Internet Engineering Task Force (IETF)
Request for Comments: 5914
Category: Standards Track
ISSN: 2070-1721

R. Housley
Vigil Security, LLC
S. Ashmore
National Security Agency
C. Wallace
Cygnacom Solutions

June 2010

Trust Anchor Format

Abstract

This document describes a structure for representing trust anchor information. A trust anchor is an authoritative entity represented by a public key and associated data. The public key is used to verify digital signatures, and the associated data is used to constrain the types of information or actions for which the trust anchor is authoritative. The structures defined in this document are intended to satisfy the format-related requirements defined in Trust Anchor Management Requirements.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Trust anchors are widely used to verify digital signatures and validate certification paths [RFC5280][X.509]. They are required when validating certification paths. Though widely used, there is no standard format for representing trust anchor information. This document describes the TrustAnchorInfo structure. This structure is intended to satisfy the format-related requirements expressed in Trust Anchor Management Requirements [TA-MGMT-REQS] and is expressed using ASN.1 [X.680]. It can provide a more compact alternative to X.509 certificates for exchanging trust anchor information and provides a means of associating additional or alternative constraints with certificates without breaking the signature on the certificate.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Trust Anchor Information Syntax

This section describes the TrustAnchorInfo structure.

```
TrustAnchorInfo ::= SEQUENCE {
   version   TrustAnchorInfoVersion DEFAULT v1,
   pubKey    SubjectPublicKeyInfo,
   keyId     KeyIdentifier,
   taTitle   TrustAnchorTitle OPTIONAL,
   certPath  CertPathControls OPTIONAL,
   exts      [1] EXPLICIT Extensions   OPTIONAL,
   taTitleLangTag   [2] UTF8String OPTIONAL }

TrustAnchorInfoVersion ::= INTEGER { v1(1) }
```

2.1. Version

version identifies the version of TrustAnchorInfo. Future updates to this document may include changes to the TrustAnchorInfo structure, in which case the version number should be incremented. However, the default value, v1, cannot be changed.

2.2. Public Key

pubKey identifies the public key and algorithm associated with the trust anchor using the SubjectPublicKeyInfo structure [RFC5280]. The SubjectPublicKeyInfo structure contains the algorithm identifier followed by the public key itself. The algorithm field is an

AlgorithmIdentifier, which contains an object identifier and OPTIONAL parameters. The object identifier names the public key algorithm and indicates the syntax of the parameters, if present, as well as the format of the public key. The public key is encoded as a BIT STRING.

2.3. Key Identifier

keyId contains the public key identifier of the trust anchor public key. See Section 4.2.1.2 of [RFC5280] for a description of common key identifier calculation methods.

2.4. Trust Anchor Title

```
TrustAnchorTitle ::= UTF8String (SIZE (1..64))
```

taTitle is OPTIONAL. When it is present, it provides a human-readable name for the trust anchor. The text is encoded in UTF-8 [RFC3629], which accommodates most of the world's writing systems. The taTitleLangTag field identifies the language used to express the taTitle. When taTitleLangTag is absent, English ("en" language tag) is used. The value of the taTitleLangTag should be a language tag as described in [RFC5646].

2.5. Certification Path Controls

```
CertPathControls ::= SEQUENCE {
  taName           Name,
  certificate      [0] Certificate OPTIONAL,
  policySet        [1] CertificatePolicies OPTIONAL,
  policyFlags      [2] CertPolicyFlags OPTIONAL,
  nameConstr       [3] NameConstraints OPTIONAL,
  pathLenConstraint[4] INTEGER (0..MAX) OPTIONAL}
```

certPath is OPTIONAL. When it is present, it provides the controls needed to initialize an X.509 certification path validation algorithm implementation (see Section 6 of [RFC5280]). When absent, the trust anchor cannot be used to validate the signature on an X.509 certificate.

taName provides the X.500 distinguished name associated with the trust anchor, and this distinguished name is used to construct and validate an X.509 certification path. The name MUST NOT be an empty sequence.

certificate provides an OPTIONAL X.509 certificate, which can be used in some environments to represent the trust anchor in certification path development and validation. If the certificate is present, the subject name in the certificate MUST exactly match the X.500

distinguished name provided in the taName field, the public key MUST exactly match the public key in the pubKey field, and the subjectKeyIdentifier extension, if present, MUST exactly match the key identifier in the keyId field. The complete description of the syntax and semantics of the Certificate are provided in [RFC5280]. Constraints defined in the policySet, policyFlags, nameConstr, pathLenConstraint, and exts fields within TrustAnchorInfo replace values contained in a certificate or provide values for extensions not present in the certificate. Values defined in these TrustAnchorInfo fields are always enforced. Extensions included in a certificate are enforced only if there is no corresponding value in the TrustAnchorInfo. Correspondence between extensions within certificate and TrustAnchorInfo fields is defined as follows:

o an id-ce-certificatePolicies certificate extension corresponds to the CertPathControls.policySet field.

o an id-ce-policyConstraints certificate extension corresponds to the CertPolicyFlags.inhibitPolicyMapping and CertPolicyFlags.requireExplicitPolicy fields.

o an id-ce-inhibitAnyPolicy certificate extension corresponds to the CertPolicyFlags.inhibitAnyPolicy field.

o an id-ce-nameConstraints certificate extension corresponds to the CertPathControls.nameConstr field.

o the pathLenConstraint field of an id-ce-basicConstraints certificate extension corresponds to the CertPathControls.pathLenConstraint field (the presence of a CertPathControls structure corresponds to a TRUE value in the cA field of a BasicConstraints extension).

o any other certificate extension corresponds to the same type of extension in the TrustAnchorInfo.exts field.

```
 CertificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

 PolicyInformation ::= SEQUENCE {
   policyIdentifier  CertPolicyId,
   policyQualifiers  SEQUENCE SIZE (1..MAX) OF
                           PolicyQualifierInfo OPTIONAL }

 CertPolicyId ::= OBJECT IDENTIFIER
```

policySet is OPTIONAL. When present, it contains a sequence of certificate policy identifiers to be provided as inputs to the certification path validation algorithm. When absent, the special value any-policy is provided as the input to the certification path validation algorithm. The complete description of the syntax and semantics of the CertificatePolicies are provided in [RFC5280], including the syntax for PolicyInformation. In this context, the OPTIONAL policyQualifiers structure MUST NOT be included.

```
 CertPolicyFlags ::= BIT STRING {
   inhibitPolicyMapping   (0),
   requireExplicitPolicy  (1),
   inhibitAnyPolicy       (2) }
```

policyFlags is OPTIONAL. When present, three Boolean values for input to the certification path validation algorithm are provided in a BIT STRING. When absent, the input to the certification path validation algorithm is { FALSE, FALSE, FALSE }, which represents the most liberal setting for these flags. The three bits are used as follows:

inhibitPolicyMapping indicates if policy mapping is allowed in the certification path. When set to TRUE, policy mapping is not permitted. This value represents the initial-policy-mapping-inhibit input value to the certification path validation algorithm described in Section 6.1.1 of [RFC5280].

requireExplicitPolicy indicates if the certification path MUST be valid for at least one of the certificate policies in the policySet. When set to TRUE, all certificates in the certification path MUST contain an acceptable policy identifier in the certificate policies extension. This value represents the initial-explicit-policy input value to the certification path validation algorithm described in Section 6.1.1 of [RFC5280]. An acceptable policy identifier is a member of the policySet or the identifier of a policy that is declared to be equivalent through policy mapping. This bit MUST be set to FALSE if policySet is absent.

inhibitAnyPolicy indicates whether the special anyPolicy policy identifier, with the value { 2 5 29 32 0 }, is considered an explicit match for other certificate policies. This value represents the initial-any-policy-inhibit input value to the certification path validation algorithm described in Section 6.1.1 of [RFC5280].

```
 NameConstraints ::= SEQUENCE {
   permittedSubtrees  [0] GeneralSubtrees OPTIONAL,
   excludedSubtrees   [1] GeneralSubtrees OPTIONAL }

 GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree
```

```
     GeneralSubtree ::= SEQUENCE {
       base     GeneralName,
       minimum  [0] BaseDistance DEFAULT 0,
       maximum  [1] BaseDistance OPTIONAL }

     BaseDistance ::= INTEGER (0..MAX)
```

nameConstr is OPTIONAL. It has the same syntax and semantics as the Name Constraints certificate extension [RFC5280], which includes a list of permitted names and a list of excluded names. The definition of GeneralName can be found in [RFC5280]. When it is present, constraints are provided on names (including alternative names) that might appear in subsequent X.509 certificates in a certification path. This field is used to set the initial-permitted-subtrees and initial-excluded-subtrees input values to the certification path validation algorithm described in Section 6.1.1 of [RFC5280]. When this field is absent, the initial-permitted-subtrees variable is unbounded and the initial-excluded-subtrees variable is empty.

The pathLenConstraint field gives the maximum number of non-self-issued intermediate certificates that may follow this certificate in a valid certification path. (Note: The last certificate in the certification path is not an intermediate certificate and is not included in this limit. Usually, the last certificate is an end entity certificate, but it can be a CA certificate.) A pathLenConstraint of zero indicates that no non-self-issued intermediate certification authority (CA) certificates may follow in a valid certification path. Where it appears, the pathLenConstraint field MUST be greater than or equal to zero. Where pathLenConstraint does not appear, no limit is imposed.

When the trust anchor is used to validate a certification path, CertPathControls provides limitations on certification paths that will successfully validate. An application that is validating a certification path SHOULD NOT ignore these limitations, but the application can impose additional limitations to ensure that the validated certification path is appropriate for the intended application context. As input to the certification path validation algorithm, an application MAY:

- o Provide a subset of the certification policies provided in the policySet;
- o Provide a TRUE value, if appropriate, for any of the flags in the policyFlags;
- o Provide a subset of the permitted names provided in the nameConstr;

o Provide additional excluded names to the ones that are provided in the nameConstr;

o Provide a smaller value for pathLenConstraint.

2.6. Extensions

exts is OPTIONAL. When it is present, it can be used to associate additional information with the trust anchor using the standard Extensions structure. Extensions that are anticipated to be widely used have been included in the CertPathControls structure to avoid overhead associated with use of the Extensions structure. To avoid duplication with the CertPathControls field, the following types of extensions MUST NOT appear in the exts field and are ignored if they do appear: id-ce-certificatePolicies, id-ce-policyConstraints, id-ce-inhibitAnyPolicy, or id-ce-nameConstraints.

3. Trust Anchor List

TrustAnchorInfo allows for the representation of a single trust anchor. In many cases, it is convenient to represent a collection of trust anchors. The TrustAnchorList structure is defined for this purpose. TrustAnchorList is defined as a sequence of one or more TrustAnchorChoice objects. TrustAnchorChoice provides three options for representing a trust anchor. The certificate option allows for the use of a certificate with no additional associated constraints. The tbsCert option allows for associating constraints by removing a signature on a certificate and changing the extensions field. The taInfo option allows for use of the TrustAnchorInfo structure defined in this document.

```
   TrustAnchorList ::= SEQUENCE SIZE (1..MAX) OF TrustAnchorChoice

   TrustAnchorChoice ::= CHOICE {
      certificate  Certificate,
      tbsCert      [1] EXPLICIT TBSCertificate,
      taInfo       [2] EXPLICIT TrustAnchorInfo }

    trust-anchor-list PKCS7-CONTENT-TYPE ::=
       { TrustAnchorList IDENTIFIED BY id-ct-trustAnchorList }
```

The TrustAnchorList structure can be protected using the SignedData structure defined in the Cryptographic Message Syntax (CMS) [RFC5652]. The id-ct-trustAnchorList object identifier has been defined to represent TrustAnchorList payloads with CMS structures.

4. Security Considerations

Compromise of a trust anchor private key permits unauthorized parties to masquerade as the trust anchor, with potentially severe consequences. Where TA-based constraints are enforced, the unauthorized holder of the trust anchor private key will be limited by the certification path controls associated with the trust anchor, as expressed in the certPath and exts fields. For example, name constraints in the trust anchor will determine the name space that will be accepted in certificates that are validated using the compromised trust anchor. Reliance on an inappropriate or incorrect trust anchor public key has similar potentially severe consequences.

The compromise of a CA's private key leads to the same type of problems as the compromise of a trust anchor private key. The unauthorized holder of the CA private key will be limited by the certification path controls associated with the trust anchor, as expressed in the certPath field or as an extension.

Usage of a certificate independent of the TrustAnchorInfo structure that envelopes it must be carefully managed to avoid violating constraints expressed in the TrustAnchorInfo. When enveloping a certificate in a TrustAnchorInfo structure, values included in the certificate should be evaluated to ensure there is no confusion or conflict with values in the TrustAnchorInfo structure.

5. References

5.1. Normative References

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, June 2010.

[X.680] "ITU-T Recommendation X.680: Information Technology - Abstract Syntax Notation One", 2002.

5.2. Informative References

[TA-MGMT-REQS] Reddy, R. and C. Wallace, "Trust Anchor Management Requirements", Work in Progress, March 2010.

[X.509] "ITU-T Recommendation X.509 - The Directory - Authentication Framework", 2000.

Appendix A. ASN.1 Modules

A.1. ASN.1 Module Using 2002 Syntax

Appendix A.1 provides the normative ASN.1 definitions for the structures described in this specification using ASN.1 as defined in [X.680]. It includes definitions imported from [RFC5280] and [RFC5912].

```
   TrustAnchorInfoModule
   { joint-iso-ccitt(2) country(16) us(840) organization(1)
      gov(101) dod(2) infosec(1) modules(0) 33 }

   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   IMPORTS
   Certificate, Name, SubjectPublicKeyInfo, TBSCertificate
   FROM PKIX1Explicit-2009 -- from [RFC5912]
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51)}
   CertificatePolicies, KeyIdentifier, NameConstraints
   FROM PKIX1Implicit-2009 -- from [RFC5912]
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}
   Extensions{}
   FROM PKIX-CommonTypes-2009 -- from [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-pkixCommon-02(57) }         ;

   TrustAnchorInfo ::= SEQUENCE {
      version   TrustAnchorInfoVersion DEFAULT v1,
      pubKey    SubjectPublicKeyInfo,
      keyId     KeyIdentifier,
      taTitle   TrustAnchorTitle OPTIONAL,
      certPath  CertPathControls OPTIONAL,
      exts      [1] EXPLICIT Extensions {{...}}   OPTIONAL,
      taTitleLangTag   [2] UTF8String OPTIONAL }

   TrustAnchorInfoVersion ::= INTEGER { v1(1) }

   TrustAnchorTitle ::= UTF8String (SIZE (1..64))

   CertPathControls ::= SEQUENCE {
     taName           Name,
     certificate      [0] Certificate OPTIONAL,
     policySet        [1] CertificatePolicies OPTIONAL,
     policyFlags      [2] CertPolicyFlags OPTIONAL,
     nameConstr       [3] NameConstraints OPTIONAL,
     pathLenConstraint[4] INTEGER (0..MAX) OPTIONAL}

   CertPolicyFlags ::= BIT STRING {
     inhibitPolicyMapping    (0),
     requireExplicitPolicy   (1),
     inhibitAnyPolicy        (2) }

   TrustAnchorList ::= SEQUENCE SIZE (1..MAX) OF TrustAnchorChoice

   TrustAnchorChoice ::= CHOICE {
     certificate  Certificate,
     tbsCert      [1] EXPLICIT TBSCertificate,
     taInfo       [2] EXPLICIT TrustAnchorInfo }

   id-ct-trustAnchorList      OBJECT IDENTIFIER ::= { iso(1)
       member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       id-smime(16) id-ct(1) 34 }

   PKCS7-CONTENT-TYPE ::= TYPE-IDENTIFIER

   trust-anchor-list PKCS7-CONTENT-TYPE ::=
       { TrustAnchorList IDENTIFIED BY id-ct-trustAnchorList }

   END
```

A.2. ASN.1 Module Using 1988 Syntax

Appendix A.2 provides the normative ASN.1 definitions for the structures described in this specification using ASN.1 as defined in [X.680].

A.2.1. ASN.1 Module

```
   TrustAnchorInfoModule-88
       { joint-iso-ccitt(2) country(16) us(840) organization(1)
         gov(101) dod(2) infosec(1) modules(0) 37 }

   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   IMPORTS
   Certificate, Name, Extensions,
   SubjectPublicKeyInfo, TBSCertificate
      FROM PKIX1Explicit88 -- from [RFC5280]
           { iso(1) identified-organization(3) dod(6) internet(1)
             security(5) mechanisms(5) pkix(7) id-mod(0)
             id-pkix1-explicit(18) }
   CertificatePolicies, KeyIdentifier, NameConstraints
      FROM PKIX1Implicit88 -- [RFC5280]
           { iso(1) identified-organization(3) dod(6) internet(1)
             security(5) mechanisms(5) pkix(7) id-mod(0)
             id-pkix1-implicit(19) }
    ;

   TrustAnchorInfo ::= SEQUENCE {
      version   TrustAnchorInfoVersion DEFAULT v1,
      pubKey    SubjectPublicKeyInfo,
      keyId     KeyIdentifier,
      taTitle   TrustAnchorTitle OPTIONAL,
      certPath  CertPathControls OPTIONAL,
      exts      [1] EXPLICIT Extensions   OPTIONAL,
      taTitleLangTag   [2] UTF8String OPTIONAL }

   TrustAnchorInfoVersion ::= INTEGER { v1(1) }

   TrustAnchorTitle ::= UTF8String (SIZE (1..64))

   CertPathControls ::= SEQUENCE {
     taName           Name,
     certificate      [0] Certificate OPTIONAL,
     policySet        [1] CertificatePolicies OPTIONAL,
     policyFlags      [2] CertPolicyFlags OPTIONAL,
     nameConstr       [3] NameConstraints OPTIONAL,
     pathLenConstraint[4] INTEGER (0..MAX) OPTIONAL}

   CertPolicyFlags ::= BIT STRING {
     inhibitPolicyMapping    (0),
     requireExplicitPolicy   (1),
     inhibitAnyPolicy        (2) }

   TrustAnchorList ::= SEQUENCE SIZE (1..MAX) OF TrustAnchorChoice

   TrustAnchorChoice ::= CHOICE {
     certificate  Certificate,
     tbsCert      [1] EXPLICIT TBSCertificate,
     taInfo       [2] EXPLICIT TrustAnchorInfo }

   id-ct-trustAnchorList      OBJECT IDENTIFIER ::= { iso(1)
       member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       id-smime(16) id-ct(1) 34 }

   END
```

Authors' Addresses

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170

EMail: housley@vigilsec.com

Sam Ashmore
National Security Agency
Suite 6751
9800 Savage Road
Fort Meade, MD 20755

EMail: srashmo@radium.ncsc.mil

Carl Wallace
Cygnacom Solutions
Suite 5400
7925 Jones Branch Drive
McLean, VA 22102

EMail: cwallace@cygnacom.com