Implementing a Kerberos Single Sign-on Infrastructure

Gary Tagg
IT Security Consultant,
Tagg Consulting Ltd
gary.tagg@itsecure.demon.co.uk

Abstract
Kerberos provides secure authentication, single sign-on and encryption for computer networks. This paper is written for IT managers currently considering a Kerberos strategy, and project managers tasked with implementing a Kerberos infrastructure. It provides a high level view of the Kerberos protocol, the security problems it can solve, and considerations for designing and implementing a Kerberos Infrastructure.

Introduction

Kerberos provides secure authentication, single sign-on and encryption for computer networks. It was developed at MIT to help secure their UNIX environment, but has recently gained prominence in the Microsoft Windows world with its integration into Windows 2000.

This paper is written for IT managers currently considering a Kerberos strategy, and Project Managers tasked with implementing a Kerberos infrastructure. It provides the following:

- A high level overview of Kerberos
- A description of the security problems that Kerberos can solve
- Designing a Kerberos infrastructure
- Implementation considerations

Resources

The starting point for anyone wanting to learn about Kerberos is the home page at MIT - http://web.mit.edu/kerberos/www/. From this page, follow the numerous links to other sites.

A must read is the comp.protocols.kerberos FAQ[1], which contains lots of links to other useful resources.

J. G. Steiner, B. Clifford Neuman, and J.I. Schiller give a good overview of the KERBEROS V4 protocol in their paper[2].

The Microsoft Web site has a good selection of white papers on Kerberos and its implementation in Windows 2000. See [3] for an overview and [4] for a white paper on Windows 2000 Kerberos Interoperability.

Terminology

Below is a list of terms used throughout this document.

Term	**Definition**
Authentication	Verifying the claimed identity of a principal.
Authentication Request	A message from a client to a server requesting access to a server. It consists of the Server Ticket issued by the TGS and a fresh authenticator.
Authentication Server	The Authentication Server authenticates a user or client and issues a Ticket Granting Ticket (TGT) that enables the user or client to request application server tickets.
Authorisation	The process of determining whether a client may use a service, which objects the client is allowed to access, and the type of access allowed for each.
Client	A process that makes use of a network service on behalf of a user.
Credential Cache	An area on the client computer where active Kerberos tickets are stored.
Host Principal	A Principal used by system servers such as Telnet and FTP daemons.
KDC	The Key Distribution Centre is a network service that authenticates users and supplies tickets and session keys.
Kerberos	Aside from the 3-headed dog

Term	Definition
	guarding Hades, it is the name given to the authentication service and protocol developed at MIT for project Athena.
Kerberised	A client or server program is “kerberised” if it uses the Kerberos system for authentication or encryption.
Principal	A principal is the Kerberos term for a uniquely named user, client or server instance that participates in a network communication.
Principal identifier	The name used to uniquely identify each different principal.
Secret key	An encryption key shared by a principal and the KDC.
Server	A particular Principal which provides a resource or service to network clients.
Session key	A temporary encryption key used between two principals, with a lifetime limited to the duration of a single log-on session.
Server Ticket	A message that helps a client authenticate itself to a server. It only serves to authenticate a client when presented along with a fresh Authenticator within an authentication request.
TGS	Ticket Granting Service. The TGS issues server tickets to clients.
TGT	Ticket Granting Ticket. The ticket issued by the Authentication Service that is used to request server tickets from the TGS.
User	A person who uses a program (e.g. client or service).

Kerberos – An overview

Background

The Kerberos system originated from MIT’s project Athena, to help secure MIT’s distributed network of computers. There have been two main releases of Kerberos, version 4 which was defined in the late 1980s and version 5 defined by RFC1510[5] in 1993. Versions 1-3 were internal development versions. Further development work is ongoing resulting in a number of recent draft RFCs.

Kerberos’s presence in the public domain for over ten years has enabled the protocol to be extensively reviewed creating a high level of confidence that the protocol is sound. Version 5 is largely a result of this public review process. Steven Bellovin and Michael Merritt prepared an excellent paper[6] describing the limitations of the protocol (mainly version 4) that directly influenced Version 5 during its development.

Kerberos Security functionality

The Kerberos protocol can provide the following security functionality.

- Secure authentication based upon a shared secret (typically a password) without sending the password in clear text over the network
- Secure authentication based upon smartcards and public key certificates, RSA’s Secur-ID token and other authentication mechanisms
- Single sign-on to kerberised applications
- Cryptographic strength data integrity
- Data encryption

Security issues addressed by Kerberos

The above security functionality can help address the following network security issues.

Clear text passwords travelling over network links

Most Unix users log-on to remote host systems via Telnet and transfer files using FTP. Both of these protocols require a user-id and password, which are sent in clear text across the network to the remote system. A network sniffer can capture these user-ids and passwords, allowing an attacker to log-in as that user. Kerberos resolves this issue by encrypting all critical authentication information.

Unattended Processes

The file transfer protocol (FTP) is a standard way to transfer files between systems. These FTP jobs need to store locally a user-id and password to authenticate to the remote system. These user-ids and passwords are typically stored in clear text within scripts or configuration files.

With a Kerberos system, an authentication file can be extracted and stored locally in a non-readable format removing the need for clear text passwords. These authentication files can be regularly changed via automated jobs.

Managing many user-ids and passwords

People use many IT systems in the course of their work and organisations are continually bringing new systems on-line. Many of these systems build their own user databases and management systems. This increases the development cost, as well as the password management overhead.

This problem is not limited to business users. In a typical UNIX IT department, system administrators manage a large number of hosts. To overcome the password management and repeated log-on problems, system administrators use the Unix “r” commands. This creates a security whole in the network allowing an attacker to move unchallenged from host to host.

Kerberos overcomes these issues by providing a centralised user database and authenticated single sign-on.

Modification of data

It is feasible for an attacker to capture data on route to another system, modify it and then send it on. Should this data be a financial transaction then there is the potential for profit from this type of attack. The Kerberos protocol includes a cryptographic hash and a sequence number that allows the remote system to detect any change to the data whilst traversing the network.

Loss of confidentiality

Many businesses have data that is confidential, and this data needs to travel over networks. For example, data can be sent from a user’s workstation to a server, or sent from one server to another for further processing. Attackers can use network sniffers to easily collect this confidential data.

The Kerberos protocol addresses this issue via an option to encrypt all data exchanged between a client and a server.

Introduction to the Kerberos protocol

There are already a number of very good papers on the protocol itself so this section limits itself to a high level overview which should be sufficient for the intended reader. However, if you would like more detailed information then Bill Bryant has written an excellent and imaginative paper[7] on how the protocol solves network security issues. For all the details then reference [2] provides a good description of version 4 of the protocol and RFC1510[5] defines version 5.

Kerberos is a trusted third party authentication service based upon the Needham and Schroeder model[8]. At the heart of the system is a central database of symmetric encryption keys with each Principal (user, client, server) having a shared secret (a symmetric key) with the system. This database and its supporting services are known as a Key Distribution Centre (KDC).

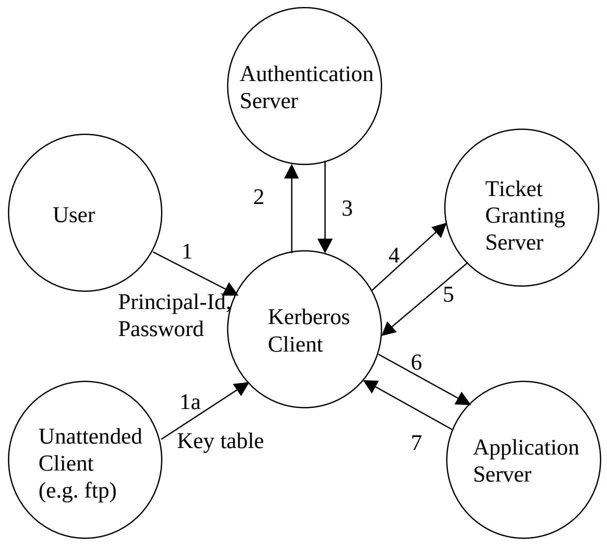
The system works by the KDC issuing a ticket that the client presents to the application server. This ticket contains time limited authentication information encrypted under the keys of the client and the application server. This enables the client and server to securely authenticate one another without passing clear text passwords across the network. This also means that no direct trust relationship is needed between principals. Both the client and server principals use their trust relationship with the KDC (the shared key) to authenticate one another and establish encrypted sessions.

Referring to figure 1 below, there are three main parts to the protocol. These are:

- User/Client authentication and the issuing by the Authentication Server of a special type of ticket known as a Ticket Granting Ticket(TGT) (1,2,3)
- The issuing of an Application Server ticket by the Ticket Granting Server (TGS) by presentation of the TGT (4,5)
- The client presents the Application Server ticket and other unique time dependent authentication information to the Application Server (6). The client also optionally authenticates the Application Server and establishes a sub-session key (7).

The entire message flow above (1-7) does not have to be followed for each authentication to an Application Server. Normally, steps 1-3 are done only when the user logs-on. The outcome of this message pair is the Ticket Granting Ticket (TGT), which provides access to the Ticket Granting Server. This TGT is stored locally in a credential cache.

When access to an Application Server is required, the Kerberos client sends its TGT and a fresh authenticator to the TGS. In this way, Kerberos can provide single sign-on without the need to store the user's password. The TGS returns an Application Server ticket that the client stores in its local credential cache before forwarding it to the Application Server. In this way, only messages 6&7 are needed when a user already has a valid ticket.



1. User enters his Principal-id and password
1a. The Kerberos client retrieves the Principal-id and authenticator (key) from a local key table
2. The Kerberos client sends an Authentication Request to the Authentication Server
3. The authentication Server returns a ticket for the Ticket Granting Server(TGS).
4. The Kerberos Client requests an application server ticket from the TGS
5. The TGS sends back an application server ticket.
6. The Kerberos client sends an authentication request to the application Server
7. The Application Server returns an authentication response to the Kerberos client

Figure 1 – The Kerberos Protocol

Realms

In common with most authentication systems, Kerberos is structured into named authentication domains known as realms. Kerberos V5 introduced cross realm authentication enabling the users in one realm to securely access resources in other realms. Kerberos supports two types of realm structure, one to one cross authentication and hierarchical cross authentication.

User authentication

Kerberos is based on the KDC and a user having a shared secret, which in the case of users, is a cryptographic transformation of the password and other information. In the main protocol, the password is not sent to the KDC to authenticate the user. Instead, the Principal-id is sent along with a request for a ticket for the TGS. The Authentication Server returns a ticket known as the TGT with parts of it encrypted under the shared secret. The user enters a password, and if it is correct, the TGT can be decrypted and used to obtain application server tickets from the TGS.

This protocol has two disadvantages, these being:

- The only possible authenticator is the password
- The KDC does not know about invalid passwords and therefore cannot lockout a principal when too many incorrect passwords have been entered.

To counter these disadvantages, a pre-authentication option was added to Kerberos V5. This allows users to demonstrate to the Authentication Server that they have the correct password before a TGT is issued. The Authentication Server can also use this information to manage an invalid password counter and lockout a principal when too many incorrect passwords have been entered.

This pre-authentication option also provides support for other authentication mechanisms such as smartcards, public key certificates, and token cards such as RSA's Secur-ID. When authenticated via these mechanisms, a Kerberos TGT is returned which the user submits to the TGS as normal to request Application Server tickets.

PKINIT

PKINIT is a draft IETF standard [9]defining how Kerberos can authenticate users via public key certificates. This extends Kerberos authentication to include smartcards as well as file based public key credentials. The current draft of PKINIT is supported by the Cybersafe and Microsoft Kerberos implementations.

Host Principals

Automated servers such as Telnet and FTP daemons also have a shared secret with the KDC and need to load it when the daemon starts. However, it is operationally impractical and insecure to have an operator manually enter a password when the daemon starts. Instead, Kerberos implementations provide a tool for generating a random shared secret, and extracting this into a binary key file that the daemon loads automatically at start-up. A good Kerberos implementation will also provide functionality for securely delegating the generation of these host principals to system administrators and the system itself. This allows the system to automatically change its key file in accordance with security policy.

Encryption

When a Kerberos session is started, there is an option to establish an encrypted session. This uses a session key generated by the KDC when the Application Server Ticket is issued. This session key is common to all sessions established between a client and server during the lifetime of the client's TGT. However, Kerberos can establish a sub-session key as part of the mutual authentication option.

Authentication versus Authorisation

Kerberos is an authentication system. It does not contain any authorisation functionality other than to provide a place for it. For example, Kerberos will issue you with a ticket for a server, even if you are not authorised to access that server. It is the server itself that decides whether a user is allowed to access its resources.

This can concern newcomers to Kerberos, but it is correct that this authorisation decision is not part of Kerberos. When you Telnet to a UNIX server, your user-id and password are first checked against a user database. Once authenticated, then you are taken to your home directory and log-on scripts run. All authorisations are controlled by the UNIX Operating System.

The same process applies within a Kerberos system. The Kerberos authentication takes the place of the password authentication. The kerberised Telnet daemon checks the local user database to ensure the user is valid for this system and then completes the normal log-on process.

Kerberos protocol vulnerabilities

No security system can protect against all types of attacks and Kerberos is no exception. The three main areas of attack upon a Kerberos system are:

- Network time attack
- Password guessing attack
- Insecure host systems

Network time attack

One potential threat to a Kerberos system is to replay a previously sent authentication request. To counter this threat a unique time based authenticator is included within the authentication request. The Application Server checks the time the authenticator was issued against its own clock, and if it is out by a defined period (5 minutes is usual) then the request is rejected. To prevent replay attacks during the 5-minute period the Kerberos protocol specifies that authenticators are cached and checked for a replay.

However, if an attacker can alter the network time service, then the replay of authentication requests becomes possible.

To overcome this threat a Kerberos infrastructure requires a secure network time service to prevent the system clock being altered by an attacker's network time service. This should be normal practice for most IT departments in any case, because a reliable network time service is needed for other reasons, for example, auditing.

Password guessing attacks

The protocol returns the TGT encrypted under the user's secret key, which is a cryptographic transformation of the user's password. An attacker can collect these replies and apply standard password guessing attacks to determine the user's password.

Insecure systems

It is normal practice to secure IT systems providing business services. Any weaknesses in the security of the underlying system allows attackers to obtain unauthorised access to business systems or direct access to the databases.

Kerberos also requires host systems to be secured to prevent an attacker from:

- Altering the system time and playing back an old authenticator to obtain access
- Copying the key tables that hold the Kerberos credentials for telnet and ftp daemons and client processes
- Copying a user's active tickets from credential cache.

Designing a Kerberos Infrastructure

Components of a Kerberos System

A Kerberos infrastructure consists of the following components:

- Master Kerberos Security Server
- One or more slave Security Servers
- A database of principals (users, clients, servers)
- An Authentication Server
- A Ticket Granting Server
- An Administration Server
- A Database Propagation Server
- Kerberos libraries
- Credential cache
- Kerberised client and server applications

These components are shown in Figure 2 and are described in the following Sections:

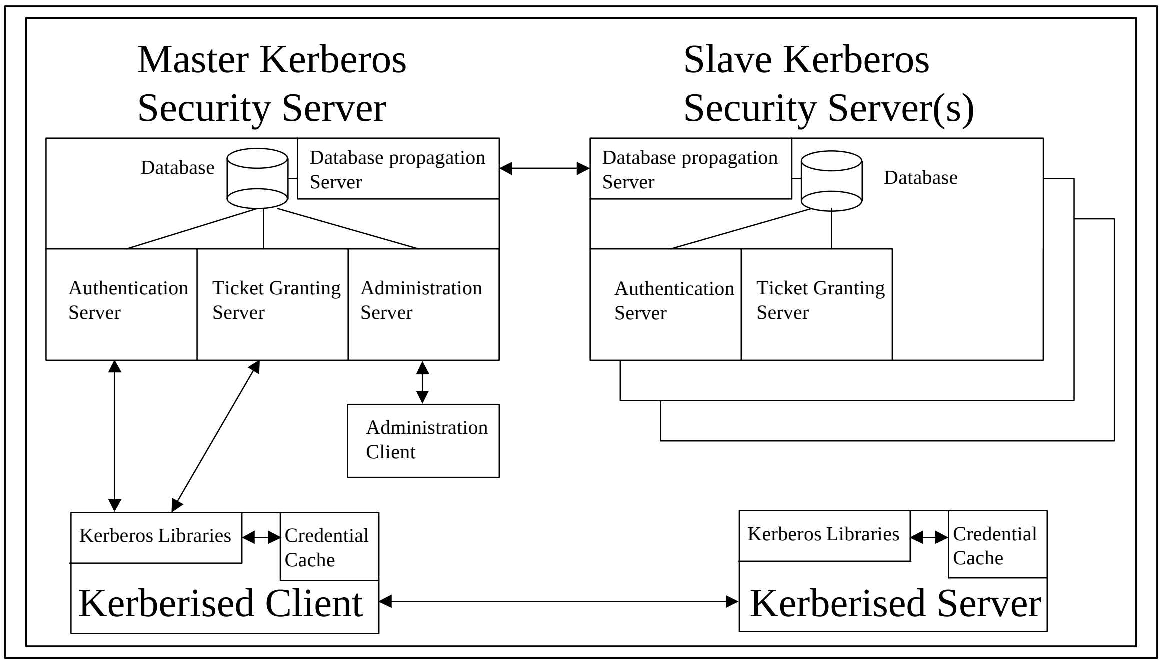


Figure 2 – A Kerberos Infrastructure

Kerberos Security Servers

At the heart of a Kerberos infrastructure are the Kerberos Security Servers, which are also known as Key Distributions Centres (KDC). In a Kerberos infrastructure there is a master server, which contains the master database of Principals. To provide resilience and scalability, there are usually one or more slave security servers that contain a copy of the database and have their own Authentication and Ticket Granting Servers.

The physical and logical security of these security servers is absolutely vital. To prevent unauthorised access to the Kerberos Security Servers, these need to have a hardened operating system and be securely managed to a similar standard as a firewall.

Authentication Server

The Authentication Server runs as a process on the KDC and its role is to authenticate a user and issue a ticket for the Ticket Granting Server. This ticket is known as a Ticket Granting Ticket.

Ticket Granting Server

The Ticket Granting Server is in reality a kerberised Application Server with a role to issue tickets for other Application Servers. Like the Authentication Server, it needs direct access to the database so is usually a process running on the Security Servers.

Administration Server and Client

A Kerberos infrastructure requires an administration system for managing Users, clients and servers in the Principal database. This system consists of an Administration Server that runs on the Security Server plus a separate client application to enable remote administration.

Database Propagation Server

A Kerberos infrastructure will have a master server and one or more slaves. The database propagation server runs on all security servers with the role of propagating updates from the master server to the slaves.

There are two main propagation architectures:

- Regular copies of the entire database are sent out at fixed intervals
- Incremental changes are frequently sent out.

For any enterprise deployment, choose an implementation that propagates incremental changes, because the addition of new users and password changes can be quickly propagated to the slave systems whilst minimising network traffic.

Kerberos Libraries and credential cache

Every Kerberos client and Application Server needs access to the underlying cryptography and Kerberos functionality that comprises the Kerberos protocol. This is usually a separate series of libraries provided by the Kerberos vendor. When a user logs-on, a local credential cache is created to store the user's tickets.

Kerberos Client and Server Applications

A Kerberos infrastructure is a white elephant unless there are applications that support the protocol. To this end Kerberos vendors provide a suite of client applications, which usually consist of the following:

- Single sign-on client for Windows 9x & NT
- Clients and daemons for Telnet, FTP, r commands, and graphics subsystem (i.e. Solaris's CDE).

Many third party software vendors already support Kerberos within their products. Examples include Oracle, Sybase and SAP/R3, which integrate tightly with Cybersafe's ActiveTrust product.

With the integration of Kerberos into Microsoft's Windows 2000, we can expect more applications within the Windows world to become kerberised. This will ultimately enable users to have single sign-on across the Windows and UNIX worlds for a wide variety of applications.

Design factors

Number of Realms

One of the key decisions to be made when designing a Kerberos Infrastructure is the number of realms. This decision will be influenced by the following considerations:

- **Administration domains.** Kerberos administrators can manage all principals within a realm. Therefore requirements for segregated administration domains may require separate Kerberos realms.
- **Location and autonomy of company offices.** Passwords can only be changed on the master database and are then propagated to the slave servers. If the Wide Area Network to the remote office is unreliable, or the remote office does not want to be dependent upon systems in other offices, then a separate realm may be required
- **Structure and number of Windows NT domains.** If your Kerberos system is to tightly integrate with Microsoft Windows domains, then there may be a requirement to have a one for one relationship between Kerberos realms and Windows domains.
- **Use of Kerberos for authenticating customers to IT systems.** A separate realm for customers may be required to segregate customers from internal applications.

Cross Realm Authentication

Kerberos supports cross realm authentication enabling the users in one realm to securely access resources in other realms. Kerberos supports two types of realm structure, one to one cross authentication and hierarchical cross authentication.

One to one cross authentication is manageable for small numbers of realms. However, a key management problem emerges as the number of realms increases because the number of relationships grows exponentially to the number of realms. For example, if you have 4 realms then each realm will need 3 cross realm keys totalling 12 cross realm keys to be managed. 10 realms would require 90 cross realm keys.

Availability

The availability of the Kerberos infrastructure is absolutely vital. If the infrastructure is unavailable then users cannot access any kerberised server. To maximise availability the following steps should be taken:

- The security servers (KDCs) are physically and logically secured to prevent theft and accidental or deliberate denial of service attacks
- Have a back-up master server and a number of slave servers
- Have a contingency plan for each application so that a non-kerberised service

can be established in the event of a major problem with the Kerberos infrastructure.

Slave servers

Good commercial implementations of Kerberos support slave servers. When designing a Kerberos system, it is good design to have clients authenticating to and requesting tickets from the slave servers rather than the master. The reasons for this approach are:

- The master can handle a larger user database because its role is limited to handling password changes, propagating these changes to slave servers, and user management
- The scalability of the service can be easily increased by adding additionally slave servers and load balancing between them
- Performance can be improved to remote offices with slow network links by having local slave servers.

Choosing a Kerberos system

The previous sections mention a number of features that are necessary in an enterprise Kerberos infrastructure. This section brings these key features together to help an evaluation of Kerberos systems.

1. Choose a commercial implementation of Kerberos, unless your organisation has the resources and desire to keep a dedicated team of Kerberos developers. MIT provide source code for Kerberos, but this is not an enterprise ready system.
2. Ensure the vendor has the resources to support you and a solid roadmap for the further development of the product.
3. The product is supported on a wide range of platforms otherwise you may have enclaves of systems that cannot integrate into your Kerberos infrastructure.
4. The capability to provide single sign-on over both Windows and UNIX worlds.
5. Support for slave servers with replication by incremental updates and a configurable replication time. If you are running a load balanced pool of slave servers then any changes in a user's password or status needs to get to all the slave servers at roughly the same time. The replication system must also propagate any configuration files that control the system.
6. A reliable and functional database product managing the Principal database.
7. A hot standby for the master server with an automated fail-over. If the primary server fails then password changes and user management cannot be done.
8. Support for pre-authentication for passwords and token cards. Without pre-authentication, the Kerberos system is limited to user-id & password authentication and cannot track failed authentications in support of account locking.
9. Support for PKINIT. This allows Kerberos to authenticate clients via smartcards and public key certificates. This extends single sign-on out to PKI authentication models.
10. The capability to easily change the master database's password and propagate the change to the slave security servers.
11. The capability to delegate the management of host principals to system administrators and servers. This allows the servers to periodically and automatically change their own key file in accordance with security policy.
12. Includes a suite of Kerberised applications and a list of third party vendor products that are certified to work with the Kerberos system. The applications typically needed by large organisations for both Windows and UNIX client and servers are a single sign-on client, Telnet, FTP, r commands and X.
13. A Java and C++ SDK certified for a wide variety of platforms. This allows an organisation to integrate their bespoke applications into their Kerberos single sign-on infrastructure.
14. Integration with third party databases such as Oracle and Sybase is highly desirable because it extends the single sign-on model out to these popular databases.
15. Integration with third party Enterprise Administration products such as BMC's Control SA is a desirable feature, because it integrates the Kerberos realm into a single point of administration architecture. This helps to reduce the number of administration tools used by security administrators.

Implementation considerations

This section contains some general considerations when implementing, configuring and supporting a Kerberos infrastructure.

Kerberos V4 Support

To provide backward compatibility with Kerberos V4 applications, a Kerberos V5 KDC may also support V4. It is best to disable Kerberos V4 compatibility mode in a Kerberos V5 system, because the system becomes vulnerable to security issues resolved by Kerberos V5.

Setting the ticket lifetime

The KDC needs to be configured with a default ticket lifetime. This needs to be set long enough so that users' TGTs do not expire during the normal working day, otherwise users may have to authenticate themselves again. An alternative solution to this problem is for the Kerberos client to automatically request a new TGT when the existing Ticket expires as in the Microsoft client.

Managing the Master database password

In some Kerberos implementations, the master database password needs to be the same for the master and all the slave security servers, and the engineers that set-up slave servers need to know this password. Therefore this password needs to be securely stored and regularly changed, especially when a support engineer leaves the company, or if the password becomes known outside of the support team.

Administering Kerberised Servers

A Systems Administrator will be called in to investigate the failure of an automated job, for example, a kerberised FTP. Therefore, Systems Administrators will need to be trained in supporting kerberised services. Part of this role will be to generate and extract the shared secret that the daemons need to load at start-up. The system administrator will also need to set-up a job to automatically change the shared secret in accordance with security policy.

Contingency

A well-designed Kerberos infrastructure will provide a very high level of availability in normal failure situations. However, should a fault be present in the vendor's software, then there is the possibility that the entire infrastructure could fail and prevent your users from working.

To guard against this catastrophe, then any Kerberos client that integrates with another log-in system, for example Windows NT, must have a fall through mode that allows the user to drop through to the operating system log-on. In this way only the kerberised applications will be unavailable.

Additionally, each application should have a contingency plan prepared for an outage of the Kerberos infrastructure. For example, for FTP based systems, the backup could be to log-on via a non-kerberised user-id and have an emergency script to switch on non-kerberised daemons. User-ids and passwords for the FTP job can then be set as normal using the operating system tools.

Supporting the Infrastructure

For IT project managers to feel comfortable about integrating Kerberos into their applications, the Kerberos system needs to be fully supported from both a development and an infrastructure viewpoint.

There are generally three categories of support roles, these are:

- **Security Server Support.** This support is required to ensure that the security servers themselves are maintained in an operational state. This category includes hardware and operating system support as well as support for the Kerberos Software
- **Kerberised application support.** When a kerberised application fails, the support staff need to be able to diagnose and fix any problems caused by Kerberos
- **Application implementation support.** This role assists application developers to integrate with the Kerberos Infrastructure. The role ranges from helping developers use an SDK, to providing sample scripts for host principals and automated clients.

Taking the Infrastructure into Production

Absolutely vital to the success of the project is to have the infrastructure fully supported before integrating applications into it. The next step is to have a couple of low risk pilot projects to identify any teething problems before stepping up the pace of the rollout.

Integrating Kerberos into an existing application

Integrating Kerberos into an existing application needs to be carefully managed. Most applications have their own user databases giving users many different user-ids. Integrating an application into a Kerberos single sign-on system requires a review of the user-ids to bring them into step with their Kerberos Principal-id. Users may also need to be added to the Principal database. IT may also be necessary to identify the users' workstations so that any single sign-on client can be rolled out.

Vendors of Kerberos systems

The following vendors supply Kerberos KDCs.

- Cybersafe Corporation
- Microsoft Corporation
- Sun Microsystems

Cybersafe Corporation

Cybersafe Corporation has been in the Kerberos market for a number of years and has a mature product offering for Windows and a wide range of UNIX variants including Solaris, AIX and HP-UX. The current version of the product is called ActiveTrust (previously called TrustBroker) and can provide single sign-on across Windows and all these UNIX environments. The KDC supports pre-authentication extending authentication capability to include RSA's Secur-Id tokens and smartcards via PKINIT. Additionally, Oracle, Sybase and SAP/R3 integrate with ActiveTrust extending single sign-on out to these popular products. The system also comes with kerberised versions of telnet, ftp and the r commands along with an SDK for integrating bespoke applications.

For further information refer to www.cybersafe.com

Microsoft Corporation

Microsoft introduced Kerberos into Windows 2000 for network authentication and note a number of benefits over the existing Windows NT LAN Manager (NTLM) protocol[3]. The KDC is a process on the domain controller and all user information is stored in the Microsoft active directory. The Microsoft implementation is interoperable with other Kerberos systems[4]. For further information refer to www.microsoft.com.

Sun Microsystems

In Solaris 8, Sun introduced their Secure Enterprise Authentication Mechanism(Seam) which is a Kerberos v5 KDC with hot standby capability and built in support for ftp, NFS, telnet and the r commands. It also comes with an SDK for integrating bespoke applications. For further information refer to www.sun.com.

References

[1] Ken Hornstein – Kerberos FAQ - http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html

[2] J. G. Steiner, B. Clifford Neuman, and J.I. Schiller - "Kerberos: An Authentication Service for Open Network Systems". - <ftp://athena-dist.mit.edu/pub/kerberos/doc/usenix.PS>

[3] Microsoft Corporation - "Windows 2000 Kerberos Authentication White Paper". - http://www.microsoft.com/windows/server/Technical/security/kerberos.asp

[4] Microsoft Corporation- "Windows 2000 Kerberos Interoperability White Paper" - http://www.microsoft.com/DirectAccess/Products/CRK/Server/SERSECWPS.asp

[5] J. Kohl, C. Neuman - “The Kerberos Network Authentication Service (V5),” Internet Request for Comments 1510 (September 1993).

[6] S. Bellovin and M. Merritt - “limitations of the kerberos authentication system” - Usenix Winter ‘91

[7] Bill Bryant - "Designing an Authentication System: A Dialogue in Four Scenes." <http://web.mit.edu/kerberos/www/dialogue.html>

[8] R.M. Needham and M.D. Schroeder - “Using Encryption for Authentication in Large Networks of Computers”, Communications of the ACM 21(12), pp. 993-999 (December, 1978)

[9] C. Neuman, J. Wray, B. Tung, J. Trostle, M. Hur, A. Medvinsky, S. Medvinsky, “Public Key Cryptography for Initial Authentication in Kerberos,” Internet Draft (November 1998).