Securing Storage Appliances via UNIX based Kerberos Authentication

Latesh Kumar K.J
Dept.of.Computer Science
S.I.T, B.H.Road
Tumkur, Karnataka

ABSTRACT
Securing storage systems to use UNIX-based Kerberos version 5 servers for NFS storage authentication using both NFS version 3 and 4. NFS version 4 is the NFS Implementation and mandates Kerberos authentication as part of the NFS client and server specification Integrate their storage systems with Kerberos version 5 to achieve strong NFS storage authentication.

General Terms
Storage Security, Kerberos, Algorithms.

Keywords
KDC, SEAM, MIT, NFS, CIFS.

1. INTRODUCTION
Kerberos is a network authentication protocol used in client-server applications. There are two versions of Kerberos currently in use, version 4 and 5. Kerberos versions 1 through 3 were internal development versions and never released. Kerberos version 4 has a number of known weaknesses and should no long be used. There are several popular Kerberos version5 distributions today, including MIT Kerberos, Heimdal Kerberos, Sun® Enterprise Authentication Mechanism (SEAM), and the Kerberos implementation in Microsoft® Active Directory. MIT and Heimdal Kerberos are freely distributed in public domains. The major difference between MIT and Heimdal Kerberos is that MIT Kerberos is subject to U.S. government export regulations, while Heimdal Kerberos is not. The Heimdal Kerberos distribution is available as a port (security/Heimdal), and a minimal installation of it is included in the base FreeBSD install.

Kerberos 5 (RFC 1510) added security enhancements that were not available in Kerberos 4. MIT Kerberos supports both 4 and 5, and the newer Heimdal Kerberos implementation supports only version 5. However, there is often confusion about which NFS versions support Kerberos authentication. One common Misconception is that NFS v2 and v3 do not support Kerberos authentication. NFS v2 supports Kerberos 4, and NFS v4 supports Kerberos 5 which is the best authentication mechanism for NFSv4. Storage systems fully support Kerberos 5 and Microsoft Active Directory-based Kerberos and these can be used with NFS and CIFS in storage environment. This document provides guidance to users to implement Kerberos in their existing NFS storage fabric to accomplish strong authentication. After reading this document you can install, setup and configure Kerberos version 5 in your storage environment.

2. HOW KERBEROS WORKS
Kerberos is an authentication protocol which uses a shared secret and a trusted third party arbitrator in order to validate the identity of clients. In Kerberos, clients may be users, servers, or pieces of software. The trusted third party arbitrator is a server known as a Key Distribution Center (KDC) which runs the Kerberos daemons. The shared secret is the users password transformed into a cryptographic key. In the case of servers or software systems, a random key is generated. In Kerberos, users are known as principals. The KDC has a database of principals and their secret keys which is uses to perform authentication. In Kerberos knowledge of the secret key is considered sufficient for proof of identity. Since knowledge of a secret key translates into proof of identity in Kerberos, the Kerberos server can be trusted to authenticate any client to any other client. Authentication is Kerberos is done with out sending any clear text passwords across the wire.

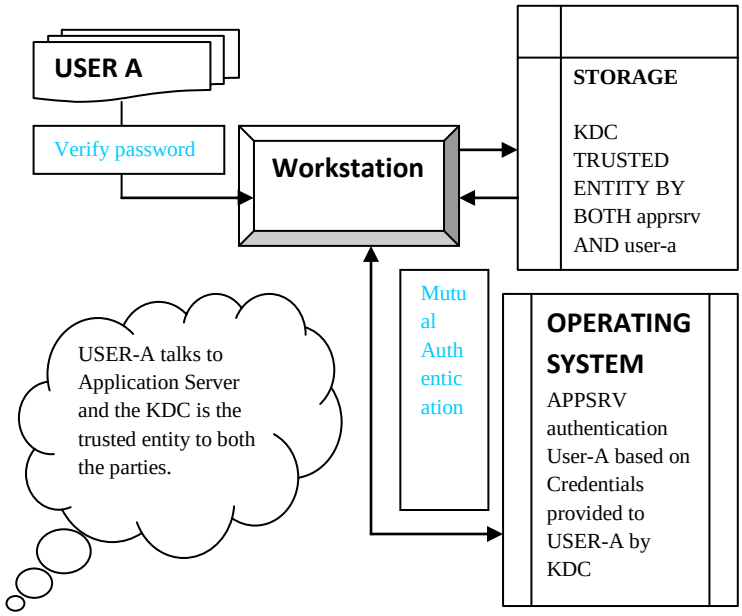


Fig 1: How Kerberos Works

2.1 Messages

Signal	Comments
KRB_AS_REQ or AS_REQ	Used to ask for initial TGT
KRB_AS_REP or AS_REP	Used to return TGT
KRB_TGS_REQ or TGS_REQ	Used to ask for Ticket for service/application
KRB_TGS_REP or	Used to return Ticket for service/application

TGS_REP	
KRB_AS_ERR	There are many Used by KDC to report why it cannot grant a ticket in response to AS_REQ or TGS_REQ
KRB_AP_ERR	Again, there are many Used by the Application Server to report why authentication failed

2.2 Key Commands

Command	Description
Kadmin.local	Add/Delete/Modify principals Generate keys (keytab files)
kdb5_util	Manage KDC database
Klist	List contents of keytab files
Kinit	To get the initial TGT
Ktadd	Add a Kerberos key to a keytab file
Ktutil	Another command that you can use to administer keytab files is the ktutil command. This interactive command enables you to manage a local host's keytab file without having Kerberos administration Privileges, because ktutil doesn't interact with the Kerberos database as Kadmin does. So, after a principal is added to a keytab file, you can use ktutil to view the keylist in a keytab file or to Temporarily disable authentication for a service. With ktutil you can Temporarily Disable Authentication for a Service on a Host

3. CONFIGURING KERBEROS ON SUN SOLARIS

3.1 Configuring Kerberos on Solaris® Solaris

The Kerberos setup constitutes 3 major steps

1. KDC (Key Distribution center)

2. Kerberos Client (Client to access the Storage)

3. Kerberos Server (Storage)

3.2 Setting up KDC

The realm used is "BIGFOOT.REALM1.COM". The steps to configure the KDC are as follows:

INITIALIZE THE KERBEROS DATABASE

Use the kdb_util program to initialize the Kerberos database for the realm

'SNOWMAN.LAB.ENG.SHADOW.SHADOW.IN'. The syntax of the command is:

kdb5_util create -r <realm-name>

sun123# kdb5_util create -r BIGFOOT.REALM1.COM –s

3.3 Configuring KDC

The KDC is configured through the config file /etc/krb5/kdc.conf. A sample configuration is shown below:

[realms]

BIGFOOT.REALM1.COM= {

profile = /etc/krb5/krb5.conf

database_name = /var/krb5/principal.BIGFOOT.REALM1.COM

admin_keytab = /var/krb5/kadm5.keytab.BIGFOOT.REALM1.COM

acl_file = /etc/krb5/kadm5.acl.BIGFOOT.REALM1.COM

Kadmind_port = 749

max_life = 8h 0m 0s

max_renewable_life = 7d 0h 0m 0s

default_principal_flags = +preauth

supported_enctypes = des-cbc-crc:normal des-cbc-md5:normal

3.4 Generate Service Key

The service key will be shared between the server and the KDC. This key is used to encrypt the ticket that the KDC grants to the client. Launch the application 'Kadmin.local' on the KDC.

Generate keys for the service "nfs" as follows

Kadmin.local: add_principal -e "des-cbc-crc:normal des-cbc-md5:normal" -randkey

nfs/storage123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

3.5 Generate Keytab file for SERVER

The service key generated in the previous step gets added to the Kerberos database. To pull this out into a keytab file, the following command is used:

Kadmin.local: ktadd -k /tmp/storage123.keytab -e "des-cbc-crc:normal des-cbcmd5:

normal" nfs/storage123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

3.6 Generate Keys for Client

Generate keys for the clients. Note that for Linux, the service needs to be named as "nfs/<client

FQDN>@<realm>" and for Solaris this would be named as "root/<client FQDN>@<realm>"

For Linux client with FQDN 'lnx123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

Kadmin.local: add_principal -e "des-cbc-crc:normal des-cbc-md5:normal" –randkey nfs/lnx123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

For Solaris client with FQDN 'sun123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

Kadmin.local: add_principal -e "des-cbc-crc:normal des-cbc-md5:normal" -randkey

root/sun123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

3.7 Generate Keytab Files

Kadmin.local: ktadd -k /tmp/lnx123.keytab -e "des-cbc-crc:normal des-cbcmd5:

normal" nfs/lnx123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

Kadmin.local: ktadd -k /tmp/sun123.keytab -e "des-cbc-crc:normal des-cbcmd5:

normal" root/sun123.lab.eng.shadow.shadow.in@ BIGFOOT.REALM1.COM

4. PURPOSE AND BENEFITS

The Kerberos configuration on the Client and the Application Server is done through the krb5.conf file. This file will be placed in the /etc/krb5 directory in case of Solaris and in the /etc directory in case of Linux. On the Storage, this file is placed in the /etc directory. A sample configuration is shown:

[libdefaults]

default_realm = BIGFOOT.REALM1.COM

default_tgs_enctypes = des-cbc-md5 des-cbc-crc

default_tkt_enctypes = des-cbc-md5 des-cbc-crc

[realms]

BIGFOOT.REALM1.COM= {

kdc = sun217-21.BIGFOOT.REALM1.COM:88

admin_server = sun217-21.BIGFOOT.REALM1.COM :749

default_domain = BIGFOOT.REALM1.COM }

11 Configuration and Best Practises – Kerberised NFS in storage box

[logging]

kdc = FILE:/var/log/krb5kdc.log

admin_server = FILE:/var/log/Kadmin.log

default = FILE:/var/log/krb5lib.log

[domain_realm]

.lab.eng.shadow.shadow.in = BIGFOOT.REALM1.COM

lab.eng.shadow.shadow.in = BIGFOOT.REALM1.COM

[appdefaults]

pam = {

debug = false

ticket_lifetime = 36000

renew_lifetime = 36000

forwardable = true

krb4_convert = false

5. CLIENT CONFIGURATION

Copy the keytab files, Copy the keytab file generated as generated above to the Client under the /etc directory. Rename the file as krb5.keytab. Note: Kerberos server doesn't does not encrypt while copying keytab files across servers and clients manually. It is highly recommended to use secure copy tools like scp, ssh to copy the keytab files either to Shadow Storage or Clients.

Enable NFS security Create/edit the file /etc/sysconfig/nfs and add the following line SECURE_NFS=yes

Restart rpcgssd

Restart the service using the command:

/etc/init.d/rpcgssd restart

5.1 Configure Solaris as Client

Copy the keytab files

Copy the keytab file generated in #5 from the KDC Configuration section on to the Client under the

/etc/krb5 directory. Rename the file as krb5.keytab

Enable NFS security

Un-comment all the krb lines in /etc/nfssec.conf

Restart rpcdgssd

svcs | grep gss

svc:/network/rpc/gss:default

svcadm -v disable svc:/network/rpc/gss:default

svcadm -v enable svc:/network/rpc/gss:default

6. KERBEROS REPLICATION

Kerberos is designed to allow Master/Slave replication of a cluster. A master KDC serves as the primaryserver and at least one slave KDC which is a backup. The master and slave servers may be thought of as Primary and Secondary servers respectively.

Kerberos client applications are designed to attempt authentication against secondary servers if the primary is down. The administrative features of Kerberos do not provide for automatic failover. In the event that primary server fails, Kadmind will be unavailable. Therefore, administrative functions will be unavailable until the primary server is restored. Specifically, principal management, key creation, and key changes, cannot be done during a primary server failure.

IMPLEMENTATION

Server replication is handled by the kprop command and kprop must be run on the primary master KDC. It should be run in a scheduled cron job to keep the principal database in sync across all servers.

The first step in setting up replication is to set up ACLs for kpropd. The kpropd acl filename is by default located at /var/Kerberos/krb5kdc/kpropd.acl. In our example, it would have the following contents:

host/kerberos1.test.com@test.com

host/kerberos2.gnud.ie@ sit.com

The kpropd.acl file should only exist on the slave Kerberos server. In Fedora derived GNU/Linux, Kadmin will not run on a Kerberos server on which /var/Kerberos/krb5kdc/kpropd.acl exists.

Next we'll have to create host keys for your master and slave Kerberos servers:

{Kerberos1}bash# Kadmin.local

{Kerberos1}Kadmin.local: addprinc -randkey host/kerberos1.test.com

{Kerberos1}Kadmin.local: addprinc -randkey host/kerberos2. sit.com

The next step is to extract these keys to the keytab file. The keytab file is a keyring which contains the cryptographic keys needed to authenticate with the KDC. Extraction of keys to the keytab is done with the

ktadd sub command:

{Kerberos1}Kadmin.local: ktadd host/kerberos1.test.com

{Kerberos1}Kadmin.local: ktadd host/kerberos2. sit.com

Then finally, copy the keytab over to the slave server so that it has the keys it needs available to authenticate.

{Kerberos2}bash# scp root@kerberos1.gnud.ie:/etc/krb5.keytab /etc

Here is a crontab entry from the master Kerberos server used to synchronize principal databases every fifteen minutes:

15 * * * * /usr/local/bin/krb5prop.sh

Here are the contents of the krb5prop.sh script:

#!/bin/sh

/usr/Kerberos/sbin/kdb5_util dump /var/Kerberos/krb5kdc/slave_datatrans

/usr/Kerberos/sbin/kprop -f /var/Kerberos/krb5kdc/slave_datatrans kerberos2. sit.com > /dev/null

Initially running this command by hand, you should see something similar to the following:

{Kerberos1} bash# /usr/Kerberos/sbin/kdb5_util dump /var/Kerberos/krb5kdc/slave_datatrans

{Kerberos1}bash# /usr/Kerberos/sbin/kprop -d -f /var/Kerberos/krb5kdc/slave_datatrans kerberos2.sit.com

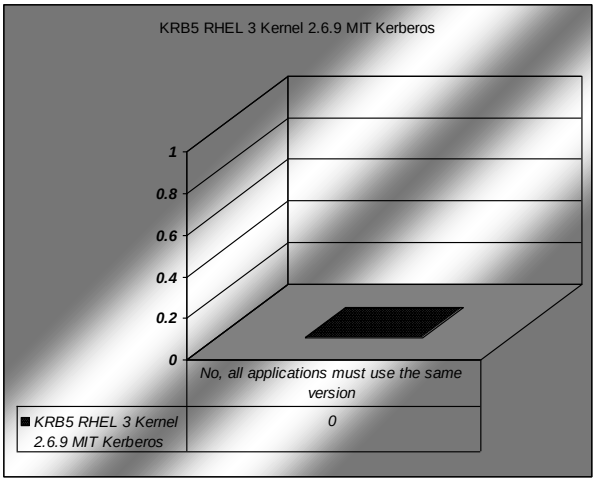
3234 bytes sent.

Database propagation to kerberos2. sit.com: SUCCEEDED

{Kerberos1}bash# The slave server will now synchronize its principal database with the master server.

7. STATISTICAL ANAYLSIS

The below chart illustrates the features supportability for the web application via storage connected..



8. CONCLUSION

Although Kerberos removes a common and severe security threat, it may be difficult to implement for a variety of reasons:

• Migrating user passwords from a standard UNIX password database, such as /etc/passwd or

/etc/shadow, to a Kerberos password database can be tedious, as there is no automated mechanism to perform this task. Refer to the online Kerberos FAQ:

• Kerberos has only partial compatibility with the Pluggable Authentication Modules (PAM) system used by most Red Hat Enterprise Linux servers. (Only with Linux)

• Kerberos assumes that each user is trusted but is using an untrusted host on an untrusted network. Its primary goal is to prevent unencrypted passwords from being transmitted across

that network. However, if anyone other than the proper user has access to the one host that issues tickets used for authentication — called the key distribution center (KDC) — the entire Kerberos authentication system is at risk.

• For an application to use Kerberos, its source must be modified to make the appropriate calls into The Kerberos libraries. Applications modified in this way are considered to be Kerberos-aware, or kerberized. For some applications, this can be quite problematic due to the size of the application or its design. For other incompatible applications, changes must be made to the way in which the server and client communicate. Again, this may require extensive programming. Closed-source applications that do not have Kerberos support by default are often the most problematic.

9. REFERENCES

[1] B. Clifford Neuman and Theodore Ts'o, Kerberos: An Authentication Service for Computer Networks, IEEE Communications 32 (1994), no. 9, 33—38

[2] S. M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. Computer

Communication Review, 20(5):119-132, October 1990. postscript

[3] Prof R.P. Arora, Garima Verma, "Implementation of Authentication and Transaction Security based on Kerberos", IJITCE, Feb 2011 7..

[4] "How Kerberos Authentication Works",Learn Networking on line magazine, Jan'2008

[5] Ravi Ganesan, "Yaksha' : Augmenting Kerberos with Public Key cryptography"