   Using Trust Anchor Constraints during Certification Path Processing

Abstract

   This document describes how to use information associated with a
   trust anchor public key when validating certification paths. This
   information can be used to constrain the usage of a trust anchor.
   Typically, constraints are used to limit the certificate policies and
   names that can appear in certification paths validated using a trust
   anchor.

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. The Trust Anchor Format (TAF)
   specification [RFC5914] defines a means for limiting the scope in
   which a trust anchor may be used. [RFC5280] describes how to
   validate a certification path. The algorithm requires processing the
   name and key from a trust anchor. Usage of other information,
   including enforcement of constraints, is permitted but not required,
   and the processing rules are not specified (see Section 6.2 of
   RFC 5280).

   This document defines a mechanism to identify constraints that should
   be enforced and the supplementary processing rules. The
   supplementary rules specify an additional input and extend the
   initialization procedures in the [RFC5280] path validation algorithm.
   Post-initialization processing steps are not affected.

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. Identifying Trust Anchor Constraints

   TAF supports three formats for representing trust anchor information:
   TrustAnchorInfo, Certificate, and TBSCertificate. In all three
   cases, trust anchor constraints may be represented as extensions. In
   the TrustAnchorInfo structure, certificate policies, policy
   constraints, name constraints, inhibit any policy, and basic
   constraints do not appear as extensions and instead appear as part of
   the CertPathControls field.

   Extensions may be marked critical or not critical. When trust anchor
   constraints are enforced, clients MUST reject certification paths
   containing a trust anchor with unrecognized critical extensions.
   When trust anchor constraints are not enforced, clients MAY accept
   certification paths containing a trust anchor with unrecognized
   critical extensions. Information appearing in the CertPathControls
   field of a TrustAnchorInfo object MUST be processed, ensuring
   enforcement of the constraints indicated by this field in all cases.

   For some types of trust anchor constraints, there is a type mismatch
   between the input parameters for the certification path validation
   algorithm and the extension that contains the constraint. The
   certification path validation algorithm essentially defines the

   initial-any-policy-inhibit, initial-policy-mapping-inhibit, and
   initial-explicit-policy as Boolean values. The inhibitAnyPolicy and
   policyConstraints extensions that correspond to these inputs are
   expressed as integer values. In the steps described below, presence
   of an inhibitAnyPolicy extension results in the initial-any-policy-
   inhibit value being set to true. If a policyConstraints extension is
   present and contains a requireExplicitPolicy field, the initial-
   explicit-policy value is set to true. If a policyConstraints
   extension is present and contains an inhibitPolicyMapping field, the
   initial-policy-mapping-inhibit value is set to true.

3. Using Trust Anchor Constraints during Certification Path Processing

3.1. Inputs

   This algorithm assumes that the nine inputs defined in Section 6.1.1
   of RFC 5280 are provided to the path processing logic, plus one
   additional variable:

   o enforceTrustAnchorConstraints: indicates if trust anchor
      constraints should be enforced

   Conforming implementations are not required to support the setting of
   the enforceTrustAnchorConstraints input. If a conforming
   implementation does not support the setting of this flag, it MUST
   validate all certification paths using a value of TRUE for
   enforceTrustAnchorConstraints.

3.2. Initialization

   If enforceTrustAnchorConstraints is false, no additional
   initialization steps are required.

   If enforceTrustAnchorConstraints is true, perform the following
   initialization steps described below. These steps (or equivalent)
   MUST be performed prior to initialization steps described in
   RFC 5280.

   o If no subject distinguished name is associated with the trust
      anchor, path validation fails. The name may appear in the subject
      field of a Certificate or TBSCertificate structure or in the
      taName field of CertPathControls in a TrustAnchorInfo structure.

   o If name constraints are associated with the trust anchor, set the
      initial-permitted-subtrees variable equal to the intersection of
      the permitted subtrees from the trust anchor and the user-provided

   initial-permitted-subtrees. If one of these two inputs is not
   provided, the initial-permitted-subtrees variable is set to the
   value that is available. If neither is provided, the initial-
   permitted-subtrees variable is set to an infinite set.

   o If name constraints are associated with the trust anchor, set the
      initial-excluded-subtrees variable equal to the union of the
      excluded subtrees from the trust anchor and the user-provided
      initial-excluded-subtrees. If one of these two inputs is not
      provided, the initial-excluded-subtrees variable is set to the
      value that is available. If neither is provided, the initial-
      excluded-subtrees variable is set to an empty set.

   o If certificate policies are associated with the trust anchor, set
      the user-initial-policy-set variable equal to the intersection of
      the certificate policies associated with the trust anchor and the
      user-provided user-initial-policy-set. If one of these two inputs
      is not provided, the user-initial-policy-set variable is set to
      the value that is available. If neither is provided, the
      user-initial-policy-set variable is set to any-policy.

   o If an inhibit any policy value of true is associated with the
      trust anchor (either in a CertPathControls or in an
      inhibitAnyPolicy extension) and the initial-any-policy-inhibit
      value is false, set the initial-any-policy-inhibit value to true.

   o If a require explicit policy value of true is associated with the
      trust anchor (either in a CertPathControls or in a
      PolicyConstraints extension) and the initial-explicit-policy value
      is false, set the initial-explicit-policy value to true.

   o If an inhibit policy mapping value of true is associated with the
      trust anchor (either in a CertPathControls or in a
      PolicyConstraints extension) and the initial-policy-mapping-
      inhibit value is false, set the initial-policy-mapping-inhibit
      value to true.

   o If a basic constraints extension is associated with the trust
      anchor and contains a pathLenConstraint value, set the
      max_path_length state variable equal to the pathLenConstraint
      value from the basic constraints extension.

3.3. Basic Certificate Processing

   This document does not require any augmentation of the basic
   certificate processing steps described in Section 6.1.3 of RFC 5280.
   However, some types of trust anchor constraints may have defined
   additional steps, for example, CMS Content Constraints or Authority
   Clearance Constraints.

3.4. Preparation for Certificate i+1

   This document does not require any augmentation of the steps to
   prepare for processing of certificate i+1 described in Section 6.1.4
   of RFC 5280. However, some types of trust anchor constraints may
   have defined additional steps, for example, CMS Content Constraints
   or Authority Clearance Constraints.

3.5. Wrap-Up Procedure

   This document does not require any augmentation of the wrap-up
   procedure steps described in Section 6.1.5 of RFC 5280. However,
   some types of trust anchor constraints may have defined additional
   steps, for example, CMS Content Constraints or Authority Clearance
   Constraints.

4. Relationship to RFC 5280

   The processing described above can be incorporated into an RFC 5280
   implementation or be implemented as pre-processing of RFC 5280 inputs
   and post-processing of RFC 5280 outputs, i.e., as a wrapper around an
   RFC 5280 compliant implementation.

   For name constraints and policy-related constraints, pre-processing
   can be used, provided the RFC 5280 implementation allows
   configuration of the user-initial-policy-set, initial-policy-mapping-
   inhibit, initial-explicit-policy, initial-any-policy-inhibit,
   initial-permitted-subtrees, and initial-excluded-subtrees input
   values. RFC 5280 does not define an input for path length
   constraints, so basic constraints cannot be implemented using
   pre-processing. It can be implemented as post-processing, provided
   the RFC 5280 implementation returns the certification path to enable
   the post-processor to perform the length check.

   Some types of trust anchor constraints may impose additional
   requirements on an RFC 5280 implementation to support pre-processing
   or post-processing to enforce trust anchor constraints.

5. Security Considerations

   Implementations that do not enforce trust anchor constraints may
   accept some certification paths rejected by implementations that do
   enforce trust anchor constraints. For example, an application that
   does not enforce a certificate policy constraint included in a trust
   anchor may accept certificates issued under a certificate policy that
   provides a lower-than-required-level of assurance.

   Trust anchor information must be securely stored. Changes to trust
   anchor information can cause acceptance of certificates that should
   be rejected. For example, if a trust anchor definition is altered to
   remove a name constraint, applications may accept certificates
   containing names that should only be trusted in certificates that
   validate to a different trust anchor. Similarly, addition of
   inappropriate trust anchors to a trust anchor store can result in
   validation of certificates to a different trust anchor and with
   different constraints than intended.

   [RFC5914] and [RFC5934] provide additional security considerations
   regarding the preparation, storage, and usage of trust anchors.
   [RFC5280] provides additional security considerations regarding the
   usage of name constraints.

6. References

6.1. Normative References

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

   [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.

   [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor
              Format", RFC 5914, June 2010.

6.2. Informative References

   [RFC5934] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor
              Management Protocol (TAMP)", RFC 5934, August 2010.

   [X.509]   ITU-T Recommendation X.509 (2005) | ISO/IEC 9594-8:2005,
              Information technology - Open Systems Interconnection -
              The Directory: Public-key and attribute certificate
              frameworks.

Authors' Addresses

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

    EMail: srashmo@radium.ncsc.mil

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

    EMail: cwallace@cygnacom.com