defines more specific content types. It does not say anything about signed attributes.

5.3. S/MIME RFCs

[RFC8551], [RFC5751], [RFC3851], and [RFC2633] require the use of the id-data eContentType.

Section 2.5 of [RFC8551] says:

> Receiving agents MUST be able to handle zero or one instance of each of the signed attributes listed here. Sending agents SHOULD generate one instance of each of the following signed attributes in each S/MIME message:

and

> Sending agents SHOULD generate one instance of the signingCertificate or signingCertificateV2 signed attribute in each SignerInfo structure.

So the use of signed attributes is not an absolute requirement.

5.4. RFC 6257 Bundle Security Protocol Specification

Section 4 of [RFC6257] says:

> In all cases where we use CMS, implementations SHOULD NOT include additional attributes whether signed or unsigned, authenticated or unauthenticated.

5.5. RFC 5655 IP Flow Information Export (IPFIX)

[RFC5655] is a file format that uses CMS for detached signatures. It says nothing about the use of signed attributes.

5.6. RFC 5636 Traceable Anonymous Certificate

Appendix C.1.2 of [RFC5636] says:

> The signedAttr element MUST be omitted.

5.7. RFC 5126 CMS Advanced Electronic Signatures (CAdES)

Section 4.3.1 of [RFC5126] specifies mandatory signed attributes.

One of the signed attributes is used to determine which certificate is used to verify the signature, so this CaDES is not affected by the attack.

5.8. RFC 5024 ODETTE File Transfer Protocol 2

[RFC5024] uses the id-data eContentType and says nothing about signed attributes.

5.9. RFC 3126 Electronic Signature Formats for long term electronic signatures

Section 6.1 of [RFC3126] requires the MessageDigest attribute, which is a signed attribute.

6. Security Considerations

TODO Security

The vulnerability is not present in systems where the use of SignedAttributes is mandatory, for example: SCEP, Certificate Transparency, RFC 4018 firmware update, German Smart Metering CMS data format. Any protocol that uses an eContentType other than id-data is required to use signed attributes. However, this security relies on a correct implementation of the verification routine that ensures the presence of SignedAttributes.

The vulnerability is also not present when the message is signed and then encrypted, as the attacker cannot learn the signature.

Conceivably vulnerable systems (TODO: describe these better):

* Unencrypted firmware update denial of service
* Dense message space
* Signing unstructured data
* External signatures over unstructured data
* Systems with permissive parsers

7. IANA Considerations

This document has no IANA actions.

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/rfc/rfc2119>.

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

[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/rfc/rfc8174>.

8.2. Informative References

[FIPS204] "Module-lattice-based digital signature standard", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.fips.204, August 2024, <https://doi.org/10.6028/nist.fips.204>.

[FIPS205] "Stateless hash-based digital signature standard", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.fips.205, August 2024, <https://doi.org/10.6028/nist.fips.205>.

[I-D.ietf-lamps-cms-ml-dsa] S, B., R, A., and D. Van Geest, "Use of the ML-DSA Signature Algorithm in the Cryptographic Message Syntax (CMS)", Work in Progress, Internet-Draft, draft-ietf-lamps-cms-ml-dsa-02, 17 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-ml-dsa-02>.

[I-D.ietf-lamps-cms-sphincs-plus] Housley, R., Fluhrer, S., Kampanakis, P., and B. Westerbaan, "Use of the SLH-DSA Signature Algorithm in the Cryptographic Message Syntax (CMS)", Work in Progress, Internet-Draft, draft-ietf-lamps-cms-sphincs-plus-19, 13 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-sphincs-plus-19>.

[I-D.ietf-lamps-pq-composite-sigs] Ounsworth, M., Gray, J., Pala, M., Klauテ殤er, J., and S. Fluhrer, "Composite ML-DSA for use in X.509 Public Key Infrastructure and CMS", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-sigs-04, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-04>.

[LAMPS121] Strenzke, F., "EUF-CMA for CMS SignedData", 6 November 2024, <https://datatracker.ietf.org/meeting/121/materials/slides-121-lamps-cms-euf-cma-00>.

[RFC2633] Ramsdell, B., Ed., "S/MIME Version 3 Message Specification", RFC 2633, DOI 10.17487/RFC2633, June 1999, <https://www.rfc-editor.org/rfc/rfc2633>.

[RFC3126] Pinkas, D., Ross, J., and N. Pope, "Electronic Signature Formats for long term electronic signatures", RFC 3126, DOI 10.17487/RFC3126, September 2001, <https://www.rfc-editor.org/rfc/rfc3126>.

[RFC3851] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, DOI 10.17487/RFC3851, July 2004, <https://www.rfc-editor.org/rfc/rfc3851>.

[RFC5024] Friend, I., "ODETTE File Transfer Protocol 2.0", RFC 5024, DOI 10.17487/RFC5024, November 2007, <https://www.rfc-editor.org/rfc/rfc5024>.

[RFC5126] Pinkas, D., Pope, N., and J. Ross, "CMS Advanced Electronic Signatures (CAdES)", RFC 5126, DOI 10.17487/RFC5126, March 2008, <https://www.rfc-editor.org/rfc/rfc5126>.

[RFC5636] Park, S., Park, H., Won, Y., Lee, J., and S. Kent, "Traceable Anonymous Certificate", RFC 5636, DOI 10.17487/RFC5636, August 2009, <https://www.rfc-editor.org/rfc/rfc5636>.

[RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, DOI 10.17487/RFC5655, October 2009, <https://www.rfc-editor.org/rfc/rfc5655>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <https://www.rfc-editor.org/rfc/rfc5751>.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, DOI 10.17487/RFC6257, May 2011, <https://www.rfc-editor.org/rfc/rfc6257>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/rfc/rfc8032>.

[RFC8419] Housley, R., "Use of Edwards-Curve Digital Signature Algorithm (EdDSA) Signatures in the Cryptographic Message Syntax (CMS)", RFC 8419, DOI 10.17487/RFC8419, August 2018, <https://www.rfc-editor.org/rfc/rfc8419>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/rfc/rfc8551>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/rfc/rfc8572>.

[RFC8894] Gutmann, P., "Simple Certificate Enrolment Protocol", RFC 8894, DOI 10.17487/RFC8894, September 2020, <https://www.rfc-editor.org/rfc/rfc8894>.

[Str23] Strenzke, F., "ForgedAttributes: An Existential Forgery Vulnerability of CMS and PKCS#7 Signatures", 22 November 2023, <https://ia.cr/2023/1801>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Daniel Van Geest
CryptoNext Security
Email: daniel.vangeest@cryptonext-security.com

Falko Strenzke
MTG AG
Email: falko.strenzke@mtg.de