        Securing Routing Policy Specification Language (RPSL) Objects
          with Resource Public Key Infrastructure (RPKI) Signatures

Abstract

   This document describes a method that allows parties to
   electronically sign Routing Policy Specification Language objects and
   validate such electronic signatures. This allows relying parties to
   detect accidental or malicious modifications of such objects. It
   also allows parties who run Internet Routing Registries or similar
   databases, but do not yet have authentication (based on Routing
   Policy System Security) of the maintainers of certain objects, to
   verify that the additions or modifications of such database objects
   are done by the legitimate holder(s) of the Internet resources
   mentioned in those objects. This document updates RFCs 2622 and 4012
   to add the signature attribute to supported RPSL objects.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7909.

Copyright Notice

Table of Contents

1. Introduction

   Objects stored in resource databases, like the RIPE DB, are generally
   protected by an authentication mechanism: anyone creating or
   modifying an object in the database has to have proper authorization
   to do so, and therefore has to go through an authentication procedure
   (provide a password, certificate, email signature, etc.). However,
   for objects transferred between resource databases, the
   authentication is not guaranteed. This means that when a Routing
   Policy Specification Language (RPSL) object is downloaded from a
   database, the consumer can reasonably claim that the object is
   authentic if it was locally created, but cannot make the same claim
   for an object imported from a different database. Also, once such an
   object is downloaded from the database, it becomes a simple (but
   still structured) text file with no integrity protection. More
   importantly, the authentication and integrity guarantees associated
   with these objects do not always ensure that the entity that
   generated them is authorized to make the assertions implied by the
   data contained in the objects.

   A potential use for resource certificates [RFC6487] is to use them to
   secure such (both imported and downloaded) database objects, by
   applying a digital signature over the object contents in lieu of
   methods such as Routing Policy System Security [RFC2725]. The signer
   of such signed database objects MUST possess a relevant resource
   certificate, which shows him/her as the legitimate holder of an
   Internet number resource. This mechanism allows the users of such
   database objects to verify that the contents are in fact produced by
   the legitimate holder(s) of the Internet resources mentioned in those
   objects. It also allows the signatures to cover whole RPSL objects,
   or just selected attributes of them. In other words, a digital
   signature created using the private key associated with a resource
   certificate can offer object security in addition to the channel
   security already present in most resource databases. Object security
   in turn allows such objects to be hosted in different databases and
   still be independently verifiable.

   While the approach outlined in this document mandates the use of the
   Resource Public Key Infrastructure (RPKI) for certificate
   distribution, it is not dependent upon the RPKI for correct
   functionality. Equivalent functionality can be achieved with a more
   traditional Certification Authority (CA), using the extensions
   described in [RFC3779] within the certificates, and the appropriate
   trust anchor material to verify the digital signature.

The capitalized 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
[RFC2119].

2. Signature Syntax and Semantics

When signing an RPSL object [RFC2622] [RFC4012], the input for the
signature process is transformed into a sequence of strings of ASCII
data. The approach is similar to the one used in Domain Key
Identified Mail (DKIM) [RFC6376]. In the case of RPSL, the object to
be signed closely resembles an SMTP header, so it seems reasonable to
adapt DKIM's relevant features.

2.1. General Attributes and Meta Information

The digital signature associated with an RPSL object is itself a new
attribute named "signature". It consists of mandatory and optional
fields. These fields are structured in a sequence of name and value
pairs, separated by a semicolon ";" and a whitespace. Collectively,
these fields make up the value for the new "signature" attribute.
The "name" part of such a component is always a single ASCII
character that serves as an identifier; the value is an ASCII string
the contents of which depend on the field type. Mandatory fields
MUST appear exactly once, whereas optional fields MUST appear at most
once.

Mandatory fields of the "signature" attribute:

o   Version of the signature (field "v"): This field MUST be set to
    "rpkiv1" and MAY be the first field of the signature attribute to
    simplify the parsing of the attributes' fields. The signature
    format described in this document applies when the version field
    is set to "rpkiv1". All the rest of the signature attributes are
    defined by the value of the version field.

o   Reference to the certificate corresponding to the private key used
    to sign this object (field "c"): The value of this field MUST be a
    URL of type "rsync" [RFC5781] or "http(s)" [RFC7230] that points
    to a specific resource certificate in an RPKI repository
    [RFC6481]. Any non URL-safe characters (including semicolon ";"
    and plus "+") must be URL encoded [RFC3986].

o   Signature method (field "m"): What hash and signature algorithms
    were used to create the signature. This specification follows the
    algorithms defined in RFC 6485 [RFC6485]. The algorithms are
    referenced within the signature attribute by the ASCII names of
    the algorithms.

o Time of signing (field "t"): The format of the value of this field
   MUST be in the Internet Date/Time ABNF format [RFC3339]. All
   times MUST be converted to Universal Coordinated Time (UTC), i.e.,
   the ABNF time-offset is always "Z".

o The signed attributes (field "a"): This is a list of attribute
   names, separated by an ASCII "+" character (if more than one
   attribute is enumerated). The list must include any attribute at
   most once.

o The signature itself (field "b"): This MUST be the last field in
   the list. The signature is the output of the signature algorithm
   using the appropriate private key and the calculated hash value of
   the object as inputs. The value of this field is the digital
   signature in base64 encoding (Section 4 of [RFC4648]).

Optional fields of the "signature" attribute:

o Signature expiration time (field "x"): The format of the value of
   this field MUST be in the Internet Date/Time format [RFC3339].
   All times MUST be represented in UTC.

2.2. Signed Attributes

   One can look at an RPSL object as an (ordered) set of attributes,
   each having a "key: value" syntax. Understanding this structure can
   help in developing more flexible methods for applying digital
   signatures.

   Some of these attributes are automatically added by the database,
   some are database-dependent, yet others do not carry operationally
   important information. This specification allows the maintainer of
   such an object to decide which attributes are important (signed) and
   which are not (not signed), from among all the attributes of the
   object; in other words, we define a way of including important
   attributes while excluding irrelevant ones. Allowing the maintainer
   of an object to select the attributes that are covered by the digital
   signature achieves the goals established in Section 1.

   The type of the object determines the minimum set of attributes that
   MUST be signed. The signer MAY choose to sign additional attributes,
   in order to provide integrity protection for those attributes too.

   When verifying the signature of an object, the verifier has to check
   whether the signature itself is valid, and whether all the specified
   attributes are referenced in the signature. If not, the verifier
   MUST reject the signature and treat the object as a regular, unsigned
   RPSL object.

2.3. Storage of the Signature Data

   The result of applying the signature mechanism once is exactly one
   new attribute for the object. As an illustration, the structure of a
   signed RPSL object is as follows:

      attribute1: value1
      attribute2: value2
      attribute3: value3
      ...
      signature: v=rpkiv1; c=rsync://.....; m=sha256WithRSAEncryption;
                   t=2014-12-31T23:59:60Z;
                   a=attribute1+attribute2+attribute3+...;
                   b=<base64 data>

2.4. Number Resource Coverage

   Even if the signature over the object is valid according to the
   signature validation rules, it may not be relevant to the object; it
   also needs to cover the relevant Internet number resources mentioned
   in the object.

   Therefore, the Internet number resources present in [RFC3779]
   extensions of the certificate referred to in the "c" field of the
   signature MUST cover the resources in the primary key of the object
   (e.g., value of the "aut-num:" attribute of an aut-num object, value
   of the "inetnum:" attribute of an inetnum object, values of "route:",
   and "origin:" attributes of a route object, etc.).

2.5. Validity Time of the Signature

   The validity time interval of a signature is the intersection of the
   validity time of the certificate used to verify the signature, the
   "not before" time specified by the "t" field of the signature, and
   the optional "not after" time specified by the "x" field of the
   signature.

   When checking multiple signatures, these checks are individually
   applied to each signature.

3. Signature Creation and Validation Steps

3.1. Canonicalization

   The notion of canonicalization is essential to digital signature
   generation and validation whenever data representations may change
   between a signer and one or more signature verifiers.
   Canonicalization defines how one transforms a representation of data

into a series of bits for signature generation and verification. The
task of canonicalization is to make irrelevant differences in
representations of the same object, which would otherwise cause
signature verification to fail. Examples of this could be:

o data transformations applied by the databases that host these
   objects (such as notational changes for IPv4/IPv6 prefixes,
   automatic addition/modification of "changed" attributes, etc.)

o the difference of line terminators across different systems

This means that the destination database might change parts of the
submitted data after it was signed, which would cause signature
verification to fail. This document specifies strict
canonicalization rules to overcome this problem.

The following steps MUST be applied in order to achieve canonicalized
representation of an object, before the actual signature
(verification) process can begin:

1. Comments (anything beginning with a "#") MUST be omitted.

2. Any trailing whitespace MUST be omitted.

3. A multi-line attribute MUST be converted into its single-line
    equivalent. This is accomplished by:

    * Converting all line endings to a single blank space (ASCII
       code 32).

    * Concatenating all lines into a single line.

    * Replacing the trailing blank space with a single new line
       ("\n", ASCII code 10).

4. Numerical fields MUST be converted to canonical representations.
    These include:

    * Date and time fields MUST be converted to UTC and MUST be
       represented in the Internet Date/Time format [RFC3339].

    * AS numbers MUST be converted to ASPLAIN syntax [RFC5396].

    * IPv6 addresses MUST be canonicalized as defined in [RFC5952].

    * IPv4 addresses MUST be represented as the ipv4-address type
       defined by RPSL [RFC2622].

        * All IP prefixes (IPv4 and IPv6) MUST be represented in
           Classless Inter-Domain Routing (CIDR) notation [RFC4632].

   5. All ranges, lists, or sets of numerical fields are represented
       using the appropriate RPSL attribute and each numerical element
       contained within those attributes MUST conform to the
       canonicalization rules in this document. The ordering of values
       within such fields MUST be maintained during database transfers.

   6. The name of each attribute MUST be converted into lower case, and
       MUST be kept as part of the attribute line.

   7. Tab characters ("\t", ASCII code 09) MUST be converted into
       spaces.

   8. Multiple whitespaces MUST be collapsed into a single space (" ",
       ASCII code 32) character.

   9. All line endings MUST be converted into a single new line ("\n",
       ASCII code 10) character, (thus avoiding CR vs. CRLF
       differences).

3.2. Signature Creation

   Given an RPSL object and corresponding certificate, in order to
   create the digital signature, the following steps MUST be performed:

   1. Create a list of attribute names referring to the attributes that
       will be signed (contents of the "a" field). The minimum set of
       these attributes is determined by the object type; the signer MAY
       select additional attributes.

   2. Arrange the selected attributes according to the selection
       sequence specified in the "a" field as above, omitting all
       attributes that will not be signed.

   3. Construct the new "signature" attribute, with all its fields,
       leaving the value of the "b" field empty.

   4. Apply canonicalization rules to the result (including the
       "signature" attribute).

   5. Create the signature over the results of the canonicalization
       process (according to the signature and hash algorithms specified
       in the "m" field of the signature attribute).

   6. Insert the base64-encoded value of the signature as the value of
       the "b" field.

   7. Append the resulting "signature" attribute to the original
       object.

3.3. Signature Validation

   In order to validate a signature over such an object, the following
   steps MUST be performed:

   1. Verify the syntax of the "signature" attribute (i.e., whether it
       contains the mandatory and optional components and the syntax of
       these fields matches the specification as described in
       Section 2.1).

   2. Fetch the certificate referred to in the "c" field of the
       "signature" attribute, and check its validity using the steps
       described in [RFC6487].

   3. Extract the list of attributes that were signed using the signer
       from the "a" field of the "signature" attribute.

   4. Verify that the list of signed attributes includes the minimum
       set of attributes for that object type.

   5. Arrange the selected attributes according to the selection
       sequence provided in the value of the "a" field, omitting all
       unsigned attributes.

   6. Replace the value of the signature field "b" of the "signature"
       attribute with an empty string.

   7. Apply the canonicalization procedure to the selected attributes
       (including the "signature" attribute).

   8. Check the validity of the signature using the signature algorithm
       specified in the "m" field of the signature attribute, the public
       key contained in the certificate mentioned in the "c" field of
       the signature, the signature value specified in the "b" field of
       the signature attribute, and the output of the canonicalization
       process.

4. Signed Object Types and Set of Signed Attributes

   This section describes a list of object types that MAY be signed
   using this approach. For each object type, the set of attributes
   that MUST be signed for these object types (the minimum set noted in
   Section 3.3 is enumerated.

This list generally excludes attributes that are used to maintain
referential integrity in the databases that carry these objects,
since these usually make sense only within the context of such a
database, whereas the scope of the signatures is only one specific
object. Since the attributes in the referred object (such as mnt-by,
admin-c, tech-c, etc.) can change without any modifications to the
signed object, signing such attributes could lead to a false sense of
security in terms of the contents of the signed data; therefore,
including such attributes should only be done in order to provide
full integrity protection of the object itself.

The newly constructed "signature" attribute is always included in the
list. The signature under construction MUST NOT include signature
attributes that are already present in the object.

    as-block:

    * as-block

    * signature

    aut-num:

    * aut-num
    * as-name
    * member-of
    * import
    * mp-import
    * export
    * mp-export
    * default
    * mp-default
    * signature

    inet[6]num:

    * inet[6]num
    * netname
    * country
    * status
    * signature

```
   route[6]:

   *  route[6]
   *  origin
   *  holes
   *  member-of
   *  signature
```

It should be noted that the approach defined in this document has a
limitation in signing route[6] objects. This document only supports
a single signature per object. This means that it is not possible to
properly sign route[6] objects where one resource holder possesses
the Autonomous System Number (ASN) and another resource holder
possesses the referenced prefix. A future version of this
specification may resolve this limitation.

For each signature, the extension described in RFC 3779 that appears
in the certificate used to verify the signature MUST include a
resource entry that is equivalent to, or covers (i.e., is "less
specific" than) the following resources mentioned in the object the
signature is attached to:

o For the as-block object type: the resource in the "as-block"
   attribute.

o For the aut-num object type: the resource in the "aut-num"
   attribute.

o For the inet[6]num object type: the resource in the "inet[6]num"
   attribute.

o For the route[6] object type: the resource in the "route[6]" or
   "origin" (or both) attributes.

5. Keys and Certificates Used for Signature and Verification

The certificate that is referred to in the signature (in the "c"
field):

o MUST be an end-entity (i.e., non-CA) certificate

o MUST conform to the X.509 PKIX Resource Certificate profile
   [RFC6487]

o MUST have the extension described in RFC 3779 that covers the
   Internet number resource included in a signed attribute [RFC3779]

The certificate generated will omit the Subject Information Access
(SIA) extension mandated by RFC 6487 as that extension requires an
rsync URI for the accessLocation form and RPSL currently does not
support database access via rsync.

6. Security Considerations

RPSL objects stored in the Internet Routing Registry (IRR) databases
are public, and as such there is no need for confidentiality. Each
signed RPSL object can have its integrity and authenticity verified
using the supplied digital signature and the referenced certificate.

Since the RPSL signature approach leverages X.509 extensions, the
security considerations in [RFC3779] apply here as well.
Additionally, implementers MUST follow the certificate validation
steps described in RFC 6487.

The maintainer of an object has the ability to include attributes in
the signature that are not included in the resource certificate used
to create the signature. Potentially, a maintainer may include
attributes that reference resources the maintainer is not authorized
to use.

It should be noted that this digital signature does not preclude
monkey-in-the-middle attacks where the adversary either intercepts
RPSL object transfers, deletes the signature attribute, modifies the
contents, or intercepts the transfer and drops the objects destined
for the requester.

7. References

7.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D.,
              Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra,
              "Routing Policy Specification Language (RPSL)", RFC 2622,
              DOI 10.17487/RFC2622, June 1999,
              <http://www.rfc-editor.org/info/rfc2622>.

   [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet:
              Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002,
              <http://www.rfc-editor.org/info/rfc3339>.

   [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP
              Addresses and AS Identifiers", RFC 3779,
              DOI 10.17487/RFC3779, June 2004,
              <http://www.rfc-editor.org/info/rfc3779>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <http://www.rfc-editor.org/info/rfc3986>.

   [RFC4012] Blunk, L., Damas, J., Parent, F., and A. Robachevsky,
              "Routing Policy Specification Language next generation
              (RPSLng)", RFC 4012, DOI 10.17487/RFC4012, March 2005,
              <http://www.rfc-editor.org/info/rfc4012>.

   [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
              (CIDR): The Internet Address Assignment and Aggregation
              Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August
              2006, <http://www.rfc-editor.org/info/rfc4632>.

   [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
              Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
              <http://www.rfc-editor.org/info/rfc4648>.

   [RFC5396] Huston, G. and G. Michaelson, "Textual Representation of
              Autonomous System (AS) Numbers", RFC 5396,
              DOI 10.17487/RFC5396, December 2008,
              <http://www.rfc-editor.org/info/rfc5396>.

   [RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI
              Scheme", RFC 5781, DOI 10.17487/RFC5781, February 2010,
              <http://www.rfc-editor.org/info/rfc5781>.

   [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
              Address Text Representation", RFC 5952,
              DOI 10.17487/RFC5952, August 2010,
              <http://www.rfc-editor.org/info/rfc5952>.

   [RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for
              Resource Certificate Repository Structure", RFC 6481,
              DOI 10.17487/RFC6481, February 2012,
              <http://www.rfc-editor.org/info/rfc6481>.

   [RFC6485] Huston, G., "The Profile for Algorithms and Key Sizes for
              Use in the Resource Public Key Infrastructure (RPKI)",
              RFC 6485, DOI 10.17487/RFC6485, February 2012,
              <http://www.rfc-editor.org/info/rfc6485>.

   [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for
              X.509 PKIX Resource Certificates", RFC 6487,
              DOI 10.17487/RFC6487, February 2012,
              <http://www.rfc-editor.org/info/rfc6487>.

   [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Message Syntax and Routing",
              RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <http://www.rfc-editor.org/info/rfc7230>.

7.2. Informative References

   [RFC2725] Villamizar, C., Alaettinoglu, C., Meyer, D., and S.
              Murphy, "Routing Policy System Security", RFC 2725,
              DOI 10.17487/RFC2725, December 1999,
              <http://www.rfc-editor.org/info/rfc2725>.

   [RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed.,
              "DomainKeys Identified Mail (DKIM) Signatures", STD 76,
              RFC 6376, DOI 10.17487/RFC6376, September 2011,
              <http://www.rfc-editor.org/info/rfc6376>.

Authors' Addresses

   Robert Kisteleki
   RIPE NCC

   Email: robert@ripe.net
   URI: http://www.ripe.net

   Brian Haberman
   Johns Hopkins University Applied Physics Lab

   Email: brian@innovationslab.net