INTERNET-DRAFT
draft-ietf-cat-kerberos-pk-tapp-04.txt
Expires May 9, 2001
Informational

Ari Medvinsky
Keen.com, Inc.
Matthew Hur
Cisco Systems
Sasha Medvinsky
Motorola
Clifford Neuman
USC/ISI

Public Key Utilizing Tickets for Application Servers (PKTAPP)

0. Status Of this Memo

The distribution of this memo is unlimited. It is filed as draft-ietf-cat-kerberos-pk-tapp-04.txt, and expires May 9, 2001. **Please send comments to the authors.**

1. Abstract

Public key based Kerberos for Distributed Authentication[1], (PKDA) proposed by Sirbu & Chuang, describes PK based authentication that eliminates the use of a centralized key distribution center while retaining the advantages of Kerberos tickets. This draft describes how, without any modification, the PKINIT specification[2] may be used to implement the ideas introduced in PKDA. The benefit is that only a single PK Kerberos extension is needed to address the goals of PKINIT & PKDA.

2. Introduction

With the proliferation of public key cryptography, a number of public key extensions to Kerberos have been proposed to provide interoperability with the PK infrastructure and to improve the Kerberos authentication system [4]. Among these are PKINIT[2] (under development in the CAT working group) and more recently PKDA [1] proposed by Sirbu & Chuang of CMU. One of the principal goals of PKINIT is to provide for interoperability between a PK infrastructure and Kerberos. Using PKINIT, a user can authenticate to the KDC via a public key certificate. A ticket granting ticket (TGT), returned by the KDC, enables a PK user to obtain tickets and authenticate to kerberized services. The PKDA proposal goes a step further. It supports direct client to server authentication, eliminating the need for an online key distribution center. In this draft, we describe how, without any modification, the PKINIT protocol may be applied to achieve the goals of PKDA. For direct client to server authentication, the client will use PKINIT to authenticate to the end server (instead of a central KDC), which then, will issue a ticket for itself. The benefit of this proposal, is that a single PK extension to Kerberos can addresses the goals of PKINIT and PKDA.

3. PKDA background

The PKDA proposal provides direct client to server authentication, thus eliminating the need for an online key distribution center. A client and server take part in an initial PK based authentication exchange, with an added caveat that the server acts as a Kerberos ticket granting service and issues a traditional Kerberos ticket for itself. In subsequent communication, the client makes use of the Kerberos ticket, thus eliminating the need for public key operations on the server. This approach has an advantage over SSL in that the server does not need to save state (cache session keys). Furthermore, an additional benefit, is that Kerberos tickets can facilitate delegation (see Neuman[3]).

Below is a brief overview of the PKDA protocol. For a more detailed description see [1].

SCERT_REQ: Client to Server
The client requests a certificate from the server. If the serverÆs certificate is cached locally, SCERT_REQ and SCERT_REP are omitted.

SCERT_REP: Server to Client
The server returns its certificate to the client.

PKTGS_REQ: Client to Server
The client sends a request for a service ticket to the server. To authenticate the request, the client signs, among other fields, a time

stamp and a newly generated symmetric key . The time stamp is used to foil replay attacks; the symmetric key is used by the server to secure the PKTGS_REP message.
The client provides a certificate in the request (the certificate enables the server to verify the validity of the clientÆs signature) and seals it along with the signed information using the serverÆs public key.

PKTGS_REP: Server to Client
The server returns a service ticket (which it issued for itself) along with the session key for the ticket. The session key is protected by the client-generated key from the PKTGS_REQ message.

AP_REQ: Client to Server
After the above exchange, the client can proceed in a normal fashion, using the conventional Kerberos ticket in an AP_REQ message.

4. PKINIT background

One of the principal goals of PKINIT is to provide for interoperability between a public key infrastructure and Kerberos. Using a public key certificate, a client can authenticate to the KDC and receive a TGT which enables the client to obtain service tickets to kerberized services.. In PKINIT, the AS-REQ and AS-REP messages remain the same; new preauthentication data types are used to conduct the PK exchange. Client and server certificates are exchanged via the preauthentication data. Thus, the exchange of certificates , PK authentication, and delivery of a TGT can occur in two messages.

Below is a brief overview of the PKINIT protocol. For a more detailed description see [2].

PreAuthentication data of AS-REQ: Client to Server
The client sends a list of trusted certifiers, a signed PK authenticator, and its certificate. The PK authenticator, based on the Kerberos authenticator, contains the name of the KDC, a timestamp, and a nonce.

PreAuthentication data of AS-REP: Server to Client
The server responds with its certificate and the key used for decrypting the encrypted part of the AS-REQ. This key is encrypted with the clientÆs public key.

AP_REQ: Client to Server
After the above exchange, the client can proceed in a normal fashion, using the conventional Kerberos ticket in an AP_REQ message.

5. Application of PKINIT to achieve equivalence to PKDA

While PKINIT is normally used to retrieve a ticket granting ticket (TGT), it may also be used to request an end service ticket. When used in this fashion, PKINIT is functionally equivalent to PKDA. We introduce the concept of a local ticket granting server (LTGS) to illustrate how PKINIT may be used for issuing end service tickets based on public key authentication. It is important to note that the LTGS may be built into an application server, or it may be a stand-alone server used for issuing tickets within a well-defined realm, such as a single machine. We will discuss both of these options.

5.1. The LTGS

The LTGS processes the Kerberos AS-REQ and AS-REP messages with PKINIT preauthentication data. When a client submits an AS-REQ to the LTGS, it specifies an application server, in order to receive an end service ticket instead of a TGT.

5.1.1. The LTGS as a standalone server

The LTGS may run as a separate process that serves applications which reside on the same machine. This serves to consolidate administrative functions and provide an easier migration path for a heterogeneous environment consisting of both public key and Kerberos. The LTGS would use one well-known port (port #88 - same as the KDC) for all message traffic and would share a symmetric with each service. After the client receives a service ticket, it then contacts the application server directly. This approach is similar to the one suggested by Sirbu , et al [1].

5.1.1.1. Ticket Policy for PKTAPP Clients

It is desirable for the LTGS to have access to a PKTAPP client ticket policy. This policy will contain information for each client, such as the maximum lifetime of a ticket, whether or not a ticket can be forwardable, etc. PKTAPP clients, however, use the PKINIT protocol for authentication and are not required to be registered as Kerberos principals.

As one possible solution, each public key Certification Authority could be registered in a secure database, along with the ticket policy information for all PKTAPP clients that are certified by this Certification Authority.

5.1.1.2. LTGS as a Kerberos Principal

Since the LTGS serves only PKTAPP clients and returns only end service tickets for other services, it does not require a Kerberos service key or a Kerberos principal identity. It is therefore not necessary for the LTGS to even be registered as a Kerberos principal.

The LTGS still requires public key credentials for the PKINIT exchange, and it may be desired to have some global restrictions on the Kerberos tickets that it can issue. It is recommended (but not required) that this information be associated with a Kerberos principal entry for the LTGS.

5.1.1.3. Kerberos Principal Database

Since the LTGS issues tickets for Kerberos services, it will require access to a Kerberos principal database containing entries for at least the end services. Each entry must contain a service key and may also contain restrictions on the service tickets that are issued to clients. It is recommended that (for ease of administration) this principal database be centrally administered and distributed (replicated) to all hosts where an LTGS may be running.

In the case that there are other clients that do not support PKINIT protocol, but still need access to the same Kerberos services, this principal database will also require entries for Kerberos clients and for the TGS entries.

5.1.2. The LTGS as part of an application server

The LTGS may be combined with an application server. This accomplishes direct client to application server authentication; however, it requires that applications be modified to process AS-REQ and AS-REP messages. The LTGS would communicate over the port assigned to the application server or over the well known Kerberos port for that particular application.

5.1.2.2. Ticket Policy for PKTAPP Clients

Application servers normally do not have access to a distributed principal database. Therefore, they will have to find another means of keeping track of the ticket policy information for PKTAPP clients. It is recommended that this ticket policy be kept in a directory service (such as LDAP).

It is critical, however, that both read and write access to this ticket policy is restricted with strong authentication and encryption to only the correct application server. An unauthorized party should not have the authority to modify the ticket policy. Disclosing the ticket policy to a 3rd party may aid an adversary in determining the best way to compromise the network.

It is just as critical for the application server to authenticate the directory service. Otherwise an adversary could use a man-in-the-middle attack to substitute a false ticket policy with a false directory service.

5.1.2.3. LTGS Credentials

Each LTGS (combined with an application service) will require public key credentials in order to use the PKINIT protocol. These credentials can be stored in a single file that is both encrypted with a password-derived symmetric key and also secured by an operating system. This symmetric key may be stashed somewhere on the machine for convenience, although such practice potentially weakens the overall system security and is strongly discouraged.

For added security, it is recommended that the LTGS private keys are stored inside a temper-resistant hardware module that requires a pin code for access.

5.1.2.4. Compatibility With Standard Kerberos

Even though an application server is combined with the LTGS, for backward compatibility it should still accept service tickets that have been issued by the KDC. This will allow Kerberos clients that do not support PKTAPP to authenticate to the same application server (with the help of a KDC).

5.1.3. Cross-Realm Authentication

According to the PKINIT draft, the client's realm is the X.500 name of the Certification Authority that issued the client certificate. A Kerberos application service will be in a standard Kerberos realm, which implies that the LTGS will need to issue cross-realm end service tickets. This is the only case, where cross-realm end service tickets are issued. In a standard Kerberos model, a client first acquires a cross-realm TGT, and then gets an end service ticket from the KDC that is in the same realm as the application service.

6. Protocol differences between PKINIT and PKDA

Both PKINIT and PKDA will accomplish the same goal of issuing end service tickets, based on initial public key authentication. A PKINIT-based implementation and a PKDA implementation would be functionally equivalent. The primary differences are that 1)PKDA requires the client to create the symmetric key while PKINIT requires the server to create the key and 2)PKINIT accomplishes in two messages what PKDA accomplishes in four messages.

7. Summary

The PKINIT protocol can be used, without modification to facilitate client to server authentication without the use of a central KDC. The approach described in this draft (and originally proposed in PKDA[1]) is essentially a public key authentication protocol that retains the advantages of Kerberos tickets.

Given that PKINIT has progressed through the CAT working group of the

IETF, with plans for non-commercial distribution (via MITÆs v5 Kerberos) as well as commercial support, it is worthwhile to provide PKDA functionality, under the PKINIT umbrella.

8. Security Considerations

PKTAPP is based on the PKINIT protocol and all security considerations already listed in [2] apply here.

When the LTGS is implemented as part of each application server, the secure storage of its public key credentials and of its ticket policy are both a concern. The respective security considerations are already covered in sections 5.1.2.3 and 5.1.2.2 of this document.

9. Bibliography

[1] M. Sirbu, J. Chuang. Distributed Authentication in Kerberos Using Public Key Cryptography. Symposium On Network and Distributed System Security, 1997.

[2] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky, J. Wray, **J. Trostle. Public Key Cryptography for Initial Authentication in** Kerberos. Internet Draft, July 2000. (ftp://ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt)

[3] C. Neuman, Proxy-Based Authorization and Accounting for Distributed Systems. In Proceedings of the 13th International Conference on Distributed Computing Systems, May 1993.

[4] J. Kohl, C. Neuman. The Kerberos Network Authentication Service (V5). Request for Comments 1510.

10. Expiration Date

This draft expires May 9, 2001.

11. Authors

Ari Medvinsky
Keen.com, Inc.
150 Independence Dr.
Menlo Park, CA 94025
Phone +1 650 289 3134
E-mail: ari@keen.com

Matthew Hur
Cisco Systems
500 108th Ave. NE, Suite 500
Bellevue, WA 98004
Phone:
EMail: mhur@cisco.com

Sasha Medvinsky
Motorola
6450 **Sequence Dr.**
San Diego, CA 92121
Phone: +1 858 404 2367
E-mail: smedvinsky@gi.com

Clifford Neuman
USC Information Sciences Institute
4676 **Admiralty Way Suite 1001**
Marina del Rey CA 90292-6695
Phone: +1 310 822 1511
E-mail: bcn@isi.edu