Independent Submission
Request for Comments: 6717
Category: Informational
ISSN: 2070-1721

H. Hotz
Jet Propulsion Lab, Caltech
R. Allbery
Stanford University

August 2012

kx509 Kerberized Certificate Issuance Protocol in Use in 2012

Abstract

This document describes a protocol, called kx509, for using Kerberos tickets to acquire X.509 certificates. These certificates may be used for many of the same purposes as X.509 certificates acquired by other means, but if a Kerberos infrastructure already exists, then the overhead of using kx509 may be much less.

While not standardized, this protocol is already in use at several large organizations, and certificates issued with this protocol are recognized by the International Grid Trust Federation.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

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/rfc6717.

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Protocol Data 3
2.1. Request Packet 3
2.2. Reply Packet 4
3. Protocol Operation 7
4. Acknowledgements 8
5. IANA Considerations 8
6. Security Considerations 9
7. References 10
7.1. Normative References 10
7.2. Informative References 10
Appendix A. Certificate Caching and Deployment Considerations 12
Appendix B. Historic Extensions 12
Appendix C. Example Exchange 12

1. Introduction

The two primary ways of providing cryptographically secure identification on the Internet are Kerberos tickets [RFC4120] and X.509 [RFC5280] [X.509] certificates.

In practical IT infrastructure where both are in use, it's highly desirable to deploy their support in a way that guarantees they both authoritatively refer to the same entities. There is already a widely adopted standard for using X.509 certificates to acquire corresponding Kerberos tickets called Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) [RFC4556]. This document describes the kx509 protocol for supporting the symmetric operation of acquiring X.509 certificates using Kerberos tickets.

Preparing and reviewing this document exposed a number of issues that are discussed in the security considerations. Unfortunately, some of them can only be addressed with an incompatible upgrade to this protocol. The IETF's Kerberos working group has an expected work item to address these issues.

The International Grid Trust Federation [IGTF] supports the use of Short Lived Credential Services [SLCS] as a means to authenticate for resource usage based on other, native identity stores that an organization maintains. X.509 certificates issued using the kx509 protocol based on a Kerberos identity is one of the recognized credential services. The certificate profile for that use is outside the scope of this RFC but is described in [GRID-prof].

In normal operation, kx509 can be used after a Kerberos ticket-granting-ticket (TGT) is acquired, which is most likely during user login. First, the client generates an RSA public/private key pair. Next, using the Kerberos ticket-granting-ticket, it acquires a Kerberos service ticket for the KCA (Kerberized Certificate Authority) and uses this to send the public half of its key pair. The KCA will decrypt the service ticket, verify the integrity of the incoming packet, determine the identity of the user, and use the session key to send back a corresponding X.509 certificate.

1.1. 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].

2. Protocol Data

The protocol consists of a single request/reply exchange using UDP.

Both the request and the reply packet begin with four bytes of version ID information, followed by a DER-encoded ASN.1 message. The first two bytes of the version ID are reserved. They MUST be set to zero when sent and SHOULD be ignored when received. The third and fourth bytes are the major and minor version numbers, respectively. The version of the protocol described in this document is designated 2.0, so the first four bytes of the packet are 0, 0, 2, and 0.

Incompatible variations of this protocol MUST use a different major version number.

2.1. Request Packet

The request consists of a version ID, followed by a DER-encoded ASN.1 message containing a Kerberos AP-REQ, integrity check data on the request, and public key generated by the client. The ASN.1 encoding is:

```
KX509Request ::= SEQUENCE {
        AP-REQ  OCTET STRING,
        pk-hash OCTET STRING,
        pk-key  OCTET STRING
}
```

The AP-REQ is as described in [RFC4120], Section 5.5.1.

The pk-hash is Hashed Message Authentication Code (HMAC) using SHA-1 as the underlying hash. All 160 bits are sent. The key used is the Kerberos session key. The data to be hashed is the concatenation of the following:

- 4-byte version ID at the beginning of the packet.
- OCTET STRING of the AP-REQ.
- OCTET STRING of the pk-key.

The pk-key contains a public key. This key and its corresponding private key are generated by the client before contacting the server. Implementations of this protocol MUST support RSA keys, in which case the key is a DER-encoded RSAPublicKey as defined in [RFC3447], Section A.1.1, and then it is stored in this octet string in the request. Its encoding as an OCTET STRING starts with the 0x30 byte sequence at the beginning of a DER-encoded RSAPublicKey. Use of other public-key types is not defined.

Appendix C shows an example request packet.

2.2. Reply Packet

The reply consists of a version ID, followed by a DER-encoded ASN.1 message containing an error code, an optional authentication hash, optional certificate, and optional error text. The service SHOULD return replies of the same version as the request where possible.

```
KX509Response ::= SEQUENCE {
        error-code[0]  INTEGER DEFAULT 0,
        hash[1]        OCTET STRING OPTIONAL,
        certificate[2] OCTET STRING OPTIONAL,
        e-text[3]      VisibleString OPTIONAL
}
```

Although the format of the reply contains independently optional objects, the server MUST only generate replies with one of the following allowed combinations.

error-code	hash	certificate	
error-code			e-text
	hash	certificate	
	hash		e-text

The first case is returned when the server successfully generates a certificate for the user. The certificate is a DER-encoded certificate as defined in [RFC5280], Appendix A, page 116. Its encoding as an OCTET STRING starts with the 0x30 byte sequence that is at the beginning of a DER-encoded certificate.

The second case is returned when the server successfully authenticates the user and their key, but is unable for some other reason to generate a certificate.

The third case MAY be returned if the server is unable to successfully authenticate the user and intends to return some unauthenticated information to the client.

The hash on a response is computed using SHA-1 HMAC as for the request.

The data that is hashed is the concatenation of the following things:

- 4-byte version ID at the beginning of the packet.
- DER representation of the error-code exclusive of the tag and length, if it is present.
- OCTET STRING of the certificate, if it is present.
- VisibleString representation of the e-text exclusive of the tag and length, if it is present.

In other words, the hash is computed on the data in the fields that are present, exclusive of the overall ASN.1 wrapping.

The e-text MAY be translated into other character sets for display purposes, but the hash is computed on the e-text in its VisibleString representation. If the e-text contains NUL characters, the client MAY ignore any part of the error message after the first NUL character for display purposes.

As implied by the above table, if the reply does not contain a certificate, it MUST contain an error message and a non-zero error code. Conversely, if a certificate is returned, then the error-code MUST be zero. The server SHOULD use the DEFAULT encoding for a zero error-code value by omitting any explicit error-code from the reply.

The defined values for error-code are as follows:

error-code	Condition	Example
1	Permanent problem with client request	Incompatible version
2	Solvable problem with client request	Expired Kerberos credentials
3	Temporary problem with client request	Packet loss
4	Permanent problem with the server	Internal misconfiguration
5	Temporary problem with the server	Server overloaded

If a client error is returned, the client SHOULD NOT retry the request unless some remedial action is first taken, although if error-code 3 is returned, the client MAY retry with other servers before giving up.

If a server error is returned, it is RECOMMENDED that the client retry the request with a different server if one is known.

Since all KCAs serving a Kerberos realm are intended to be equivalent, in accordance with Section 4.1.2.2 of [RFC5280], the certificates returned from different KCAs serving the same Kerberos realm MUST NOT contain duplicate serial numbers.

This protocol and document do not address certificate verification or path construction. There are no provisions for returning any additional certificates that might be needed. Any application using a returned certificate must be configured independently to address these issues. An incompatible upgrade to this protocol will provide options to address this issue.

The returned certificate MUST identify the Kerberos client principal from the AP-REQ in the original KX509Request in the subject of the certificate or in a subjectAltName extension. The identification MUST be unique within the organization's deployed infrastructure. It is RECOMMENDED that a subjectAltName extension be included of type id-pkinit-san as described in [RFC4556], Section 3.2.2. Note that the id-pkinit-san is simply a standard representation of a Kerberos principal and has no other implications with respect to PKINIT.

Other extensions MAY be added according to local policy.

Appendix C shows an example reply packet.

3. Protocol Operation

Absent errors, the protocol consists of a single request, sent via UDP, and a single reply, also sent via UDP.

There is no special provision for requests or replies that exceed the allowable size of a UDP packet. Also, some implementations have imposed hard size limits that are smaller than a typical UDP MTU and will limit the use of extensions and the supportable key size. Even without hard limits, if the request or reply exceeds the MTU size of a UDP packet for the infrastructure in use, then the reliability of the exchange will decrease significantly.

For "normal" Kerberos AP-REQ structures, and "normal" X.509 certificates, this is unlikely unless the Kerberos service ticket contains large amounts of authorization data. For this reason, it is RECOMMENDED that service tickets for the KCA be issued without authorization data. If the KCA performs authorization, it should do so by other means.

Before constructing the request, the client must know the canonical name(s) and port(s) of the server(s) to contact. It MAY determine them by looking up the service's SRV record as described in [RFC2782]. The entry to be used is _kca._udp._realm_, where _realm_ is the Kerberos realm, used as part of the DNS name.

The client has to acquire a service ticket in order to construct the AP-REQ for the service. Conventionally, the Kerberos service principal name to use for this service has a first component of "kca_service". Absent local configuration or other external knowledge of the correct principal to use, the second and final component is conventionally the canonical name of the KCA server being contacted, and the realm of the principal is determined following normal Kerberos domain-to-realm mapping conventions, as discussed in [RFC4120], Section 1.3.

When the server receives a request, it MUST verify the following properties of the request before issuing a certificate:

- The AP-REQ can be decoded and is not expired.
- If the request uses cross-realm authentication, then it satisfies the requirements of local policy and [RFC4120], Sections 1.2 and 2.7.

- The request's hash is valid.

The server SHOULD make other sanity checks, such as a minimum public key length, to the extent feasible.

The server MAY decline to respond to an erroneous request. If it does not receive a response, a client MAY retry its request, but the client SHOULD wait at least one second before doing so.

The client MUST verify any hash in the reply and MUST NOT use any certificate in a reply whose hash does not verify. The client MAY display the e-text if the hash is absent or does not verify but SHOULD indicate the message is not authenticated.

4. Acknowledgements

The original version of kx509 was implemented using Kerberos 4 at the University of Michigan and was nicely documented in [KX509]. Many thanks to them for their original work, as well as the subsequent updates.

While developing this document, important corrections and comments were provided by Jeffrey Altman and Love Hornquist Astrand. The following people also provided many helpful comments and corrections: Doug Engert, Jeffrey Hutzelman, Sam Hartman, Timothy J. Miller, Chaskiel Grundman, and Jim Schaad. Alan Sill provided the references to the International Grid Trust Federation and its acceptable credential services. Example network traffic was provided by Doug Engert, Marcus Watts, Matt Crawford, and Chaskiel Grundman from their deployments and was extremely useful for verifying the reality of this specification.

5. IANA Considerations

This service is conventionally run on UDP port 9878. IANA has registered that port in the Service Name and Transport Port Number Registry as follows:

Service Name:	kca-service
Transport Protocol:	UDP
Assignee:	IESG <iesg@ietf.org>
Contact:	IETF Chair <chair@ietf.org>
Description:	The kx509 Kerberized Certificate Issuance Protocol in Use in 2012
Reference:	RFC 6717
Port Number:	9878
Assignment Notes:	Historically, this service has been referred to as "kca_service", but this service name does

not meet the registry requirements.

The Generic Security Service Application Program Interface (GSS-API) / Kerberos / Simple Authentication and Security Layer (SASL) service name currently in use for this protocol is "kca_service". This does not meet the naming requirements for IANA's GSS-API/Kerberos/SASL service name registry, so no registration has been requested. The conflict between the conventional service name and the registry rules is expected to be addressed in a future version of this protocol. Appropriate registrations will be requested at that time.

6. Security Considerations

The only encrypted information in the protocol is that used by Kerberos itself. The considerations for any Kerberized service apply here.

The public key in the request is sent in the clear and without any guarantees that the requester actually possesses the corresponding private key. Therefore, the only appropriate uses of the returned certificate are those where the identity of the requester is unimportant or the subsequent use independently guarantees that the user possesses the private key. This issue is expected to be addressed in a future version of this protocol.

For example, if the kx509-issued certificate is used for a digital signature in a way that does not independently demonstrate proof-of-possession of the private key, then an eavesdropper could request their own valid certificate via kx509 and claim to have originated material signed by the legitimate, original requester. [RFC4211], Appendix C, contains a more detailed discussion of why proof-of-possession is important.

An example that should be safe is initial client authentication with Transport Layer Security (TLS) [RFC5246] connections. If a client certificate is used, then a Certificate Verify message (Section 7.4.8 of [RFC5246]) is added to the handshake exchange. It includes a signature of the handshake messages to that point. Those messages depend on data known only to the client and server ends of the specific connection, so computing the signature proves possession of the private key. This application was the original intended use case for kx509.

Some information, such as the public key and certificate, is transmitted in the clear but (as the name implies) is generally intended to be publicly available. However, its visibility could still raise privacy concerns. The hash is used to protect the integrity of this information.

The policies for issuing Kerberos tickets and X.509 certificates are usually expressed very differently. An implementation of this protocol should not provide a mechanism for bypassing ticket or certificate policies.

In particular, if the issued certificate can be used with PKINIT, this authentication loop SHOULD NOT bypass policy limits for either X.509 certificates or Kerberos tickets.

X.509 certificates are usually issued with considerably longer validity times than Kerberos tickets. Care should be taken that the issued certificate is not valid for longer than the intended policy should allow. Note that Section 3.2.3 of [RFC4556] REQUIRES that the lifetime of an issued ticket not exceed the lifetime of the predecessor certificate. By analogy, it is RECOMMENDED that the lifetime of an issued certificate not exceed the lifetime of the predecessor Kerberos ticket unless the implications with respect to local policy and supporting infrastructure are clearly understood and allow it.

7. References

7.1. Normative References

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

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[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.

7.2. Informative References

[GRID-prof] "Grid Certificate Profile", March 2008, <http://www.ogf.org/documents/GFD.125.pdf>.

[IGTF] "The International Grid Trust Federation", <http://www.igtf.net/>.

[KX509] Doster, W., Watts, M., and D. Hyde, "The KX.509 Protocol", September 2001, <http://www.citi.umich.edu/techreports/reports/citi-tr-01-2.pdf>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, September 2005.

[RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[SLCS] "Short Lived Credential Services", February 2009, <http://tagpma.org/authn_profiles/slcs>.

[X.509] International Telecommunications Union, "Recommendation X.509: The Directory: Public-key and attribute certificate framework", November 2008.

Appendix A. Certificate Caching and Deployment Considerations

As noted in the Security Considerations section, the functional lifetime of the acquired X.509 certificate should usually match the lifetime of its predecessor Kerberos ticket. Therefore, it is likely that X.509 certificates issued with this protocol should be deleted when the supporting Kerberos tickets are deleted. That makes the Kerberos ticket cache a reasonable location to store the certificate (and its private key).

On the other hand, applications, such as web browsers, probably expect certificates in different stores.

A widely used solution to this dichotomy is to implement a PKCS11 library that supports the kx509-acquired credentials. The credentials remain stored in the Kerberos credentials cache, but full PKI functionality is still available via a standard interface for PKI credentials.

Appendix B. Historic Extensions

This appendix documents extensions to the kx509 protocol that are either no longer in use or are expected to be dropped.

A subjectAltName othername extension of type kcaAuthRealm (OID value 1.3.6.1.4.1.250.42.1) is frequently used to include the client's realm as an ASN.1 octet string.

The Microsoft-defined userPrincipalName has frequently been used for the same purpose as the id-pkinit-san.

The historic implementations of this protocol included provisions for DSA keys in place of RSA. DSA does not appear to be in use. A future version of this protocol will use a standard certificate request structure that will provide algorithm agility.

The historic implementations of this protocol allowed an optional client-version field (at the end of the request) of type VisibleString. If present, the KCA copied it into the issued certificate as an extension with the OID 1.3.6.1.4.1.250.42.2. This feature does not appear to be in use.

Appendix C. Example Exchange

The request and reply are from the same exchange. The Ethernet, IP, and UDP headers, and the 4-byte version string at the beginning of the request and reply packets are all omitted. Only the ASN.1-encoded portions are shown.

```
  0:d=0  hl=4 l= 678 cons: SEQUENCE
  4:d=1  hl=4 l= 509 prim:  OCTET STRING
                       [HEX DUMP]:6E8201F9308201F5A003... (AP-REQ)
517:d=1  hl=2 l=  20 prim:  OCTET STRING
                       [HEX DUMP]:ECFF1C922300D0E9DD02... (pk-hash)
539:d=1  hl=3 l= 140 prim:  OCTET STRING
                       [HEX DUMP]:30818902818100B70F46... (pk-key)
```

Request Packet ASN.1 Decode

```
 0:d=0  hl=4 l= 870 cons: SEQUENCE
 4:d=1  hl=2 l=  22 cons:  cont [ 1 ]
 6:d=2  hl=2 l=  20 prim:   OCTET STRING
                    [HEX DUMP]:F3A844834C26D843B6FD... (hash)
28:d=1  hl=4 l= 842 cons:  cont [ 2 ]
32:d=2  hl=4 l= 838 prim:   OCTET STRING
                    [HEX DUMP]:308203423082022AA003... (certificate)
```

Reply Packet ASN.1 Decode

Authors' Addresses

Henry B. Hotz
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
USA

Phone: +01 818 354-4880
EMail: hotz@jpl.nasa.gov

Russ Allbery
Stanford University
P.O. Box 20066
Stanford, CA 94309
USA

EMail: rra@stanford.edu
URI: http://www.eyrie.org/~eagle/