KERBERIZED NFS IN A NETAPP STORAGE SYSTEM USING A UNIX-BASED KERBEROS AUTHENTICATION SERVER

H.T. Sun and Christopher Smith, Network Appliance, Inc.

June 2006, TR-3481

Abstract

This guide explains the basic concepts and terminologies used in the Kerberos authentication system and provides guidance on configuring NetApp storage systems to use UNIX-based Kerberos servers for NFS authentication. Integration with NFS version 4, the latest NFS implementation that mandates Kerberos authentication, is also discussed and demonstrated with examples. The goal is to help customers successfully integrate their NetApp storage systems with Kerberos to achieve strong NFS authentication.

Table of Contents

1. INTRODUCTION

Kerberos is a complex network authentication system that was introduced by MIT in 1989. In Greek mythology, Kerberos was a ferocious three-headed dog that guarded the entrance to the underworld (Hades). Kerberos ensured that only the dead, not living souls, entered Hades. In our world, Kerberos is a system that employs strong data encryption algorithms (for example, AES, DES, and 3DES) along with message digest algorithms (for example, CRC-32, MD5, and SHA1) to ensure data confidentiality and integrity during the authentication process. Confidentiality ensures that only parties with a shared key can encrypt or decrypt the data. Integrity ensures that the data is not tampered with while being transmitted across the network.

Kerberos never sends password across the network, in either cleartext or encrypted form. It relies on encrypted tickets and session keys to authenticate users before they can use network resources. The Kerberos system utilizes key distribution centers (KDCs) that contain a centralized database of usernames and passwords. The network services that support Kerberos authentication are said to be *Kerberized.* LDAP, CIFS, and NFS are among the many network services that have been Kerberized.

There are several popular Kerberos distributions today, including MIT Kerberos, Heimdal Kerberos, Sun® Enterprise Authentication Mechanism (SEAM), and the Kerberos implementation contained 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 most widely deployed Kerberos versions are 4 and 5. Generally speaking, 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.

As mentioned earlier, NFS has been Kerberized. However, there is often confusion about which NFS versions support Kerberos authentication. One common misconception is that NFS v2 and NFS v3 do not support Kerberos authentication. The truth is that NFS v2 supports Kerberos 4, and NFS v3 and NFS v4 both support Kerberos 5. NetApp storage systems fully support Kerberos 5 and Microsoft Active Directory-based Kerberos.

This paper is intended for security-aware technical audiences who plan to implement Kerberos in their existing NFS environment to achieve their security requirements. Readers are assumed to have hands-on knowledge in working with a Kerberos server, and are encouraged to read TR 3387: Security in NFS Storage Networks, for an overview of Kerberos authentication before continuing. The current paper discusses the integration of NetApp storage systems with UNIX-based Kerberos systems over NFS. The discussion of Active Directory-based Kerberos implementation is outside the scope of this document and should refer to TR 3457: "Unified Windows and UNIX Authentication Using Microsoft Active Directory Kerberos" for more information.

Support Matrix

NetApp has tested the compatibility of the Data ONTAP® Kerberos implementation with server/client platforms running different flavors of Kerberos. Here is a list of the operating systems that have been tested:

- Data ONTAP 6.4 and later supports Kerberos 5.
- KDCs: Data ONTAP has been tested with MIT KDCs on Linux®, Solaris™ (SEAM) KDCs on Solaris, and Active Directory KDCs on Windows® 2000. The KDC examples used in this document are based on MIT Kerberos 5.
- Supported UNIX clients with Kerberized NFS:
  - AIX 5.3
  - Solaris 9 and 10
  - Linux kernel 2.6.x. Supported distributions are:
    - Red Hat Enterprise Linux 4
    - Fedora Core Linux

2. BENEFITS

A number of significant benefits can be achieved by deploying Kerberos as the authentication system in the NAS environment.

- **Security**

Most authentication systems rely on exchanging either cleartext or encrypted passwords between the client and the server over the network. Kerberos, on the other hand, does not send any password whatsoever across the network. Instead, it uses session tickets that have an expiration date. When the ticket expires, users are required to renew the ticket before continuing to use the network resources. Because passwords are never sent over the network, the possibility of password attack is greatly reduced.

- **Single sign-on (SSO)**

  Single-sign-on means that users need to log in only once to access all the available network resources supported by Kerberos. When end users successfully log on to a Kerberized system, their credentials are automatically cached in the local hard disk and can be passed over to all the network resources for authentication purposes. As a result, the end user does not need to repeatedly type the password in order to use the network resources. Single sign-on not only minimizes the inconvenience of the conventional authentication process, it also ensures that authentication is still in place to protect the network resources.

- **Centralized authentication server**

  Kerberos relies on the key distribution center (KDC) to centrally manage and distribute tickets. The KDC can generate tickets for server principals as well as user principals based on preferred encryption algorithms for data confidentiality and checksum mechanisms for data integrity. From the system administrator's perspective, KDC offers the benefit of centralized management. In order to avoid the "single point of failure" scenario that may arise in centralized management, it is possible to have backup or slave KDCs for failover purposes. The slaves all synchronize their databases from the master KDC. Additionally, most Kerberos implementations allow remote manipulation of the Kerberos database.

- **Mutual authentication**

  In the Kerberos system, the user principal must be authenticated by the server principal, and also the server principal must prove to the user principal that it is indeed the server that the user intends to communicate with. Mutual authentication protects sensitive information from leaking away in the communications between the user and the server.

3. HOW KERBEROS AUTHENTICATION WORKS

3.1 KERBEROS TERMINOLOGY

Kerberos is a complex authentication system that relies on its various components functioning properly in order to accurately authenticate users or servers. For definitions of common Kerberos terms, refer to Appendix B, "Kerberos Glossary."

3.2 AUTHENTICATION PROCEDURES

This section describes conceptually how a NetApp storage system is integrated with the Kerberos system to authenticate Kerberos principals. The processes that happen behind the scenes are much lengthier and more complicated.

Basically, there are four steps to the authentication process. These steps include the exchange of encrypted tickets between the Kerberos server and client, followed by the regular NFS communication encrypted by session keys. For an overview of these steps, see Figure 1 below.

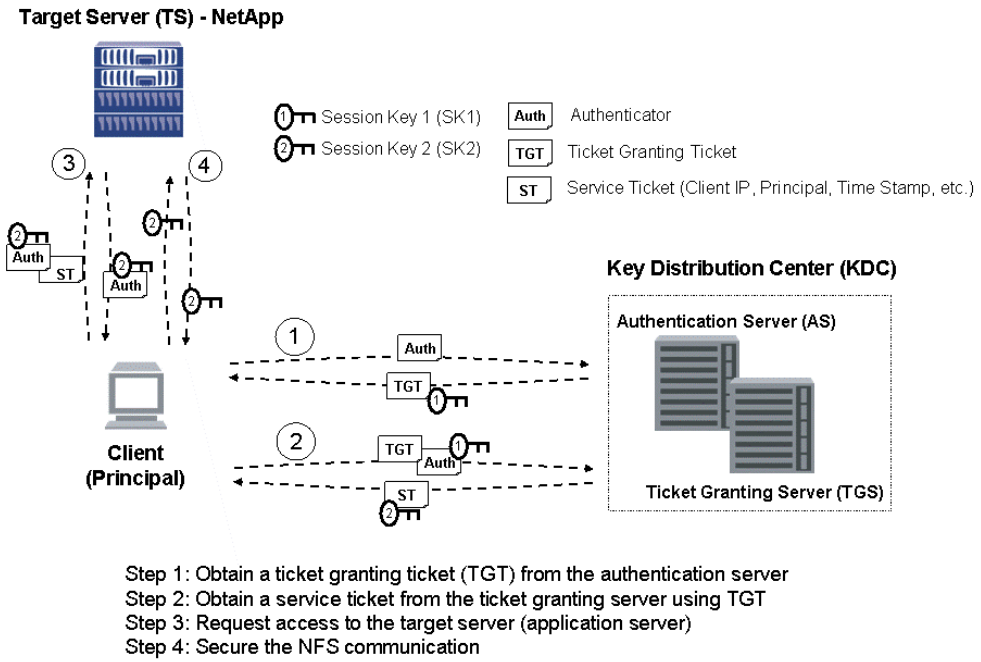

Figure 1 Integrating the NetApp storage system with the Kerberos authentication system

Step 1

When a Kerberos client (or principal) first authenticates to Kerberos, it contacts the authentication server (AS) to obtain a ticket granting ticket (TGT). The client sends an authenticator along with the TGT request to the AS. The AS looks up the client in its database. If the client exists, the AS issues a session key (SK1) for use in step 2 between the client and the ticket granting service (TGS). The AS encrypts SK1 with the client's secret key. It also encrypts the TGT, using the TGS's secret key.

Step 2

The client decrypts the cipher (encrypted message) sent by the AS in step 1 to retrieve SK1. The client then creates an authenticator, including the client's IP address (or username) and a timestamp, all encrypted using SK1, and sends it along with the TGT to the TGS.

Upon receipt of the TGT request, the TGS prepares a reply message containing a new session key (SK2) for use in step 3 between the client and the application server (NetApp storage system). Note that SK2 is encrypted with SK1 to ensure that only a validated client can retrieve SK2. Another copy SK2 is embedded inside a service ticket, which is encrypted with the desired service's (NFS) secret key. The service ticket is then included in the reply message, along with the encrypted SK2 sent back to the client. The client decrypts the message and retrieves SK2. The client then appends the service ticket and the new session key (SK2) to its credential cache for future retrieval.

Step 3

When the client wants to talk to a Kerberized service on a target server (application server), it retrieves the service ticket (encrypted with the Target Server's secret key) from the credential cache. The client also creates an authenticator encrypted with SK2 and sends it along with the service ticket to the Target Server (TS).

Upon receipt of the authenticator, the TS verifies that the client must know the SK2. In addition to this, the timestamp in the service key would thwart a "middleman attack," in which an eavesdropper secretly records the service ticket and authenticator and replays them later. The TS decrypts the service ticket and verifies whether every piece of information looks correct. If everything matches, the TS lets the real protocol communication proceed.

Step 4

In the final step, the client is authenticated successfully and allowed to use the Kerberized service on the target server. The client and the TS share the new session key (SK2) and are able to use it to secure the communication channel (message confidentiality) by encrypting the exchanged messages. Note that message confidentiality can be optional, depending on a company's security policy. For most Kerberos installations, the authentication mechanism described in the previous steps is adequate.

If mutual authentication is required, the TS returns a message encrypted by SK2, which contains a timestamp plus 1. This proves to the client that the TS actually knows the secret key to decrypt the service ticket and retrieve the timestamp.

4. INTEGRATING NETAPP STORAGE SYSTEM WITH KDC

This section discusses the procedures required to integrate a NetApp storage system with an MIT Kerberos 5 implementation for NFS protocol. The examples use Fedora Core 4 Linux (Linux kernel 2.6) as the OS for the KDC and Fedora Core 4 Linux and Solaris 10 (with SEAM) as the NFS clients.

4.1 KDC CONFIGURATION

No special options are required for the NFS integration. To the KDC, NFS is just another network service that is Kerberized and can communicate in Kerberos's terms. A typical Kerberos configuration file **krb5.conf** would look like the following:

```
[libdefaults]
 default_realm = NETAPP.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false

[realms]
 NETAPP.COM = {
  kdc = kdc.netapp.com:88
  admin_server = kdc.netapp.com
  default_domain = netapp.com
 }

[domain_realm]
 .netapp.com = NETAPP.COM
 netapp.com = NETAPP.COM
 .engineering.netapp.com = NETAPP.COM

[logging]
 default = FILE:/var/log/kdc.log
 kdc_rotate = {
    period = 1d
    version = 10
 }

[appdefaults]
 kinit = {
    renewable = true
    forwardable = true
 }
```

4.2 GENERATING USER, CLIENT, AND SERVER PRINCIPALS

In order to use Kerberos authentication for NFS, you need to generate principals for user, client, and server.

Creating NFS Principals

First, create a principal for the NetApp storage system named **fas6070.netapp.com** in Kerberos realm **NETAPP.COM**:

```
kadmin: addprinc -randkey -e des-cbc-crc:normal nfs/fas6070.netapp.com@NETAPP.COM
Principal "nfs/fas6070.netapp.com@NETAPP.COM" created.
```

This example generates a random key as the service key using the **–randkey** option. The **–e** option specifies the encryption type and salt used (separated by a colon) in the key generation. In the example, the encryption type is based on the Data Encryption Standard (DES) algorithm in the Cipher Block Chaining (CBC) mode; the checksum algorithm is based on Cyclic Redundancy Check (CRC). The NetApp storage system also supports the MD5 checksum algorithm. Since the key is derived from the password, the use of salt ensures that people who happen to pick the same password do not end up having the same key. The portion after the colon is the salt type; the example uses the Kerberos default salt type **normal**.

Second, use the same encryption/checksum algorithms and salt type (**des-cbc-crc:normal**) for the client principal **nfs/client.netapp.com**:

```
kadmin: addprinc -randkey -e des-cbc-crc:normal nfs/client.netapp.com@NETAPP.COM
Principal "nfs/client.netapp.com@NETAPP.COM" created.
```

Note that the Linux NFS client does not yet support the MD5 checksum algorithm, so you are restricted to the use of the CRC checksum only.

Finally, for the user principal **james**, you assign a password instead of using the random key:

```
kadmin: addprinc james@NETAPP.COM
WARNING: no policy specified for james@NETAPP.COM; defaulting to no policy
Enter password for principal "james@NETAPP.COM": xxxxxxx
Re-enter password for principal "james@NETAPP.COM": xxxxxxx
Principal "james@NETAPP.COM" created.
```

You can verify the creation of these three principals with the ***listprincs*** command:

```
kadmin: listprincs
kadmin/admin@NETAPP.COM
kadmin/changepw@NETAPP.COM
kadmin/history@NETAPP.COM
kadmin/kdc.netapp.com@NETAPP.COM
james@NETAPP.COM
nfs/fas6070.netapp.com@NETAPP.COM
nfs/client.netapp.com@NETAPP.COM
```

4.3 EXPORTING KEYS TO KEYTAB FILES

On the KDC, you export the client and server principal keys to two keytab files, one for the NetApp storage host and the other for the Linux client host. The keytab file is an encrypted, local, on-disk copy of the host's key:

```
kadmin: ktadd -k fas6070.keytab -e des-cbc-crc:normal nfs/fas6070.netapp.com
kadmin: ktadd -k client.keytab -e des-cbc-crc:normal nfs/client.netapp.com
```

When the keytabs are in place, you need to securely transport them to the desired destinations by using secure share (ssh) or some other means of secure file replication services. The NetApp NFS keytab file **fas6070.keytab** needs to be transported to the root volume (usually **/vol/vol0**) on the NetApp storage system and renamed to **/etc/krb5.keytab**. The keytab file for the Linux client needs to be moved to the client and renamed to **/etc/krb5.keytab**. Note that the keytab file is a potential point of entry for an attack, and once compromised, would allow unrestricted access to its host. Therefore, the keytab file should be readable only by

root or the account designated to run the desired Kerberized service. The keytab should exist only on the machine's local disk. You will need to delete the keytab files from their original locations on the KDC after the transportation and set appropriate file access permissions (**chmod 600**) on the destination keytab files.

4.4 SETTING UP KERBEROS ON A NETAPP STORAGE SYSTEM

Copying Kerberos Configuration File (krb5.conf)

First, copy **/etc/krb5.conf** on the KDC to the **/etc** directory in the root volume (usually **/vol/vol0/etc)** of the NetApp storage system.

Activating Kerberos Authentication

Second, enable Kerberos support in Data ONTAP. NetApp storage systems support both UNIX® KDC and Microsoft Active Directory-based KDC. For a UNIX KDC, you need to run the ***nfs setup*** command on the NetApp NFS server to activate the Kerberized NFS service:

```
Fas6070> nfs setup
Enable Kerberos for NFS? yes
The filer supports these types of Kerberos Key Distribution Centers (KDCs):

        1 - UNIX KDC
        2 - Microsoft Active Directory KDC

Enter the type of your KDC (1-2):  1
Enter the Kerberos realm name: NETAPP.COM
Enter the host instance of the NFS server principal name [default:
fas6070.netapp.com]: fas6070.netapp.com
NFS setup complete.
```

Note: Do not type the principal name **nfs/fas6070.netapp.com** in the previous step; enter only the fully qualified domain name of the NetApp storage system.

To verify that the setup was successful, issue the ***options nfs.kerberos*** command:

```
fas6070> options nfs.kerberos
nfs.kerberos.enable                 on
nfs.kerberos.file_keytab.enable     on
nfs.kerberos.principal              fas6070.netapp.com
nfs.kerberos.realm                  NETAPP.COM
```

Exporting NFS Qtree

Edit the **/vol/vol0/etc/exports** file to include the Kerberos security option in the export table; for example:

```
/vol/volx       -sec=sys:krb5:krb5i:krb5p,rw
/vol/volx/krb5  -sec=sys:krb5:krb5i:krb5p,rw
```

This export table offers four security options on qtree **/vol/volx/krb5** and its parent volume **/vol/volx**. It is always a good practice to ensure that the parent qtree or volume has the same security options as the child.

Data ONTAP allows you to specify multiple security options. In the previous example, four security options are allowed; it is up to the NFS client to choose one or more to use. These options (**sys**, **krb5**, **krb5i**, and **krb5p**, from least to most secure) are described as follows:

- UNIX (AUTH_SYS) authentication (**sys**): Does not use strong cryptography and is the least secure of the security services. This is the default security service used by Data ONTAP.
- Kerberos v5: Provides three methods of security:

- Authentication (**krb5**): Uses strong cryptography to prove a user's identity to the NetApp storage system and to prove the NetApp storage system's identity to a user.
- Integrity (**krb5i**): Provides a cryptographic checksum of the data portion of each request and the response message to each request. This defends against a "man in the middle" attack tampering with NFS traffic.
- Privacy (**krb5p**). Encrypts the contents of packets bi-directionally, including procedure arguments and user data, using a shared session key established by the client from the NetApp storage system.

You can now use the ***exportfs –av*** command to export the qtree.

4.5 CONFIGURING NFS CLIENT TO USE KERBEROS AUTHENTICATION

You can configure Kerberos authentication on a client that is using NFS v2, v3, or v4 by following these steps.

Fedora Core Linux

1. Copy **/etc/krb5.conf** from the KDC to the client.
2. Edit **/etc/sysconfig/nfs** to contain the string "**SECURE_NFS=yes".**
3. Start a client-side GSSAPI daemon by issuing the command ***service rpcgssd start*** to start a process called **rpc.gssd** for Kerberos.
4. Enable Kerberos authentication in PAM (Pluggable Authentication Module) by issuing the ***authconfig*** command. This automatically renews a user principal's ticket without having to manually issue the ***kinit*** command as soon as the user logs in.
5. As root, issue the following commands:

   To mount NFS v2:
   ```
   mount -o vers=2,sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
   ```

   To mount NFS v3:
   ```
   mount -o vers=3,sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
   ```

   To mount NFSv4 (for NFSv4 setup, see Appendix A):
   ```
   mount -t nfs4 –o sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
   ```

Note: At this time, Linux does not support the integrity (krb5i) and privacy (krb5p) Kerberos security services.

Solaris 9 and 10 (with SEAM Kerberos enabled)

- Copy **/etc/krb5.conf** from the KDC to the client.
- Edit **/etc/nfssec.conf** as follows, with the default security options as krb5:

```
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -           # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrit    # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy     # RPCSEC_GSS
default         390003  -       -       -                   # default is krb5
```

- As root, issue the following commands:

  To mount NFS v2:
  ```
  mount -o vers=2,sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
  ```

  To mount NFS v3:
  ```
  mount -o vers=3,sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
  ```

  To mount NFS v4 (for NFS v4 setup, see Appendix A):

```
mount -o vers=4,sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
```

Note: Solaris 9 and 10 support all three Kerberos security services: authentication (krb5), integrity (krb5i), and privacy (krb5p).

4.6 TROUBLESHOOTING

When you are troubleshooting the Kerberos integration, you need to look at the following areas.

Host Name Resolution and DNS domain

Make sure that the database on the DNS server contains all the fully qualified domain names of the client, server, and NetApp storage system. Also make sure that all three hosts are using the same DNS server.

Also, if any of the host exists in a different DNS domain (for example, **`client.engineering.netapp.com`**), you need to have an entry in the **`krb5.conf`** file to map the host to the correct Kerberos realm: for example:

```
[domain_realm]
 .netapp.com = NETAPP.COM
 netapp.com = NETAPP.COM
 .engineering.netapp.com = NETAPP.COM
```

Kerberos Realm

The Kerberos realm is case-sensitive, so make sure that all the letters of the realm are uppercase.

Time Synchronization

Because Kerberos relies on the timestamp in the tickets to authenticate principals, it is important that all the clocks on the hosts are synchronized. It is always a good idea to set up a time server and enable Network Time Protocol (NTP) on all machines to query time against the time server.

Encryption Type and Checksum Algorithms

Data ONTAP currently supports DES encryption algorithm for Kerberos. 3DES, although supported by MIT Kerberos, does not work with Data ONTAP.

Data ONTAP supports the CRC and MD5 checksum algorithms; however, some NFS client may not support both. For example, you cannot use the **`des-cbc-md5`** encryption type on a Linux NFS client.

Client Side Debugging

On Linux, you can see more logging messages by starting the **`rpc.gssd`** process with the **`-f -vvv`** options. This runs **`rpc.gssd`** in the foreground and increases the verbosity of the outputs.

5. KERBEROS AND LDAP

In the storage world, authentication is only half the answer to true data security. Authorization is equally important when it comes to serving data across the network. It is often easy to get confused between authentication and authorization. Authorization is a process to determine whether a user or client has the permissions to do certain things. By contrast, authentication is a process to verify a user's identity; in other words, to make sure that a person is indeed who she or he claims to be. For example, anyone over the age of 18 can apply for a driver's license from the Department of Motor Vehicles. The DMV authenticates the individual's identity and issues the driver's license. But whether the individual is authorized to purchase alcohol depends on whether she or he is over the age of 21.

Kerberos offers strong authentication for verifying a client's true identity; the Lightweight Directory Access Protocol (LDAP) offers the mechanism to look up authorized users in an efficient and secure manner. For more information about LDAP, see TR 3464: Integration of a NetApp Storage System with a UNIX Based LDAP Server and TR 3458: Unified Windows and UNIX Authorization Using Microsoft Active Directory LDAP as a Directory Store.

6. CONCLUSION

NFS v2 and v3 have long been seen as a nonsecure protocol. By integration with the Kerberos authentication system, NFS v2 and v3 can now achieve a much higher level of security. NF v4 mandates Kerberos and offers other security enhancements (in addition to the traditional UNIX-like permissions), such as Windows-style Access Control Lists (ACLs). Also, the client can specify whether authentication only (**`-sec=krb5`**), integrity (**`-sec=krb5i`**), or privacy (**`-sec=krb5p`**) should be used. These new security features in NFS add up to a much stronger NFS operation.

NetApp storage systems are fully committed to offering high data security by supporting Kerberos 5 authentication in all three NFS versions. This technical report has described the basic concepts of the Kerberos system, using real-life examples to help you set up Kerberos for a NetApp storage system.

APPENDIX A: NFS VERSION 4 AND KERBEROS

Use of NFS Version 4 (RFC 3530) in production environments is not encouraged at this time. Various implementations of NFS v4, including those discussed in this technical report, are still under active development and have yet to see widespread deployment. This appendix discusses the design goals of NFS v4 and how they relate to Kerberos security. It also provides both Data ONTAP and NFS client configuration examples to allow exploration of the protocol.

Feature Set Differences and Design Goals

Building on earlier versions of the protocol (RFC 1094|1813), NFS v4 has been redesigned with Internet deployment and security in mind. From a security standpoint:

- NFS v4 consolidates all of its operations into a single protocol with a single, well-defined tcp port (port number 2049). This eliminates reliance on the **`NLM`** (Network Lock Manager) and **`mountd`** external protocols, and introduces stateful file locking, file handle management, and client-side caching improvements. This simplifies deployment by consolidating services and ensuring a smaller footprint of external daemons that are open to exploitation.
- Coupled with using TCP as a transport, NFS traffic is now capable of easier, well-defined firewall traversal for exchanging data over the Internet. [TR 3085|RFC 3530] This makes it easier to deploy network security around NFS deployments.
- All implementations of NFS v4 must implement the **`RPCSEC_GSS`** protocol to be considered compliant with RFC 3530. This does not mean that each NFS v4 mount that a client initiates must use a secure authentication mechanism, merely that every implementation must be able to do so to be considered complete. As a point of contrast, NFS versions 2 and 3 can still be considered spec compliant without a secure authentication mechanism, and implementation is optional.
- By consolidating all of NFS v4 into a single protocol, **`RPCSEC_GSS`** security can be applied to every NFS v4 transaction. This differs from NFS Versions 2 and 3 because the traffic generated, and transits to and from auxiliary protocols, are not protected. This makes NFS v4 fundamentally more secure than previous NFS versions.

For a complete discussion of NFS v4 design and its features, see RFC 3530 and TR 3085: The NFS Version 4 Protocol.

Security Flavors

`RPCSEC_GSS` builds on the authentication flavors established in RFC 1831: Remote Procedure Call Protocol Version 2 by defining a final security flavor that allows RPC calls to use any security mechanism that works with the GSS_API model, established by RFC 2473. There are two primary security flavors associated with NFS v4, each suited to different types of deployment:

- **Kerberos Version 5:** The subject of this paper and the most commonly deployed form of NFS 4 security to date.
- **LIPKEY:** The Low Infrastructure Public Key (LIPKEY) provides an alternative security flavor for Internet-based deployments of NFS v4. Although Kerberos is exceptionally good at intranet and local network deployments, scaling beyond a multirealm implementation under the control of a single organization is difficult because of the need to arrange manual trust relationships with external realms.

For further details on authentication, security flavors, and their interactions with NFS, see RFC 2203: RPCSEC_GSS Protocol Specification.

NFS Version 4 Network Appliance Filer Configuration

Two additional Data ONTAP configuration options need to be set to utilize NFS v4:

- **`nfs.v4.enable`:** This setting enables the NFS v4 protocol on the filer. This is disabled by default in Data ONTAP 7.x.
- **`nfs.v4.id.domain:`** The ID Domain is used to provide the user name mappings required for tighter authentication integration mentioned in section 5. If the filer has been joined to an NIS domain, this value

is automatically set to the same domain. Otherwise, this value is set to the DNS domain name of the filer. It may be necessary to change the value of this option if your Kerberos realm does not match your DNS or NIS domain name.

To function properly with the example environment, set the above variables to the following configuration:

```
fas6070> options nfs.v4.enable on
fas6070> options nfs.v4.id.domain NETAPP.COM
fas6070> options nfs.v4
nfs.v4.acl.enable            off
nfs.v4.enable                on
nfs.v4.id.domain             NETAPP.COM
nfs.v4.read_delegation       off
nfs.v4.write_delegation      off
```

No additional changes to **/etc/exports** are required to use NFS v4; the syntax is exactly the same. For more information on mount option configuration with Data ONTAP with Kerberos, see section 4.5.

Fedora Core 4 Linux Client Configuration

All the necessary configuration steps for NFS v3 security are also necessary for NFS v4, and there are two additional requirements. This example assumes completion of the steps listed in section 4.5. The first additional step is to configure the file **/etc/idmapd.conf** to set the environment-specific settings. The **idmapd.conf** file controls the configuration and behavior of the **rpc.idmapd** daemon, which is responsible for mapping Kerberos-style user strings (for example, **james@NETAPP.COM**) to user ID and group ID numbers. The default file looks like the following example:

```
$ cat /etc/idmapd.conf

[General]
Verbosity = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs
Domain = NETAPP.COM

[Mapping]
Nobody-User = nobody
Nobody-Group = nobody

[Translation]
Method = nsswitch
```

The default file requires two configuration changes to suit the example environment.

[General]

- **Verbosity = 0**
  Currently only two verbosity levels are implemented, 0 and 1. Future versions of **idmapd** may include higher levels of detail. To enable verbosity, set this value to 1.

- **Pipefs-Directory = /var/lib/nfs/rpc_pipefs**
  This is where RPC communication pipes are stored and remain open for the duration of their existence.

- **Domain = NETAPP.COM**
  The domain entry serves to match the Kerberos realm component of the user strings correctly.

[Mapping]

Nobody-User/Nobody-Group = nobody
These settings specify the NFSv4 nobody user. Any root client request with the **root_quash** option set on an NFS mount is mapped to this user. This setting should be changed to **nfsnobody**, to match the default RHEL user that is created for this purpose.

[Nsswitch]

Nsswitch defines the method for mapping user IDs. There are currently two supported methods: **nsswitch** and **umich_ldap**. The **nsswitch** method is the default configuration, specifying the use of the standard **nsswitch** Linux library calls. The **umich_ldap** method is an LDAP schema extension that serves the same purpose and is highly experimental. Its syntax and implementation are still evolving.

When the configuration changes are complete, the file should look like the following example:

```
$ cat /etc/idmapd.conf

[General]

Verbosity = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs
Domain = NETAPP.COM

[Mapping]

Nobody-User = nfsnobody
Nobody-Group = nfsnobody

[Translation]
Method = nsswitch
```

The only remaining configuration change is to slightly modify the mount syntax with the previous example in mind:

```
fas6070> rdfile /etc/exports
/vol/volx       -sec=sys:krb5:krb5i:krb5p,rw
/vol/volx/krb5  -sec=sys:krb5:krb5i:krb5p,rw
```

Modify the mount command to use **nfs4** as the file system type:

```
mount -t nfs4 –o sec=krb5 fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
```

At this time, only **kb5** and **krb5i** are supported security types on Fedora Core 4. Future releases should include all forms of Kerberos security.

Solaris 10 Client Configuration

Solaris 10 NFS v4 client configuration is a very simple process, requiring only a single configuration file change to **/etc/default/nfs**. Like the Linux client implementation of NFS v4, Solaris requires a mechanism for mapping text-based string user names to their proper user ID and group ID numbers. This is done with a daemon called **nfsmapid**. Functionally, it fills the same technical need as **rpc.idmapd** on Linux.

To configure this daemon, edit **/etc/default/nfs** and change the setting **NFSMAPID_DOMAIN=domain** to match your Kerberos realm (**NetAPP.COM** in the case of the example). All that is left is to mount your NFS ß4 mount point:

```
mount –o vers=4,sec=krb5  fas6070.netapp.com:/vol/volx/krb5 /mnt/krb5
```

Unlike the Linux client, all versions of Kerberos security are supported at this time: **krb5**, **krb5i**, and **krb5p**.

APPENDIX B: KERBEROS GLOSSARY

Authenticator

An authenticator includes, among other fields, the current time (timestamp), a checksum, and an optional encryption key, all encrypted with the session key from the accompanying ticket. Because the timestamp is exchanged in the authentication, it is imperative that the system clocks on the application server, Kerberos server, and client are constantly synchronized.

GSSAPI

GSSAPI stands for Generic Security Services Application Programming Interface. It is a security framework used by Kerberos 5. GSSAPI provides the application vendor with a common API so that applications can work with any number of security systems.

Key (secret key)

A key or secret key is an encryption key that is shared by a principal and the key distribution center (KDC). The key is distributed outside the system with a long lifetime, as opposed to a session key, which has a much shorter lifetime. In the case of a user principal, the key is derived from a password. A keytab file contains the key for a service principal.

Key distribution center (KDC)

The KDC contains the identities and keys of every principal in the Kerberos realm. This principal information is stored in a local database in the MIT KDC. In some commercial products, a directory store such as LDAP or Microsoft Active Directory can be used to store the principal information. The KDC is often referred to as the Kerberos server. A typical KDC is a physical server that contains two logical service components,' which can also run on separate physical machines.

- **Authentication server (AS)**

  The AS issues authentication credentials, known as ticket granting tickets (TGT), to users when logging in to the Kerberos realm.

- **Ticket granting server (TGS)**

  The TGS issues service tickets to the users in response to their requests accompanied by TGT so that they can access various services in the Kerberos realm.

Principal

Every network entity in a Kerberos installation, including computers, users, and network services, has a principal associated with it. Each entity corresponds to a principal. In Kerberos 5, there are two types of principals: user and service principals. The user principal takes the form **<user>/<administrative instance>@REALM**, where the administrative instance is needed for an administrator who wants to be either a regular user or a Kerberos administrator. For instance, an administrator named James can have a regular principal **james@NETAPP.COM** or an administrative principal **james/admin@NETAPP.COM**. The service principal takes the form **<service>/<server>@REALM**, where service can be any Kerberized network service and server is the machine on which the service is running. For example, **nfs/fas6070.netapp.com@NETAPP.COM** is the nfs service principal for the NetApp storage device with a

fully qualified domain name **fas6070.netapp.com**. Kerberos principals are also referred to as Kerberos clients.

Realm

The Kerberos realm is usually the DNS domain name converted to uppercase; for example, **NETAPP.COM**. A Kerberos realm is case-sensitive. A Kerberos realm is a logical aggregation of principals.

Service ticket

Generated by the ticket granting server; clients use service tickets to authenticate with the application server before it can access services in the network. A service ticket includes session key, client principal, ticket lifetime, and client IP address.

Session key

A session is a temporary encryption key used between two principals. It has a short lifetime, limited to the duration of a single login session.

Ticket

A ticket is a record that helps a principal authenticates itself to a server. It contains information such as a client's identity, a session key, a timestamp, and other essential information that is required in an authentication process. All the information is sealed using the server's secret key.

Ticket granting ticket (TGT)

A TGT is an initial ticket obtained after a successful login. This ticket is used to get the service ticket to access a Kerberized service. With the TGT, users don't have to enter their password every time they want to connect to a Kerberized service, or even keep a copy of their password around. If a TGT is compromised, an attacker can masquerade as a user only until the ticket expires, limiting the extent of the damage caused by the attack.

APPENDIX C: REFERENCES

- TR 3445: Data ONTAP: Best Practices For Security Configuration
- TR 3464: Integration of a Netapp Storage System with a UNIX Based LDAP Server
- TR-3183: Using the Linux NFS Client with Network Appliance Filers
- Kerberos FAQ
- Data ONTAP File Access And Protocols Management Guide