Network Working Group
Internet Draft
Intended status: Proposed Standard
Expires: August 2008

P. Sangster
Symantec

February 18, 2008

PA-TNC Security: A Posture Attribute (PA) Security Protocol Compatible with TNC

draft-sangster-nea-pa-tnc-security-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on August 7, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

This document specifies PA-TNC security, a Posture Attribute Security Protocol identical to the Trusted Computing Group's IF-M Security Binding to CMS 1.0 protocol. PA Security offers origin authentication, integrity and optional confidentiality protection for one or more PA attributes. The document then evaluates PA-TNC Security against the requirements defined in the NEA Requirements specification [5].

Conventions used in this document

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 [1].

Table of Contents

1. Introduction

This document specifies PA-TNC security, a Posture Attribute Security Protocol identical to the Trusted Computing Group's IF-M Security Binding to CMS 1.0 protocol [7]. PA Security offers origin authentication, integrity and optional confidentiality protection for one or more PA attributes defined in the PA-TNC specification [6]. The document then evaluates PA-TNC Security capabilities against the requirements defined in the NEA Requirements specification [5].

1.1. Background on Trusted Computing Group

The Trusted Computing Group (TCG) is a consortium that develops specifications for trusted (secure) computing. Since its formation in 2003, TCG has published specifications for a variety of technologies such as: Trusted Platform Module (TPM), TCG Software Stack (TSS), Mobile Trusted Module (MTM), and Trusted Network Connect (TNC).

TCG members include more than 175 organizations that design, build, sell, or use trusted computing technology. Membership is open to any organization that signs the membership agreement and pays the annual membership fee. Non-members are welcome to implement the TCG specifications. Many open source implementers have already done so.

1.2. Background on Trusted Network Connect

Starting in 2004, the TCG has defined and published the Trusted Network Connect (TNC) architecture and standards for network access control. These standards enable multi-vendor interoperability at all points in the architecture and have been widely adopted and deployed.

1.3. Submission of This Document

The IETF has recently chartered the Network Endpoint Assessment (NEA) working group to develop several standards in the same area as TNC. In order to avoid the development of multiple incompatible standards, the TCG is offering several of its TNC standards to the IETF as candidates for standardization in the IETF also. This document is equivalent to TCG's IF-M Security: Bindings to CMS 1.0.

Consistent with IETF's requirements for standards track documents, the TCG has authorized the editors of this document to offer the specification to the IETF without restriction. As with other Internet-Drafts, the IETF Trust owns the copyright to this document. The IETF may modify this document, ignore it, publish it as an RFC, or take any other action. If the IETF decides to adopt a later version of this document as an RFC, the TCG plans to publish a specification for an equivalent TNC protocol to ensure compatibility.

1.4. Prerequisites

This document does not define an architecture or reference model. Instead, it defines a security protocol for protecting PA-TNC attributes consistent with the reference model described in the NEA Requirements specification. The reader is assumed to be thoroughly familiar with the NEA Requirements document particularly those aspects involving PA and its security model. Similarly, the reader should have an understanding of the PA-TNC protocol and its use of attributes. No familiarity with TCG specifications is assumed.

This specification applies and frequently references the Cryptographic Message Syntax (CMS) [3] to a set of one or more PA-TNC attributes in order to protect the attributes from a variety of threats. The readers needs to have a strong working knowledge of CMS and would benefit from a reading of other technologies that have applied CMS for similar purposes such as S/MIME [8].

1.5. Terminology

This document reuses the terminology defined in the NEA Requirements document, PA-TNC internet draft and the CMS specification. No new terminology is introduced by PA-TNC security.

One confusing area of terminology in this document is the overloaded use of the term 'attribute'. The PA-TNC specification defines a set of attributes as type-length-value (TLV) tuples. This specification uses 'attribute' or 'PA-TNC attribute' to refer to the TLV. When a portion of the TLV mentioned it will be described as for example 'attribute type' meaning the PA-TNC attribute's type field.

The other use of the term 'attribute' comes from the CMS specification. A CMS attribute is additional information associated with the CMS content but not included in the data portion of the content field. This specification uses the signedAttrs field in a signed-data to store CMS attributes. Whenever this specification is referring to a CMS oriented attribute (as opposed to a PA-TNC attribute) it will be referred to as 'CMS attribute'.

2. PA-TNC Security Description

2.1. Rationale for Using CMS

CMS was selected to protect the PA-TNC attributes because of its suitability to provide security protections for a messaging oriented protocol. Messaging protocols may wish to avoid a potentially lengthy set of roundtrip message exchanges to setup a security association prior to being able to send protected messages. PA-TNC message senders may only wish to protect one of several attributes exchanged with another party. Such additional roundtrips can cause latency issues that could result in timeouts or other undesirable behavior in some underlying protocols (e.g. 802.1X).

It is envisioned that during a PA-TNC message dialog, several messages might be exchanged that do not need (or require different) security protections. For example, a deployment may not wish to protect messages requesting posture information, but may wish to protect the resulting posture and/or any final decision related attributes. In order to allow for each message's attributes to be protected independently, a more granular security mechanism was required. Note that the use of a protected session oriented protocol, such as TLS, could be provided by the PT protocol.

CMS has been used in the IETF to protect a number of messaging oriented protocols (e.g. MIME messages, firmware upgrades [9]) so it was believed to be a good standards-based approach for protecting PA-TNC message attributes. This specification defines how CMS is applied to PA-TNC to provide origin authentication, integrity and optional confidentiality of one or more attributes. The use of other security protocols is plausible in the future; consequently this protocol ensures that PA-TNC attributes protected by CMS can be easily recognized by Posture Collectors and Posture Validators.

2.2. PA-TNC Attributes Protected by CMS

This section discusses how CMS is used to protect PA-TNC message attributes. The PA-TNC protocol specification defines how Posture Collectors and Posture Validators can exchange messages to perform an assessment. Each message is delivered to interested Posture Collector(s) or Posture Validator(s) based upon the component type (e.g. firewall) indicated in the PB-TNC message type.

Within each PA-TNC message is a set of one or more attributes expressed in TLV format. The attribute type indicates the format and semantics of the attribute's value. PA-TNC defines an extensible attribute type field allowing for both vendor defined and standard attributes to be included and easily identified by PA-TNC message recipients. For more information, see the PA-TNC specification. This specification defines the syntax and semantics of three new attribute types necessary to support CMS protection of PA-TNC attributes. The following subsections will focus on each of the new attributes.

2.3. CMS Protected Content Attribute

The CMS Protected Content attribute allows Posture Collector(s) or Posture Validator(s) to send one or more PA-TNC message attributes protected within a CMS encapsulated object. This specification identifies the profile of CMS's capabilities that are necessary to provide authentication and integrity protection and optionally confidentiality protection for the PA-TNC message attributes. Some aspects of CMS are not required to achieve these security protections, and so for simplicity these are explicitly excluded from the PA-TNC security standard.

Because this specification describes a profile of CMS that directly applies to the protection PA-TNC attributes, it does not attempt to repeat all the encoding and processing rules described by the CMS specification. Nonetheless these encoding and processing rules are required unless explicitly modified or excluded by this specification. The intention behind most of the profiling of CMS in this specification is to exclude portions of CMS or to alter (raise or remove) requirements for particular fields within the CMS structures to reflect their use in protecting PA-TNC attributes.

2.3.1. CMS Content Info and Content Types

Every CMS Protected Content attribute MUST begin with a ContentInfo structure. The ContentInfo structure encapsulates the top level ContentType identifier and the content itself. CMS allows nesting of content types so that other levels of content types may exist within the top level content field. The ContentInfo structure is described in section 3 of the CMS specification and is repeated below for the reader's convenience:

```
ContentInfo ::= SEQUENCE {
      contentType ContentType,
      content [0] EXPLICIT ANY DEFINED BY contentType }
ContentType ::= OBJECT IDENTIFIER
```

Each contentType value is an OID that indicates the syntax and semantics of the associated content field. The CMS specification defines six different contentType values and formats while allowing more to be defined in other specifications. PA-TNC message security protection requires the support of only two contentType values: signed-data and enveloped-data. The signed-data contentType provides origin authentication and integrity protection of the included set of PA-TNC attributes. The signed-data protection MUST be present in all CMS Protected Content attributes.

Optionally, a Posture Collector or Posture Validator may also wish to protect the confidentiality of a signed set of attributes. This can be accomplished by encapsulating the signed-data content within an enveloped-data contentType. The result is an encrypted version of the signed set of attributes being included in the CMS Protected Content. Therefore, all Posture Collectors or Posture Validators supporting the CMS Protected Content attribute MUST be capable of supporting the creation and/or processing of CMS Protected Content attributes containing either:

- signed-data content (signed attributes)
- signed-data content encapsulated within enveloped-data content (signed and encrypted attributes)

Other CMS contentType values MAY be supported but are outside the scope of this specification so are unlikely to offer interoperability. Implementations receiving a CMS Protected Content containing an unrecognized contentType MUST discard the attribute and SHOULD return a CMS Error Code attribute containing an errorCode of badContentType.

2.3.2. CMS Signed-Data

PA-TNC attributes that require authentication and integrity protection MUST use the signed-data CMS content type within a CMS Protected Content PA-TNC message attribute. This section defines the subset of the CMS signed-data features required for protection of PA-TNC message attributes. Readers should refer to section 5 of the CMS specification for background on the

required CMS processing rules that form the basis for the profile discussed in this subsection.

The CMS signed-data content type has the following structure present in the content field of ContentInfo:

```
SignedData ::= SEQUENCE {
     version CMSVersion,
     digestAlgorithms DigestAlgorithmIdentifiers,
     encapContentInfo EncapsulatedContentInfo,
     certificates [0] IMPLICIT CertificateSet,
     crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
     signerInfos SignerInfos }
```

To simplify support and processing of signed CMS protected PA-TNC message attributes, the following restrictions from full CMS are imposed for the signed-data field:

CMSVersion

This field contains a value following the algorithm described in section 5.1 of the CMS specification. This profile does not support certificates and CRLs of type other nor attribute certificates, therefore it is expected that this value will normally be 3 or 1 (depending on the type of SignerIdentity used).

digestAlgorithms

This field SHOULD be empty indicating that recipients need to refer to the signerInfos field to determine the digest algorithm used by the signer. This field MAY contain a single DigestAlgorithmIdentifier OID corresponding to the digest algorithm used during the single signature computation included within the attribute. If present, this field MUST match the signerInfos's digestAlgorithms field described below.

encapContentInfo

This field contains another pair of content type and content (see section 5.2 of the CMS specification for details). The content type (referred to as eContentType) MUST be set to the id-data ContentType OID and the content field MUST only contain one or more PA-TNC message attribute(s) covered by the signature. The content field MUST be present so it is not optional as stated by CMS. The encoding of the PA-TNC message attributes within the content field will match their definition from the PA-TNC specification so does not require DER or BER encoding.

certificates

This field MUST contain the signer's X.509 version 3 identity certificate and SHOULD also contain the set of certificates leading from the signer's certificate to a recipient trusted certificate authority as discussed in the CMS specification. These certificate(s) enable the recipient(s) to perform path validation of the signer's certificate as part of its trust decision. It is expected that the recipient(s) of the message will have other methods for obtaining necessary certificates in the event that this field does not contain a sufficient set of certificates to complete validation. This field SHOULD NOT contain attribute certificates although they are allowable under standard CMS.

crls

No additional restrictions are placed on this field.

signerInfos

A single SignerInfo structure MUST be included in the signerInfos field. Multiple signers MUST NOT be included. PA-TNC security does not support multiple signers so only a single SignerInfo can be present (not a set as described by CMS). The included digestAlgorithm MUST match the value included in the digestAlgorithms field above if one is present.

PA-TNC recipients SHOULD return a CMS Error Code PA-TNC message attribute containing a digestAlgorithmMismatch error code if the signerInfos's digestAlgorithm does not match the specified digestAlgorithm value.

The unsignedAttrs field MUST NOT be used as they are not necessary to meet the requirements of PA-TNC security. The Nonce CMS attribute MUST be included to provide replay protection. Other signedAttrs field MAY be used to include additional supporting information about the protection on the CMS content such as SMIMEEncryptionKeyPreference and SMIMEEncryptionKeyPreference.

2.3.2.1. CMS Signed-Data Example

This section provides a simple example of a PA-TNC message attribute (Request Attribute) encapsulated within a CMS Protected Content attribute. This simple example is intended to help visualize the contents of this attribute and the relationship between the nested CMS ASN.1 structure and the values expected for use with PA-TNC. Due to the encapsulating approach used by CMS, each level of encapsulation is increasingly indented and both the ASN.1 and the encapsulated content example are included.

Initially, each recipient of the example PA-TNC message would receive a message containing a single attribute of type Protected CMS Content. The value portion of the Protected CMS Content TLV contains the following:

```
ContentInfo ::= SEQUENCE {
    contentType ContentType,
    content[0] EXPLICIT ANY DEFINED BY contentType }
```

contentType

```
    id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

content

This field contains the signature metadata and encapsulates the PA-TNC attribute. The ASN.1 for this field is as follows:

```
ContentInfo ::= SEQUENCE {
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifier,
  encapContentInfo EncapsulatedContentInfo,
  certificates [0] IMPLICIT CertificateSet OPTIONAL,
  signerInfos SignerInfos }
```

version

Set to 1

digestAlgorithms

Empty (0 length field)

encapContentInfo

This field contains the encapsulated PA-TNC attribute that was signed. The ASN.1 for this field is as follows:

```
ContentInfo ::= SEQUENCE {
   contentType ContentType,
   content[0] EXPLICIT ANY DEFINED BY contentType }
```

contentType

```
id-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1 }
```

content

This field contains the PA-TNC message attribute(s) included in the signature. For this example, it contains the Request Attribute TLV as defined by the PA-TNC specification.

certificates

List of X.509 certificates including the sender's certificate and any parent CA certificates leading to a root trusted by the sender.

crls

Revocation information for signer's certificate

signerInfos

One set of signer information including signer's identity and algorithms used in the signature. This field can also carry a set of signed and unsigned CMS attributes. For this example, the SignerInfo instance uses issuerAndSerialNumber to denote the signer's certificate. The Nonce signed CMS attribute is included for replay protection. No unsigned attributes are included.

2.3.2.2. Signed-Data Required Algorithms

In order to enable interoperability between independent implementations, this subsection defines the algorithms that PA-TNC security compliant implementations are expected to support. Additional algorithms and key lengths MAY be supported.

Purpose	Algorithm (Key Len.)	Requirement Level	Algorithm Reference
Digest Algorithm	SHA-1 (160)	MUST (Treat as MUST-)	RFC 3370, Sec. 2.1 FIPS 180-1
	SHA-256 (256)	MUST	IETF I-D [4] FIPS 180-2
Signature Algorithm	RSA (2048)	MUST	RFC 3370, Sec. 3.2 PKCS #1 v1.5
	ECDSA (256)	SHOULD	RFC 5008, Sec. 3 FIPS 186-2

2.3.3. CMS Enveloped-Data

PA-TNC attributes that require confidentiality protection MUST use the enveloped-data CMS content type to encapsulate and encrypt the signed-data content. PA-TNC security does not try to provide a confidentiality only security service, and therefore enveloped-data is used only in conjunction with signed-data (authentication and integrity protected) content. This subsection defines the subset of the CMS enveloped-data features required for the protection of PA-TNC message attributes already protected within a signed-data object. When a feature isn't specifically excluded or restricted by this

specification, implementations MUST follow the processing rules defined in the CMS specification.

The CMS enveloped-data content type is defined in section 6.1 of the CMS specification as having the following structure:

```
EnvelopedData ::= SEQUENCE {
   version CMSVersion,
   originatorInfo [0] IMPLICIT OriginatorInfo,
   recipientInfos RecipientInfos,
   encryptedContentInfo EncryptedContentInfo,
   unprotectedAttrs [1] IMPLICIT UnprotectedAttributes
       OPTIONAL }
```

To simplify support and processing of encrypted CMS protected PA-TNC message attributes, the following restrictions from full CMS are imposed on the encapsulated-data field:

CMSVersion

This field contains a value derived from the algorithm described in section 5.1 of the CMS specification. Because this profile does not use certificates and CRLs of type other and unprotected attributes MUST NOT be used, it is expected that this value will normally be 0.

originatorInfo

This field MUST contain the signer's X.509 version 3 identity certificate and SHOULD also contain the set of certificates leading from the signer's certificate to a recipient trusted certificate authority as discussed in the CMS specification. These certificates enable the recipient(s) to perform path validation of the signer's identity certificate. It is expected that the recipient(s) of the message will have other ways to obtain necessary certificates in the event that this field does not contain a sufficient set of certificates to complete validation. This field SHOULD NOT contain attribute certificates despite being allowable under standard CMS.

Optionally this field may also include CRL information used to check the validity of the certificates presented by the originator. This specification does not change the CMS specification handling of CRLs.

recipientInfos

This field contains a set of per recipient information necessary to process the encrypted content. This field contains the encrypted key destined for each recipient to be used to decrypt the encryptedContentInfo. This specification does not change the CMS processing of this field so readers should refer to section 6.2 of the CMS specification for details and information about handling of different key management techniques.

encryptedContentInfo

This field contains the encrypted version of the signed-data content together with information about the encryption algorithm used.

The use of the encryptedContentInfo field is the same as specified in CMS except that this field MUST NOT be empty and MUST contain the encrypted signed-data (in an id-data content type). This field MUST NOT contain content that is not signed as it could be subject to undetectable integrity based attacks.

unprotectedAttrs

The CMS specification defines this field as optional. For PA-TNC security, this field MUST NOT be used.

2.3.3.1. CMS Enveloped-Data Example

This section shows an example of an encrypted and signed set of PA-TNC message attributes. Rather than duplicating the signed-data example from section 2.3.2.1. the example focuses on the encrypted-data content and highlights where the signed-data is included.

Initially each recipient of the example PA-TNC message would receive a message containing a single attribute of type Protected CMS Content. The value portion of the Protected CMS Content TLV would contain the following:

```
ContentInfo ::= SEQUENCE {
   contentType ContentType,
   content[0] EXPLICIT ANY DEFINED BY contentType }
```

contentType

```
id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-
body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }
```

content

This field contains the encrypted content and information required to decrypt it on a per recipient basis. The ASN.1 for this field is as follows:

```
ContentInfo ::= SEQUENCE {
   version CMSVersion,
   originatorInfo [0] IMPLICIT OriginatorInfo,
   recipientInfos RecipientInfos,
   encryptedContentInfo EncryptedContentInfo,
   unprotectedAttrs [1] IMPLICIT UnprotectedAttributes
       OPTIONAL }
```

version

Set to 0

originatorInfo

This field includes a list of X.509 certificates including the signer's certificate and potentially several parent CA certificates enabling the recipient to complete chain validation.

recipientInfos

This field contains encrypted versions of the keys associated with each recipient that are used to decrypt the encryptedContentInfo's content. Normally it is expected that a single recipient will be involved with a CMS protected message so this includes only one recipientInfo.

encryptedContentInfo

This field contains the encapsulated PA-TNC attribute that was encrypted. The ASN.1 for this field is as follows:

```
ContentInfo ::= SEQUENCE {
    contentType ContentType,
    contentEncryptionAlgorithm
        ContentEncryptionAlgorithmIdentifier,
    encryptedContent [0] IMPLICIT EncryptedContent }
```

contentType

```
    id-signedData OBJECT IDENTIFIER ::= { iso(1) member-
    body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

contentEncryptionAlgorithm

```
    joint-iso-itu-t(2) country(16) us(840)
    organization(1) gov(101) csor(3)_ nistAlgorithms(4)
    aes(1) aes128(2)
```

encryptedContent

The content of this field is an encrypted version of the signed-data example described in section 2.3.2.1. While this field is optional in CMS, it is required for PA-TNC security (external signatures are not supported).

unprotectedAttrs

This field is empty.

2.3.3.2. Enveloped-Data Required Key Management

This subsection discusses the required key management schemes as defined by the CMS specification. The key management scheme is used to establish a key that is shared by the communicating parties enabling them to perform cryptographic operations on their communications. For this specification, the exchanged content is signed-data containing one or more PA-TNC message attributes protected by a signature. In order to allow the end parties to use different types of credentials to protect this key negotiation, several key management schemes are defined. This specification follows the requirements from section 6.2 of the CMS specification which states:

> "Implementations MUST support key transport, key agreement, and previously distributed symmetric key-encryption keys, as represented by ktri, kari, and kekri, respectively. Implementations MAY support the password-based key management as represented by pwri. Implementations MAY

support any other key management technique as represented by ori."

2.3.3.3. Enveloped-Data Required Algorithms

In order to enable interoperability between independent implementations, this subsection defines the key management and content protection algorithms that PA-TNC security compliant implementations are expected to support. Additional algorithms, key lengths and key management techniques MAY be supported. The password-based key management scheme MAY be supported while the key transport, key agreement and previously distributed symmetric KEK schemes MUST be supported by compliant implementations.

Key Management Scheme	Algorithm (Key Length)	Reqmt Level	Algorithm Reference
Key Transport	RSA wrap AES CEK (2048)	MUST	RFC 3565, Sec. 2.2
Key Agreement	ESDH w/AES KEK (128 & 256)	MUST	RFC 3565, Sec. 2.3
Prev Distributed Symmetric KEK	AES Key Wrap (128 & 256)	MUST	RFC 3565, Sec. 2.4
Password Based	Passwd derived AES(128 & 256)	MUST (*)	RFC 3565, Sec. 2.5

* - Optional to implement, so mandatory if supported

The above described key management schemes are used to establish a symmetric content encryption key that protects the signed PA-TNC attributes. PA-TNC security compliant implementations MUST support the use of the following algorithms for content encryption:

Purpose	Algorithm (Key Length)	Reqmt Level	Algorithm Reference
Content Encryption	AES (128 & 256)	MUST	RFC 3565, Sec. 2.1

2.4. Security Capabilities Attribute

The Security Capabilities attribute type allows a Posture Collector or Posture Validator to determine the supported set of cryptographic algorithms supported by the recipient(s) prior to creating a protected message. This provides a simple cryptographic algorithm discovery mechanism to assist the sender's selection of an algorithm consistent with the sender's policy and supported by the recipient. The algorithm list is encapsulated within a signed CMS message that the recipient can use to verify the authenticity and integrity of the algorithm list. If confidentiality protection of the Security Capability attribute is desired, the sender can encapsulate it within an enveloped-data content type. Note however that the sender of this attribute will not be aware of the cryptographic algorithms supported by the recipient (since it is replying to a cryptographic discovery request). For this reason, implementations MAY support encryption of the security capabilities content using the enveloped-data; however, one of the mandatory encryption algorithms SHOULD be used to maximize the possibility that the recipient supports the algorithm.

In order for a PA-TNC message sender to determine the security capabilities supported by recipient(s), the Posture Collector or Posture Validator would include the Security Capabilities attribute type in a PA-TNC Attribute Request attribute (see the PA-TNC specification for details). The Attribute Request may include other PA-TNC attribute types in the list if appropriate. The recipient(s) of the Attribute Request attribute containing the Security Capabilities attribute type respond with the Security Capabilities attribute described in this section. NOTE that typically a Posture Collector does not send an Attribute Request attribute to a Posture Validator

during an assessment as it normally is responding to requests for attributes. However if an Posture Collector wishes to determine the security algorithms supported by recipient Posture Validator(s), it may send a Request Attribute containing only the Security Capabilities attribute type. The PA-TNC Security Capabilities attribute MUST consist of a single CMS signed-data content containing a single signed attribute in the signerInfo and an empty eContent (no other data) within the encapContentInfo. The Security Capabilities attribute SHOULD be signed with a mandatory signature algorithm (specified in section 2.3.2.2.) to ensure that the recipient will be able to verify the signature. Note that CMS signature also includes a field called signed attribute (signedAttrs) that is information outside of the CMS content. This specification does not use unsigned CMS attributes but does use the signed CMS attribute to convey the supported security algorithms. For PA-TNC security, the attributes described in the PA-TNC specification are present in the data portion (eContent in encapContentInfo) of the CMS content; whereas the CMS defined attributes exist outside of the eContent section, such as in the signerInfos field, and are represented by ASN.1 in this specification.

This specification defines a CMS attribute called paTncSecurityCapabilities that contains a prioritized list of the cryptographic algorithms supported for various purposes by the sender. The purpose of each algorithm is reflected by the OID definition and can include: signing, data encryption, key wrapping and digesting. The prioritized algorithm list MUST be grouped according to the algorithms' purpose to ease processing by the recipient. The CMS paTncSecurityCapabilities attribute is based on the SMIMECapabilities attribute defined in section 2.5.2 of the SMIME specification. The processing rules for the SMIMECapabilities CMS attribute apply to the paTncSecurityCapabilities CMS attribute unless stated otherwise in this section.

2.4.1. paTncSecurityCapabilities Within Signed-Data

The paTncSecurityCapabilities CMS attribute exists within the signed-data content type described in section 2.3.2. of this specification. Rather than repeating all the detail of the signed-data section, this section will focus on the differences between a signed set of PA-TNC attributes and a paTNCSecurityCapability CMS attribute encapsulated within signed-data content.

The paTncSecurityCapabilities CMS attribute is present in the signedAttrs field; consequently it is included in the CMS signature. This enables recipients to detect modification of the sender's claimed security capabilities and to authenticate the sender's identity. Unlike normal signed-data content, the paTncSecurityCapabilities CMS attribute MUST exist in all Security Capabilities attributes and MUST be the only CMS attribute present in the signedAttrs list besides the required Nonce CMS (replay protection) attribute. This differs from normal signed-data content that is allowed to include other CMS attributes.

Another difference concerns the use of the encapContentInfo's eContent field. In the case of signed-data content, this field normally includes the PA-TNC message attributes being protected. For a Security Capabilities attribute, the eContent field MUST be empty. This is because the sole purpose of this attribute is to indicate the security capabilities of the sender and those capabilities are included in the signedAttrs field. No other PA-TNC message attributes are allowed to be encapsulated in this attribute. Note that a PA-TNC message can contain several attributes so other attributes could be sent in addition to the Security Capabilities attribute.

2.4.2. paTncSecurityCapabilities ASN.1

The paTncSecurityCapabilities content mirrors the SMIMECapabilities attribute as described in section 2.5.2 of the SMIME specification. The ASN.1 defined for the paTncSecurityCapabilities attribute is as follows:

```
paTncSecurityCapability ::= SEQUENCE {
   capabilityID OBJECT IDENTIFIER,
   parameters ANY DEFINED BY capabilityID OPTIONAL }

paTncSecurityCapabilities ::= SEQUENCE of
   paTncSecurityCapability
```

The paTncSecurityCapabilities CMS attribute is simply a prioritized (preference order) list of OIDs and associated cryptographic parameters of the algorithms supported by the sender. Ordering the list by preference provides another piece of information to those wishing to send protected information to the sender. This specification leverages the CMS Algorithms specification defined set of ASN.1 for the OIDs and parameters for the security algorithms represented in this list. For more

information see section 7 of the CMS Algorithm specification and the AES algorithm specification. An in-depth discussion of the SMIMECapabilities CMS attribute that parallels the paTncSecurityCapabilities CMS attribute is included in the SMIME specification in section 2.5.2

2.5. CMS Error Code Attribute

This PA-TNC attribute allows a recipient of an invalid security protected PA-TNC message to send an integrity protected error response indicating the reason for the failure. In order to return protected error information related to the processing of CMS Protected Content attributes, PA-TNC security encapsulates the error code within of a signed attribute, itself encapsulated within CMS signed-data content. Using a signed attribute allows recipients to verify the integrity and origin authentication of error status preventing spoofing and other related attacks. In some uncommon situations, recipients may not be able to verify the signature (e.g. the use of an unsupported digest algorithm) or establish trust in the sender (e.g. no common trust anchor) but at least the recipient can view the returned error code and decide whether to trust it and therefore how to act on it. Care should be taken when trusting information whose integrity can not be verified as it could leave the recipient open to various attacks.

All CMS processing errors MUST result in a response PA-TNC message containing a CMS Error Code Attribute. The CMS Error Code attribute MUST only contain a single CMS ContentInfo of content type signed-data. The Signed-Data element MUST contain an empty eContent and include only the Nonce CMS attribute and the paTncErrorCode CMS attribute in the signerInfo.

The CMS Error Code attribute SHOULD be signed with one of the mandatory signature algorithms (specified in section 2.3.2.2.) to ensure that the recipient will be able to verify the signature.

2.5.1. paTncErrorCode Within Signed-Data

The paTncErrorCode CMS attribute exists within the signed attributes portion of the signed-data content type. Rather than repeat all the detail of section 2.3.2. this section will describe the differences between a signed set of PA-TNC attributes and a paTNCSecurityErrorCode CMS attribute housed within signed-data content. Note that the CMS Error Code

attribute and the Security Capabilities attribute both use the same CMS fields.

The paTncErrorCode CMS attribute is an attribute that is present in the signedAttrs field so it is included in the CMS signature. This enables recipients to detect modification of the error information and to authenticate the sender's identity. Unlike normal signed-data content, the paTncErrorCode CMS attribute MUST exist in all CMS Error Code attributes and MUST be the only CMS attribute present in the signedAttrs list besides the required Nonce CMS (replay protection) attribute. This differs from normal signed-data content that is allowed to include other CMS attributes.

The other difference is the use of the encapContentInfo's eContent field. Normally in signed-data content, this field must include the PA-TNC message attributes being protected. For a CMS Error Code attribute, the eContent field MUST be empty. This is because the sole purpose of this attribute is to carry the error code related to an earlier PA-TNC message to the recipient and the error information is included in the signedAttrs field. No other PA-TNC message attributes (e.g. Request Attribute) are allowed to be encapsulated in this attribute. Note that a PA-TNC message can contain several attributes so other attributes could be sent in addition to the CMS Error Code attribute.

2.5.2. paTncErrorCode ASN.1

The paTncErrorCode CMS attribute is placed in the signerInfo's signedAttrs field. The signedAttrs field is included in the signature applied so that the recipient can verify the authenticity and integrity of the information before taking action. The following describes the syntax and semantics of the paTncErrorCode CMS attribute.

```
paTncErrorCode ::= SEQUENCE {
   vendorID OBJECT IDENTIFIER,
   status errorCode,
   ContentInfo originalContent OPTIONAL }
```

vendorID

This field indicates the Private Enterprise Number (PEN) OID as a Vendor ID [10] of the party who owns the errorCode name space that is being used in the errorCode field. For example, this value for Symantec would be iso(1) org(3) dod(6) internet(1) private(4) enterprise(1) symantec(393). This allows vendors to have vendor-defined error codes outside of the standard name space. For IETF standard PA-TNC security errors, the vendorID field MUST be set to zero.

errorCode

This field MUST contain the error code reflecting the error that occurred while processing the CMS message. The IETF standard error codes are listed in section 2.5.3.

originalContent

This field SHOULD contain the contents of the CMS content that cause the error. If the original content is large and the deployment is bandwidth constrained this field MAY be empty.

2.5.3. IETF Standard paTncErrorCode Values

This section defines an initial set of IETF standard PA-TNC security error code values. IANA maintains a registry of IETF PA-TNC standard error codes. Entries may only be added to this registry by IETF Consensus. That is, they MUST be defined in an RFC approved by the IESG.

The PA-TNC Security error codes MUST always be used with a vendorID field value of zero. The following table briefly describes the initial set of the IETF standard error codes used in the errorCode field of a paTncErrorCode value. Values not defined in this table MUST NOT be used with an IETF (zero) vendorID unless approved and included in the IANA paTNCSecurityErrorCode registry.

Posture Collectors and Posture Validators MUST NOT require support for particular vendor-specific PA-TNC Error Code and MUST interoperate with other parties despite any differences in the set of vendor-specific PA-TNC errorCode values supported. This ensures interoperability while allowing for vendor experimentation and additional functionality outside of the IETF standard name space.

Implementations MUST use their organization's assigned PEN OID in the vendorID to include non-IETF standard error codes. The following error codes were initially based on early work using CMS for Trust Anchor Management Protocol (TAMP) [12].

Value	Name	Description
0	Reserved	This value MUST NOT be used
1	decodeFailure	Unable to decode content, doesn't match provided type
2	badContentInfo	Unknown or invalid ContentInfo syntax used in content
3	badSignedData	Unknown, invalid or non-compliant signed-data format
4	badEnvelopedData	Unknown or non-compliant enveloped-data format
5	badCertificate	Invalid syntax used for included certificate(s)
6	badSignerInfo	Invalid or unsupported SignerInfo syntax
7	badSignedAttrs	Invalid or unsupported use of signed attributes
8	badUnsignedAttrs	Non-compliant use of unsigned attributes
9	missingContent	Non-compliant empty eContent field in signed-data
10	noTrustAnchor	Lack of trust anchor associated with the signer's certificate
11	notAuthorized	Requestor's not authorized to perform operation
12	badDigestAlgorithm	Digest algorithm used is unsupported or unknown
13	badSignatureAlgorithm	Signature algorithm used is unknown or unsupported
14	unsupportedKeySize	Key used is an unsupported length (too short or long)
15	unsupportedParameters	Algorithm parameters indicate unsupported values
16	signatureFailure	Recipient computed signature does not match provided
17	decryptionFailure	Unable to decrypt content using provided content decryption key
18	keyManageFailure	Unable to determine provided content encryption key
19	badKeyManage	Unknown or unsupported key management technique used
20	nonceMissing	Received signed content lacking required nonce attribute
21	invalidNonce	Unexpected or invalid nonce received
22	repeatedNonce	Received nonce was recently used so possible replay
23	nonceOrdering	Received nonce was not one greater then prior value
24	badContentType	Invalid or unsupported contentType found
25	digestAlgMismatch	Different algorithms in signerInfos &

		digestAlgorithm
29	missingSignature	Signed-data missing required signature

30	resourcesBusy	Recipient lacks resources to process received content
31	versionNumberMismatch	Version in received message was unsupported
33	revokedCertificate	Certificate used was revoked by issuer
65535	other	Unable to process message for reason other than above

2.6. Nonce CMS Attribute

Unlike the above three PA-TNC attributes, this attribute is a CMS attribute that is located in the signedAttrs field of the signed-data content within other PA-TNC security protected attributes. For example a CMS Protected Content attribute would include a Nonce CMS attribute in its signedAttrs field to detect replay attacks.

The Nonce CMS attribute allows the sender of a PA-TNC security protected attribute to include a nonce that can be used by the recipient to detect a replay attack. The Nonce CMS attribute MUST be used in all PA-TNC security messages as defined within this specification. The Nonce CMS attribute is a signed attribute that MUST exist within any signed-data content type including the Security Capabilities, CMS Error Code, and CMS Protected Content PA-TNC attributes. The Nonce CMS attribute MUST NOT be used in the enveloped-data content type to simplify processing of such messages because the enveloped-data will encapsulate signed-data content that must include the nonce anyway.

The value of the nonce MUST be unpredictable to third parties so MUST NOT be based on network observable information. Use of good sources of entropy is highly desired, however implementations may use persistently stored sequence numbers that do not repeat (even across reboots and other disruptive events). The Nonce CMS attribute contains two separate values each under the control of a Posture Collector or Posture Validator. This allows both sides of the message exchange to provide entropy and receive replay protection.

The initial sender of a CMS message generates its nonce and includes it in the Nonce CMS attribute with a zero value for the other party. When initially responding to a CMS protected message containing a zero value nonce, the responder generates its nonce and includes it in the reply together with a copy of the nonce sent by the other party. If the initial sender wishes to send another signed-data message to the other party it creates a Nonce CMS attribute by copying the other party's

nonce and by incrementing its own nonce by one. If the resulting value is 2^32 then it should randomly generate a new

nonce. This process continues until the completion of an assessment. Implementations unable to generate a good nonce value MAY use persistent sequence numbers providing that it can ensure that no repeated values are used in a predictable manner.

When a CMS message recipient receives a message, it must check the message's nonce attribute to ensure that its nonce matches the value of the nonce that it sent or contains a zero. Similarly it must also check that the nonce created by its peer is one greater than the last received assessment message nonce if it is not the first CMS protected message of the assessment.

2.6.1. paTncNonce Within Signed-Data

The Nonce CMS attribute MUST be present in the signedAttrs list in all CMS signed-data content used by PA-TNC security. Recipients of CMS signed-data protected attributes lacking a Nonce CMS attribute MUST return an error to the sender and MUST NOT process the CMS message. The Nonce CMS attribute is defined as paTncNonce below.

As mentioned above, the Nonce CMS attribute (paTncNonce) only exists within the CMS signed-data's list of signed attributes (signedAttrs) and does not require changes to other fields within the signed-data content. This allows paTncNonce to be included in signed-data content that carries data (e.g. CMS Protected Content) or is empty (e.g. Security Capabilities).

2.6.2. paTncNonce CMS Attribute ASN.1

The following ASN.1 shows the syntax of the paTncNonce CMS attribute that is included in the signed-data content's signedAttrs field.

```
NonceType ::= INTEGER (0 .. 4294967295)

paTncNonce ::= SEQUENCE {
   pcNonce NonceType,       -- Posture Collector's nonce
   pvNonce NonceType }      -- Posture Validator's nonce
```

pcNonce

This field contains an unpredictable 32 bit unsigned integer of the Posture Collector's choosing. The selection of this value MUST be consistent with the following rules: Initial value during assessment:

- o If a Posture Collector is sending an initial CMS protected attribute during an assessment, the Posture Collector MUST select an unpredictable, non-zero nonce value for this field.
- o If a Posture Validator is sending an initial CMS protected attribute during an assessment, the Posture Validator MUST set this field to zero. Zero indicates that a Posture Collector has not yet had an opportunity to establish an initial nonce value.

Non-initial value during assessment:

- o Posture Collector MUST increment by one the prior pcNonce value used during this assessment and if <2^32 include this value in this field. If 2^32 is reached, a new unpredictable, non-zero value MUST be selected. The selected value SHOULD be compared against a list of those recently used to avoid causing the recipients to consider this a replay and sending an error. Use of the prior pcNonce + one approach to new nonce selection was done to ease nonce create and replay table maintenance.
- o Posture Validator MUST copy pcNonce from most recent valid CMS protected message from Posture Collector during this assessment
- o Recipients MUST verify appropriate nonce used in both fields to detect replay attempts. Recipients SHOULD maintain a table of recently used nonce ranges for each peer.

pvNonce

This field contains an unpredictable 32 bit unsigned integer of the Posture Validator's choosing. The selection of this value MUST be consistent with the following rules:

Initial value during assessment:

- o If Posture Validator is sending the initial CMS protected attribute during an assessment, the Posture Validator MUST create an unpredictable, non-zero nonce value for this field.
- o If Posture Collector is sending the initial CMS protected attribute during an assessment, the Posture Collector MUST set this field to zero. Zero indicates that the

Posture Validator has not yet had an opportunity to establish an initial nonce value.

Non-initial value during assessment:

- Posture Validator MUST increment the prior pcNonce value used during this assessment and if <2^32 include the value in this field. If 2^32 reached, a new unpredictable, non-zero value MUST be selected. The selected value SHOULD be compared against a list of those recently used to avoid causing the recipients for considering this a replay and sending an error.
- Posture Collector MUST copy pcNonce from most recent valid CMS protected message from the Posture Validator during this assessment.

Recipients MUST verify appropriate nonce used in both fields to detect replay attempts. Recipients SHOULD maintain a table of recently used nonce ranges for each peer.

2.6.3. paTncNonce CMS Attribute Example

This section provides a simple example of a nonce value exchange. In this example, a single Posture Collector and Posture Validator will participate in a two roundtrip exchange including three CMS protected attribute messages. The sub-bullet in each step describes the contents of the pcNonce and pvNonce fields.

1. Posture Validator sends an unprotected PA-TNC Request Attribute containing the Security Capabilities attribute type
   - No nonces involved with this message (unprotected).
2. Posture Collector responds with a PA-TNC Security Capabilities attribute
   - pcNonce = initial value X; pvNonce = 0
3. Posture Validator sends a PA-TNC CMS Protected Content attribute containing a PA-TNC Request Attribute requesting Product Information about endpoint's operating system
   - Verify X was not recently used by Posture Collector
   - pcNonce = X; pvNonce = initial value Y

4. Posture Collector responds with a PA-TNC Product Information attribute encapsulated within a CMS Protected Content attribute
   - o Verify Y was not recently used by Posture Validator
   - o pcNonce = X+1; pvNonce = Y
5. Posture Validator sends an assessment result in a CMS Protected Content attribute
   - o Verify pcNonce is last nonce + 1
   - o pcNonce = X+1; pvNonce = Y+1

Note that this example does not involve X or Y reaching 2^32 so no new unpredictable values were required. If this was required the recipient would need to verify that the last nonce value was 2^32-1 and the new value had not been used recently. Using this algorithm both parties can detect replayed messages from the other party (or an attacking imposter). One further benefit is that the loss of a message during a CMS exchange can be detected by the recipient who can respond to this failure by sending an error message (nonceOrdering) to the sender, who could resend the prior message if appropriate.

3. Evaluation Against NEA Requirements

This section evaluated the PA-TNC security protocol against the requirements defined in the NEA Requirements document. Each subsection considers a separate requirement from the NEA Requirements document. Only common requirements (C-1 through C-10) and PA security oriented requirements are considered, since these are the only ones that apply to PA security.

3.1. Evaluation Against Requirement C-1

Requirement C-1 says:

C-1 NEA protocols MUST support multiple round trips between the NEA Client and NEA Server in a single assessment.

PA-TNC security meets this requirement fully. It allows an unlimited number of round trips between the NEA Client and NEA Server.

3.2. Evaluation Against Requirement C-2

Requirement C-2 says:

> C-2 NEA protocols SHOULD provide a way for both the NEA Client and the NEA Server to initiate a posture assessment or reassessment as needed.

PA-TNC security meets this requirement. Either the NEA Client or the NEA Server can initiate a posture assessment or reassessment as PA security is independent of the assessment initiation process and allows either party to send any of the protected attributes.

3.3. Evaluation Against Requirement C-3

Requirement C-3 says:

> C-3 NEA protocols including security capabilities MUST be capable of protecting against active and passive attacks by intermediaries and endpoints including prevention from replay based attacks.

PA-TNC security provides cryptographic protection for one or more PA-TNC attributes. This protection includes strong authentication of attribute sender's identity, the integrity of the attribute information sent and optionally the confidentiality of the integrity protected attributes. PA-TNC security also includes nonce-based detection of replayed attributes so even active intermediaries are unable to inject, modify or replay attributes observed on the network.

3.4. Evaluation Against Requirement C-4

Requirement C-4 says:

> C-4 The PA and PB protocols MUST be capable of operating over any PT protocol. For example, the PB protocol must provide a transport independent interface allowing the PA protocol to operate without change across a variety of network protocol environments (e.g. EAP/802.1X, PANA, TLS and IKE/IPsec).

PA-TNC security meets this requirement. PA-TNC security has no dependencies or interactions with the underlying PB or PT protocols. PA-TNC security protocol should be able to operate over any protocol that PA-TNC can use.

3.5. Evaluation Against Requirement C-5

Requirement C-5 says:

> C-5 The selection process for NEA protocols MUST evaluate and prefer the reuse of existing open standards that meet the requirements before defining new ones. The goal of NEA is not to create additional alternative protocols where acceptable solutions already exist.

Based on this requirement, PA-TNC security should receive a strong preference. PA-TNC security is equivalent with IF-M Security 1.0, an open TCG specification. IF-M is the attribute exchange protocol for the existing TCG architecture that has been implemented by a number of open source projects and commercial vendors.

3.6. Evaluation Against Requirement C-6

Requirement C-6 says:

> C-6 NEA protocols MUST be highly scalable; the protocols MUST support many Posture Collectors on a large number of NEA Clients to be assessed by numerous Posture Validators residing on multiple NEA Servers.

PA-TNC security meets this requirement. PA-TNC security is capable of include many PA-TNC attributes within a single CMS content or can be repeatedly used to individually protect any number of PA-TNC attributes within one or more PA-TNC messages. PA-TNC security is independent of per Posture Collector or Posture Validator information so scales very well to large deployments. The one exception is that a sender of CMS protected information may include per-recipient content decryption keys using an extensible set of key management techniques. The number of recipients can be extremely large before the CMS limit is reached, but even in this unlikely situation the sender could still send multiple separate copies of the protected attribute in a PA-TNC message each to a different set of recipients.

3.7. Evaluation Against Requirement C-7

Requirement C-7 says:

C-7 The protocols MUST support efficient transport of a large number of attribute messages between the NEA Client and the NEA Server.

PA-TNC security meets this requirement. The use of CMS allows for an efficient encoding of many PA-TNC attributes and the associated security meta-data (signatures, algorithms etc.) inside a PA-TNC attribute. Many of the PA-TNC attributes can be combined in a PA-TNC message and because the protocol supports multiple round trips, several related PA-TNC messages can be sent in one or more PB-TNC batches between the Posture Collector(s) and Posture Validator(s).

3.8. Evaluation Against Requirement C-8

Requirement C-8 says:

C-8 NEA protocols MUST operate efficiently over low bandwidth or high latency links.

PA-TNC security meets this requirement. A minimal CMS signed-data content adds minimal overhead to the encapsulated attributes so is efficient even over low bandwidth links. This specification carefully profiled full CMS so we only include those portions of CMS that are required to meet NEA's functional requirements.

3.9. Evaluation Against Requirement C-9

Requirement C-9 says:

C-9 For any strings intended for display to a user, the protocols MUST support adapting these strings to the user's language preferences.

PA-TNC security meets this requirement. The PA-TNC security protocol does not explicitly introduce strings destined for the user.

3.10. Evaluation Against Requirement C-10

Requirement C-10 says:

C-10 NEA protocols MUST support encoding of strings in UTF-8 format.

PA-TNC security meets this requirement. The PA-TNC security protocol does not use strings.

3.11. Evaluation Against Requirement PA-1

Requirement PA-1 says:

PA-1 The PA protocol MUST support communication of an extensible set of NEA standards defined attributes. These attributes will be uniquely identifiable from non-standard attributes.

PA-TNC security meets this requirement. The PA-TNC security protocol blindly encapsulates the PA-TNC attributes so is unaware of which attributes are present. PA-TNC security uses CMS ContentType identifiers to uniquely identify its internal extensible set of attributes.

3.12. Evaluation Against Requirement PA-2

Requirement PA-2 says:

PA-2 The PA protocol MUST support communication of an extensible set of vendor-specific attributes. These attributes will be segmented into uniquely identifiable vendor specific name spaces.

PA-TNC security meets this requirement. The PA-TNC security protocol blindly encapsulates the PA-TNC attributes so is unaware of which attributes are present. The PA-TNC security protocol leverages OIDs to allow for vendor defined name spaces and to allow extensibility for new types of CMS attribute, algorithms and other types.

3.13. Evaluation Against Requirement PA-3

Requirement PA-3 says:

PA-3 The PA protocol MUST enable a Posture Validator to make one or more requests for attributes from a Posture Collector within a single assessment. This enables the Posture Validator to reassess the posture of a particular endpoint feature or to request additional posture including from other parts of the endpoint.

PA-TNC security meets this requirement. The PA-TNC security protocol allows for multiple roundtrips and does not get

involved in deciding when an assessment is complete. This allows the Posture Validator and Posture Collector to decide when sufficient information has been exchanged using the base PA-TNC protocol.

3.14. Evaluation Against Requirement PA-4

Requirement PA-4 says:

> PA-4 The PA protocol MUST be capable of returning attributes from a Posture Validator to a Posture Collector. For example, this might enable the Posture Collector to learn the specific reason for a failed assessment and to aid in remediation and notification of the system owner.

PA-TNC security meets this requirement. The PA-TNC security protocol allows for multiple roundtrips and does not get involved in deciding when an assessment is complete. Therefore the PA-TNC security protocol does not constrain when Posture Validators may send PA-TNC messages.

3.15. Evaluation Against Requirement PA-5

Requirement PA-5 says:

> PA-5 The PA protocol SHOULD provide authentication, integrity, and confidentiality of attributes communicated between a Posture Collector and Posture Validator. This enables end-to-end security across a NEA deployment that might involve traversal of several systems or trust boundaries.

PA-TNC security meets this requirement. This requirement is the primary reason a PA-TNC security protocol was defined. PA-TNC security protocol provides cryptographic authentication of the attribute sender, integrity protection of the attribute contents and optional confidentiality of attributes between Posture Collector(s) and Posture Validator(s). This protection is provided end to end so even if the PT security protections are terminated prior to reaching the Posture Validator, the PA-TNC protections will remain. This allows for PA-TNC security protected attributes to be transported over unprotected communication channels spanning multiple trust boundaries.

3.16. Evaluation Against Requirement PA-6

Requirement PA-6 says:

PA-6 The PA protocol MUST be capable of carrying attributes that contain non-binary and binary data including encrypted content.

PA-TNC security meets this requirement fully. The PA-TNC security protocol encapsulates PA-TNC attributes and is unaware of their contents. The PA-TNC security protocol is able to transport binary and non-binary attributes as it does not impose any sort of PA-TNC attribute encoding or transport that would alter the attributes original content.

4. Security Considerations

This section discusses how the security countermeasures provided by the PA-TNC security protocol address the threats to PA-TNC messages discussed in the security considerations section of the PA-TNC specification. This section also discusses some additional potential threats specific to the use of CMS to protect the PA-TNC protocol.

4.1. Countermeasures to PA-TNC Threats

The PA-TNC specification discusses a range of potential threats to the PA-TNC protocol and its attributes. Some deployment environments may have mitigating controls already in place on the network or have a threat model that accepts the identified risks. For example, many deployments may deploy cryptographically protected IF-T protocols and trust the NEA Client and NEA Server not to compromise the attributes exchanged. For deployments that require security protection of the attributes sent between the Posture Collectors and Posture Validators, the following sections discuss how the use of CMS can provide the necessary protection.

The following subsections are organized along the capabilities of CMS protected PA-TNC attributes. This allows a single discussion of the cryptographic protection provided by the countermeasure and a summary of the threats addressed by the countermeasure. The PA-TNC security leverages the signed-data and enveloped-data content types to provide different levels of protection for one or more attributes. The following subsections discuss how each content type's protections address the PA-TNC threats.

4.1.1. Threats Addressed by Signed Attributes

The signed-data content type of CMS provides a cryptographic signature around the set of one or more attributes. This cryptographic protection enables the recipient of a PA-TNC message to detect any changes to the content of the signed attributes that occurred after the data was signed. This protection includes both CMS signed attributes in the signedAttrs field or PA-TNC level attributes included in the content portion of CMS. Similarly the recipient can authenticate the identity of the sender of the attributes, and so is able to detect adversaries attempting to masquerade as a trustworthy origin of the attribute contents.

Section 5.2.2 through 5.2.5 of the PA-TNC specification discusses potential attacks against the integrity of the attributes exchange by creating falsified attributes, modifying legitimate attributes, inserting attributes within an exchange or replaying prior attributes.

The use of a digital signature covering the attributes' content allows each recipient to detect fabricated attributes that were claiming to come from a party other than the authenticated identity. The signer of a set of attributes must have the appropriate credentials in order to create a valid signature associated with a trusted sender. The digital signature includes a cryptographic digest of the contents of the attributes that enables the recipient to detect any alterations, additions or deletions to the signed content. Because the signature can cover multiple PA-TNC attributes, an attack can not remove one of the attributes without invalidating the hash value. The paTncNonce CMS attribute included in the signedAttrs field is also included in the CMS hash and signature. These CMS attribute also are protected from modification. Because the paTncNonce CMS attribute is mandated by this specification and includes freshness values from each party, attempts to replay previously valid attributes can be detected by the recipient using a replay cache. It is critical that Posture Collectors and Posture Validators check the nonce values prior to operating upon a received set of attributes to avoid replay attacks. This check includes validating that the nonce values are appropriate (incremented from prior values) and checking a cache of previously used initial nonce values. Finally, deployments could choose to also use enveloped-data encapsulation of the signed-data content. Enveloped-data provides encryption of the signed-data

using per-session encryption keys that would not be known (or replayable) by network based intermediaries.

4.1.2. Threats Addressed by Encrypted Attributes

The CMS enveloped-data content type used by PA-TNC security provides an encrypted envelope around the signed-data content to protect the signed data from disclosure while traveling between the Posture Collector and Posture Validator (even when traveling through the NEA posture brokers). The encryption of the signed set of attributes allows the attributes to pass through untrustworthy intermediary devices and components while maintaining the confidentiality and privacy of the information.

Section 5.2.1 of the PA-TNC specification discusses the threat of information theft by adversaries capable of intercepting the attributes while traversing the network and TNC architecture components. Deployers wishing to protect the exchanged attributes without trusting or using other countermeasures to protect the attributes can use enveloped-data to establish private attribute exchanges between Posture Collectors and Posture Validators. Malicious intermediaries would require knowledge of the encryption key (or indirectly via the key encrypting key) to obtain the attribute information.

4.2. Potential Threats Against PA-TNC use of CMS

The use of CMS with PA-TNC provides security protections for the exchanged PA-TNC attributes but CMS itself may be directly attacked by adversaries. This section discusses some potential threats to CMS.

4.2.1. Cryptography

CMS protections are based on the use of cryptographic digests, signatures, and encryption (both content and key). Signing, encryption and key management keys must be protected from a variety of potential threats that would result in their discovery by adversaries.

The encryption algorithms themselves become weaker over time and eventually may become vulnerable to various forms of attack including brute force. This risk is elevated as computing performance increases and new mathematical weaknesses are discovered allowing faster searching of the key space. Implementations should be agile enough to support protected dynamic negotiation and addition of new algorithms as

necessary. PA-TNC security offers dynamic discovery of supported cryptographic capabilities; this allows senders to use newer and stronger algorithms when recipients are also deployed with those algorithms. PA-TNC message senders should use care to not to send data using weak signature or encryption algorithms that are no longer appropriate for the sensitivity of the attributes being protected.

4.2.2. Threats to Keys

Signed-data content makes use of X.509 certificates for communicating the signer's public key and associated metadata, such as the holder's identity to recipients. These certificates are protected from alteration as long as recipients verify the content signature and properly inspect the signing certificate for validity, authenticity and trustworthiness prior to usage. Part of the validation process normally involves consulting one or more trust anchors typically manifested as a set of certificates associated with trusted certificate authorities. Implementations need to protect the trust anchor database from unauthorized modification, addition or deletion in order to ensure that only trusted certificate authorities are present. If an adversary is able to alter the trust anchor database then falsified certificates could pass validation and cause harm to the NEA deployment.

Enveloped-data content can make use of data encryption keys, initialization vectors and padding that are generated by the sender and that must be unpredictable by third parties using entropy that can not be influenced or predicted by untrusted software [11]. The generated keys must be resilient to passive eavesdropping and active attacks that attempt to steal them for future use. Therefore, CMS encrypts these keys when sent between the Posture Collector and Posture Validator using a variety of types of key management algorithms discussed in the CMS specification. Any non-public key used to encrypt the content encryption keys must also be protected from prediction or disclosure on the network, NEA Client or NEA Server system. Key management schemes that make use of previously distributed key encrypting key require those keys are protected from unauthorized access while on persistent storage and in memory. Failure to do so could lead to the exposure of the content encryption keys and thus the protected attributes.

4.2.3. Denial of Service

PA-TNC security provides a protective CMS wrapper around a set of one or more attributes allowing the recipient to detect attacks on the PA-TNC message attributes. However, while detection is possible, repair of the attribute is not, so recipients are forced to drop protected contents that have been altered. If an attacker can modify every protected attribute, this would result in the protected attributes being dropped and thus a denial of service (DoS) of the assessment.

Implementations should provide proper audit logging facilities and alerting capabilities to enable deployers to become aware of when such attacks are in progress. These facilities may also be used to cause other DoS attacks, so the amount of logging and alerting should be able to be throttled by deployer controls (e.g. notify the admin at most once per hour). A similar DoS attack can be achieved by a malicious intermediary that just drops all TNC messages.

Another form of DoS against the CMS protected content involves sending a high rate of PA-TNC messages containing large falsified or replayed enveloped-data protected attributes. This will cause the recipients to spend CPU cycles decrypting the messages before finding out the content is falsified or replayed when the attributes signatures is verified. This threat may not be feasible when an authenticated PT protocol is present.

Other forms of DoS attack target the CMS wrapper information for enveloped-data. This information is outside of the CMS signature so could be modified to cause problems for recipients processing the message after significant CPU time has occurred. For example an attacker might modify the recipientInfos structure to break the key management schemes used to exchange the content encryption keys. This could result in the encrypted content no longer be able to decrypt and the message would be discarded.

Finally, DoS attacks are possible by hostile intermediaries modifying the paTncErrorCode, paTncSecurityCapabilities or paTncNonce CMS attributes such that potential senders of protected information are unable to find common algorithms with their target recipients or pass the replay checks. Because the CMS signed attributes are contained in signedAttrs field, these modifications will be detected and thus the information discarded

5. IANA Considerations

One new IANA registry is defined by this specification: IETF Standard PA-TNC Error Code. This section explains how this registry will work.

First, it is important to note that the PA-TNC Error Code name space can support both IETF standard values listed in the IANA registry while allowing for vendor specific attributes to be used. The PA-TNC Error Codes are always accompanied by an SMI Private Enterprise Number (PEN) based OID, also known as the vendor ID. If this vendor ID is zero, the accompanying PA-TNC Error Code is an IETF standard value listed in the IANA registry and its meaning is defined in the RFC listed. If the vendorID OID is not zero, the meaning of the PA-TNC Error Code has a vendor-specific defined by the vendor identified by the vendorID OID (as listed in the IANA registry for SMI PENs).

The following subsections provide guidance to the IANA in creating and managing the new IANA registry defined by this specification.

5.1. Registry for IETF Standard PA-TNC Error Codes

The name for this registry is "IETF Standard PA-TNC Error Codes". Each entry in this registry should include a human-readable name, a decimal integer value between 0 and 2^16-1, and a reference to an RFC (long lived document) where this error code is defined. This RFC must define the meaning of this error code and a description of when it occurs. The RFC can be any form of RFC including experimental and be an individual submission.

Entries to this registry may only be added by IETF Consensus, as defined in RFC 2434 [2]. That is, they can only be added in an RFC approved by the IESG.

The following entries for this registry are defined in this document. Once this document becomes an RFC, they should become the initial entries in the registry for IETF Standard PB-TNC Error Codes.

Value	Name	Defining RFC
0	Reserved	This value MUST NOT be used
1	decodeFailure	RFC # Assigned to this I-D
2	badContentInfo	RFC # Assigned to this I-D
3	badSignedData	RFC # Assigned to this I-D
4	badEnvelopedData	RFC # Assigned to this I-D
5	badCertificate	RFC # Assigned to this I-D
6	badSignerInfo	RFC # Assigned to this I-D
7	badSignedAttrs	RFC # Assigned to this I-D
8	badUnsignedAttrs	RFC # Assigned to this I-D
9	missingContent	RFC # Assigned to this I-D
10	noTrustAnchor	RFC # Assigned to this I-D
11	notAuthorized	RFC # Assigned to this I-D
12	badDigestAlgorithm	RFC # Assigned to this I-D
13	badSignatureAlgorithm	RFC # Assigned to this I-D
14	unsupportedKeySize	RFC # Assigned to this I-D
15	unsupportedParameters	RFC # Assigned to this I-D
16	signatureFailure	RFC # Assigned to this I-D
17	decryptionFailure	RFC # Assigned to this I-D
18	keyManageFailure	RFC # Assigned to this I-D
19	badKeyManage	RFC # Assigned to this I-D
20	nonceMissing	RFC # Assigned to this I-D
21	invalidNonce	RFC # Assigned to this I-D
22	repeatedNonce	RFC # Assigned to this I-D
23	nonceOrdering	RFC # Assigned to this I-D
24	badContentType	RFC # Assigned to this I-D
25	digestAlgMismatch	RFC # Assigned to this I-D
29	missingSignature	RFC # Assigned to this I-D
30	resourcesBusy	RFC # Assigned to this I-D
31	versionNumberMismatch	RFC # Assigned to this I-D
33	revokedCertificate	RFC # Assigned to this I-D
65535	other	RFC # Assigned to this I-D

6. Acknowledgments

The authors of this draft would like to acknowledge the following people who have contributed to or provided substantial input on the preparation of this document or predecessors to it: Diana Arroyo, Stuart Bailey, Scott Cochrane, Sandilya Garimella, Lauren Giroux, Steve Hanna, Thomas Hardjono, Chris Hessing, Josh Howlett, John Jerrim, Meenakshi Kaushik, Greg Kazmierczak, Scott Kelly, PJ Kirner, Sung Lee, Lisa Lorenzin, Mahalingam Mani, Mauricio Sanchez, Ravi Sahita, Curtis Simonson, Brad Upson, Han Yin.

This document was prepared using 2-Word-v2.0.template.dot.

7. References

7.1. Normative References

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

[2] Alvestrand, H. and Narten T., "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

[3] Housley R., "Cryptographic Message Syntax (CMS) Algorithms", http://www.ietf.org/rfc/rfc3370.txt, IETF, August 2002.

[4] Turner. S., "Using SHA2 Algorithms with Cryptographic Message Syntax", IETF, Internet Draft, Work in Progress.

7.2. Informative References

[5] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and Tardo J., "Network Endpoint Assessment (NEA): Overview and Requirements", draft-ietf-nea-requirements-05.txt, Work In Progress, November 2007.

[6] Sangster, P., "PA-TNC: A Posture Attribute Protocol (PA) Compatible with TNC", draft-sangster-nea-pa-tnc-00.txt, February 2008.

[7] Sangster, P., "TNC IF-M Security: Bindings to CMS", Trusted Computing Group, February 2008.

[8] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", http://www.ietf.org/rfc/rfc3851.txt, IETF, July 2004.

[9] Housley R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", http://www.ietf.org/rfc/rfc4108.txt, IETF, August 2005.

[10] IANA, "Private Enterprise Numbers", http://www.iana.org/assignments/enterprise-numbers.

[11] Eastlake 3 , D., Crocker, S., and Schiller, J., "Randomness Recommendations for Security", http://www.ietf.org/rfc/rfc1740.txt, IETF, December 1994.

[12] Housley R., Wallace C., "Trust Anchor Management Protocol", draft-housley-tamp-00.txt, IETF, December 1994.

Author's Addresses

Paul Sangster
Symantec Corporation
6825 Citrine Dr
Carlsbad, CA 92009 USA
email: Paul_Sangster@symantec.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be

required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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