              Kerberos Authorization Data Container Authenticated by Multiple
                     Message Authentication Codes (MACs)

Abstract

   This document specifies a Kerberos authorization data container that
   supersedes AD-KDC-ISSUED. It allows for multiple Message
   Authentication Codes (MACs) or signatures to authenticate the
   contained authorization data elements. The multiple MACs are needed
   to mitigate shortcomings in the existing AD-KDC-ISSUED container.
   This document updates RFC 4120.

Table of Contents

1. Introduction

   This document specifies a new authorization data container for
   Kerberos, called the CAMMAC (Container Authenticated by Multiple
   MACs). The Abstract Syntax Notation One (ASN.1) type implementing
   the CAMMAC concept is the AD-CAMMAC, which supersedes the AD-KDC-
   ISSUED authorization data type specified in [RFC4120]. This new
   container allows both the receiving application service and the Key
   Distribution Center (KDC) itself to verify the authenticity of the
   contained authorization data. The AD-CAMMAC container can also
   include additional verifiers that "trusted services" can use to
   verify the contained authorization data.

2. Requirements Language

   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].

3. Motivations

   The Kerberos protocol allows clients to submit arbitrary
   authorization data for a KDC to insert into a Kerberos ticket. These
   client-requested authorization data allow the client to express
   authorization restrictions that the application service will
   interpret. With few exceptions, the KDC can safely copy these
   client-requested authorization data to the issued ticket without
   necessarily inspecting, interpreting, or filtering their contents.

   The AD-KDC-ISSUED authorization data container specified in RFC 4120
   [RFC4120] is a means for KDCs to include positive or permissive
   (rather than restrictive) authorization data in service tickets in a
   way that the service named in a ticket can verify that the KDC has

issued the contained authorization data. This capability takes
advantage of a shared symmetric key between the KDC and the service
to assure the service that the KDC did not merely copy client-
requested authorization data to the ticket without inspecting them.

The AD-KDC-ISSUED container works well for situations where the flow
of authorization data is from the KDC to the service. However,
protocol extensions such as Constrained Delegation (S4U2Proxy
[MS-SFU]) require that a service present to the KDC a service ticket
that the KDC previously issued, as evidence that the service is
authorized to impersonate the client principal named in that ticket.
In the S4U2Proxy extension, the KDC uses the evidence ticket as the
basis for issuing a derivative ticket that the service can then use
to impersonate the client. The authorization data contained within
the evidence ticket constitute a flow of authorization data from the
application service to the KDC. The properties of the AD-KDC-ISSUED
container are insufficient for this use case because the service
knows the symmetric key for the checksum in the AD-KDC-ISSUED
container. Therefore, the KDC has no way to detect whether the
service has tampered with the contents of the AD-KDC-ISSUED container
within the evidence ticket.

The new AD-CAMMAC authorization data container specified in this
document improves upon AD-KDC-ISSUED by including additional verifier
elements. The svc-verifier (service verifier) element of the
AD-CAMMAC has the same functional and security properties as the
ad-checksum element of AD-KDC-ISSUED; the svc-verifier allows the
service to verify the integrity of the AD-CAMMAC contents as it
already could with the AD-KDC-ISSUED container. The kdc-verifier and
other-verifiers elements are new to AD-CAMMAC and provide its
enhanced capabilities.

The kdc-verifier element of the AD-CAMMAC container allows a KDC to
verify the integrity of authorization data that it previously
inserted into a ticket by using a key that only the KDC knows. The
KDC thus avoids recomputing all of the authorization data for the
issued ticket; this recomputation might not always be possible when
that data includes ephemeral information such as the strength or type
of authentication method used to obtain the original ticket.

The verifiers in the other-verifiers element of the AD-CAMMAC
container are not required but can be useful when a lesser-privileged
service receives a ticket from a client and needs to extract the
AD-CAMMAC to demonstrate to a higher-privileged "trusted service" on
the same host that it is legitimately acting on behalf of that
client. The trusted service can use a verifier in the
other-verifiers element to validate the contents of the AD-CAMMAC
without further communication with the KDC.

4. Encoding

   The Kerberos protocol is defined in [RFC4120] using ASN.1 [X.680] and
   using the ASN.1 Distinguished Encoding Rules (DER) [X.690]. For
   consistency, this specification also uses ASN.1 for specifying the
   layout of AD-CAMMAC. The ad-data of the AD-CAMMAC authorization data
   element is the ASN.1 DER encoding of the AD-CAMMAC ASN.1 type
   specified below.

```
   KerberosV5CAMMAC {
           iso(1) identified-organization(3) dod(6) internet(1)
           security(5) kerberosV5(2) modules(4) cammac(7)
   } DEFINITIONS EXPLICIT TAGS ::= BEGIN

   IMPORTS
         AuthorizationData, PrincipalName, Checksum, UInt32, Int32
           FROM KerberosV5Spec2 { iso(1) identified-organization(3)
             dod(6) internet(1) security(5) kerberosV5(2)
             modules(4) krb5spec2(2) };
             -- as defined in RFC 4120.

   AD-CAMMAC                   ::= SEQUENCE {
         elements              [0] AuthorizationData,
         kdc-verifier          [1] Verifier-MAC OPTIONAL,
         svc-verifier          [2] Verifier-MAC OPTIONAL,
         other-verifiers       [3] SEQUENCE (SIZE (1..MAX))
                                   OF Verifier OPTIONAL
   }

   Verifier              ::= CHOICE {
         mac               Verifier-MAC,
         ...
   }

   Verifier-MAC          ::= SEQUENCE {
         identifier    [0] PrincipalName OPTIONAL,
         kvno          [1] UInt32 OPTIONAL,
         enctype       [2] Int32 OPTIONAL,
         mac           [3] Checksum
   }

   END
```

   elements:
      A sequence of authorization data elements issued by the KDC.
      These elements are the authorization data that the verifier fields
      authenticate.

Verifier:
   A CHOICE type that currently contains only one alternative:
   Verifier-MAC. Future extensions might add support for public-key
   signatures.

Verifier-MAC:
   Contains an RFC 3961 [RFC3961] checksum (MAC) computed over the
   ASN.1 DER encoding of the AuthorizationData value in the elements
   field of the AD-CAMMAC. The identifier, kvno, and enctype fields
   help the recipient locate the key required for verifying the MAC.
   For the kdc-verifier and the svc-verifier, the identifier, kvno,
   and enctype fields are often obvious from context and MAY be
   omitted. For the kdc-verifier, the MAC is computed differently
   than for the svc-verifier and the other-verifiers, as described
   later. The key usage number for computing the MAC (checksum) is
   64.

kdc-verifier:
   A Verifier-MAC where the key is a long-term key of the local
   Ticket-Granting Service (TGS). The checksum type is the required
   checksum type for the enctype of the TGS key. In contrast to the
   other Verifier-MAC elements, the KDC computes the MAC in the kdc-
   verifier over the ASN.1 DER encoding of a modified version of the
   EncTicketPart of the surrounding ticket. To construct this
   modified version of the EncTicketPart, the KDC replaces the
   AuthorizationData value that would have appeared in the
   authorization-data field of the EncTicketPart with the
   AuthorizationData value from the elements field of the AD-CAMMAC.
   The original authorization-data field in the EncTicketPart would
   have contained the AD-CAMMAC itself, possibly accompanied by other
   authorization data outside of the AD-CAMMAC. This altered
   Verifier-MAC computation binds the kdc-verifier to the other
   contents of the ticket, assuring the KDC that a malicious service
   has not substituted a mismatched AD-CAMMAC received from another
   ticket.

svc-verifier:
   A Verifier-MAC where the key is the same long-term service key
   that the KDC uses to encrypt the surrounding ticket. The checksum
   type is the required checksum type for the enctype of the service
   key used to encrypt the ticket. This field MUST be present if the
   service principal of the ticket is not the local TGS, including
   when the ticket is a cross-realm Ticket-Granting Ticket (TGT).

   other-verifiers:
      A sequence of additional verifiers. In each additional Verifier-
      MAC, the key is a long-term key of the principal name specified in
      the identifier field. The PrincipalName MUST be present and be a
      valid principal in the realm. KDCs MAY add one or more "trusted
      service" verifiers. Unless otherwise administratively configured,
      the KDC SHOULD determine the "trusted service" principal name by
      replacing the service identifier component of the sname element of
      the surrounding ticket with "host". The checksum is computed
      using a long-term key of the identified principal, and the
      checksum type is the required checksum type for the enctype of
      that long-term key. The kvno and enctype SHOULD be specified to
      disambiguate which of the long-term keys of the trusted service is
      used.

5. Usage

   Application servers and KDCs MAY ignore the AD-CAMMAC container and
   the authorization data elements it contains. For compatibility with
   older Kerberos implementations, a KDC issuing an AD-CAMMAC SHOULD
   enclose it in an AD-IF-RELEVANT container [RFC4120] unless the KDC
   knows that the application server is likely to recognize it.

6. Assigned Numbers

   RFC 4120 is updated in the following ways:

   o The ad-type number 96 is assigned for AD-CAMMAC, updating the
      table in Section 7.5.4 of [RFC4120].

   o The table in Section 5.2.6 of [RFC4120] is updated to map the ad-
      type 96 to "DER encoding of AD-CAMMAC".

   o The key usage number 64 is assigned for the Verifier-MAC checksum,
      updating the table in Section 7.5.1 of [RFC4120].

7. Security Considerations

   The CAMMAC provides data origin authentication for authorization data
   contained in it, attesting that it originated from the KDC. This
   section describes the precautions required to maintain the integrity
   of that data origin authentication through various information flows
   involving a Kerberos ticket containing a CAMMAC.

   When handling a TGS-REQ containing a CAMMAC, a KDC makes a policy
   decision on how to produce the CAMMAC contents of the newly issued
   ticket based on properties of the ticket(s) accompanying the TGS-REQ.
   This policy decision can involve filtering, transforming, or verbatim

copying of the original CAMMAC contents. The following paragraphs
provide some guidance on formulating such policies.

A KDC verifies a CAMMAC as originating from a local-realm KDC when at
least one of following the criteria is true:

1. the KDC successfully verifies the kdc-verifier; or

2. the KDC successfully verifies the svc-verifier, and the svc-
    verifier uses a key known only to the local-realm KDCs; or

3. no verifiers are present, the ticket-encrypting key is known only
    to local-realm KDCs, and all local-realm KDCs properly filter out
    client-submitted CAMMACs. (This can require particular caution
    in a realm that has KDCs with mixed CAMMAC support, as might
    happen when incrementally upgrading KDCs in a realm to support
    CAMMAC.)

A CAMMAC that originates from a local-realm KDC might contain
information that originates from elsewhere. Originating from a
local-realm KDC means that a local-realm KDC attests that the CAMMAC
contents conform to the policies of the local realm, regardless of
the ultimate origin of the information in the CAMMAC (which could be
a remote realm in the case of a CAMMAC contained in a cross-realm
TGT).

Local policy determines when a KDC can apply a kdc-verifier to a
CAMMAC (or otherwise creates a CAMMAC that satisfies the local-origin
criteria listed above). Semantically, a CAMMAC that a KDC verifies
as originating from a local-realm KDC attests that the CAMMAC
contents conformed to local policy at the time of creation of the
CAMMAC. Such a local policy can include allowing verbatim copying of
CAMMAC contents from cross-realm TGTs from designated remote realms
and applying a kdc-verifier to the new CAMMAC.

Usually, when a KDC verifies a CAMMAC as originating from a local-
realm KDC, the KDC can assume that the CAMMAC contents continue to
conform to the policies of the local realm. It is generally safe for
a KDC to make verbatim copies of the contents of such a CAMMAC into a
new CAMMAC when handling a TGS-REQ. Particularly strict
implementations might conduct additional policy checks on the
contents of a CAMMAC originating from a local-realm KDC if the policy
of the local realm materially changes during the life of the CAMMAC.

A KDC MAY omit the kdc-verifier from the CAMMAC when it is not needed
according to how realm policies will subsequently treat the
containing ticket. An implementation might choose to do this

omission to reduce the size of tickets it issues. Some examples of
when such an omission is safe are:

1. For a local-realm TGT, if all local-realm KDCs correctly filter
    out client-submitted CAMMACs, the local-realm origin criteria
    listed above allow omission of the kdc-verifier.

2. An application service might not use the S4U2Proxy extension, or
    the realm policy might disallow the use of S4U2Proxy by that
    service. In such situations where there is no flow of
    authorization data from the service to the KDC, the application
    service could modify the CAMMAC contents, but such modifications
    would have no effect on other services. Because of the lack of
    security impact to other application services, the KDC MAY omit
    the kdc-verifier from a CAMMAC contained in a ticket for that
    service.

Extracting a CAMMAC from a ticket for use as a credential removes it
from the context of the ticket. In the general case, this could turn
it into a bearer token, with all of the associated security
implications. Also, the CAMMAC does not itself necessarily contain
sufficient information to identify the client principal. Therefore,
application protocols that rely on extracted CAMMACs might need to
duplicate a substantial portion of the ticket contents and include
that duplicated information in the authorization data contained
within the CAMMAC. The extent of this duplication would depend on
the security properties required by the application protocol.

The method for computing the kdc-verifier binds it only to the
authorization data contained within the CAMMAC; it does not bind the
CAMMAC to any authorization data within the containing ticket but
outside the CAMMAC. At least one (non-standard) authorization data
type, AD-SIGNEDPATH, attempts to bind to other authorization data in
a ticket, and it is very difficult for two such authorization data
types to coexist.

The kdc-verifier in CAMMAC does not bind the service principal name
to the CAMMAC contents because the service principal name is not part
of the EncTicketPart. An entity that has access to the keys of two
different service principals can decrypt a ticket for one service and
encrypt it in the key of the other service, altering the svc-verifier
to match. Both the kdc-verifier and the svc-verifier would still
validate, but the KDC never issued this fabricated ticket. The
impact of this manipulation is minor if the CAMMAC contents only
communicate attributes related to the client. If an application
requires an authenticated binding between the service principal name
and the CAMMAC or ticket contents, the KDC MUST include in the CAMMAC
some authorization data element that names the service principal.

8. References

8.1. Normative References

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

   [RFC3961] Raeburn, K., "Encryption and Checksum Specifications for
              Kerberos 5", RFC 3961, DOI 10.17487/RFC3961, February
              2005, <http://www.rfc-editor.org/info/rfc3961>.

   [RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The
              Kerberos Network Authentication Service (V5)", RFC 4120,
              DOI 10.17487/RFC4120, July 2005,
              <http://www.rfc-editor.org/info/rfc4120>.

   [X.680]   ITU-T, "Information technology -- Abstract Syntax Notation
              One (ASN.1): Specification of basic notation", ITU-T
              Recommendation X.680, ISO/IEC International Standard
              8824-1:2008, November 2008.

   [X.690]   ITU-T, "Information technology -- ASN.1 encoding rules:
              Specification of Basic Encoding Rules (BER), Canonical
              Encoding Rules (CER) and Distinguished Encoding Rules
              (DER)", ITU-T Recommendation X.690, ISO/IEC International
              Standard 8825-1:2008, November 2008.

8.2. Informative References

   [MS-SFU]   Microsoft, "[MS-SFU]: Kerberos Protocol Extensions:
              Service for User and Constrained Delegation Protocol",
              October 2015,
              <http://msdn.microsoft.com/en-us/library/cc246071.aspx>.

Acknowledgements

Authors' Addresses

   Simo Sorce
   Red Hat

   Email: ssorce@redhat.com

   Tom Yu
   MIT

   Email: tlyu@mit.edu