NETAPP TECHNICAL REPORT

NFSv4 Enhancements and Best practices Guide - DATA ONTAP® implementation

NFSv4 in DATA ONTAP 7.3

Bikash R.Choudhury, NetApp

April 2008 | TR-3580

NFSv4 in DATA ONTAP 7.3

NFSv4 is based on RFC 3530. DATA ONTAP 7.3 follows and implements this RFC. This paper talks about the best practice principles that need to be followed while configuring different components of NFSv4 on the NetApp Storage system and clients mounting the file system

TABLE OF CONTENTS

1 INTRODUCTION

NFS (Network File System) version 4 is a distributed file system protocol based on NFS protocol versions 2 [RFC1094] and 3 [RFC1813]. Unlike earlier versions, the NFS version 4 protocol supports traditional file access while integrating support for file locking and the mount protocol. In addition, support for strong security (and its negotiation), compound operations, client caching, and internationalization have been added. Attention has also been applied to making NFS version 4 operate well in an Internet environment.

The goals of the NFS version 4 revisions are as follows:

- Improved access and good performance on the Internet
- Strong security with negotiation built into the protocol
- Good cross-platform interoperability
- Designed for protocol extensions

The general file system model used for the NFS version 4 protocols is the same as previous versions. The server file system is hierarchical; the regular files contained within are treated as opaque byte streams. In a slight departure, file and directory names are encoded using UTF-8 to deal with the basics of internationalization.

NetApp released the first NFSv4 server in Data ONTAP® 6.4. NFSv4 client implementations have changed since Data ONTAP 6.4 was released, and this has introduced some interoperability issues. DATA ONTAP 7.3 addresses many of these issues. Several client operating systems now support NFSv4 to varying degrees. The following table describes the features that have been added to DATA ONTAP NFSv4 support since its release.

DATA ONTAP does not have LIPKEY and SPKM3, and in the ACL support it does not support audit and alarm ACEs.

Release	**Features**
6.4.x	Basic NFSv4 support. All mandatory features except for LIPKEY and SPKM-3 and access and audit ACE.
6.5/6.5.1 and 6.5.2	Basic NFSv4 support. All mandatory features except for LIPKEY and SPKM-3 and access and audit ACE. Read delegation support added.
6.5.3/ 7.0	Basic NFSv4 support including ACLs. All mandatory features except for LIPKEY and SPKM-3 and access and audit ACE. Shipped with Write delegations.
7.0.1/7.1	ACL inheritance was introduced in Data ONTAP 7.0.1 and controlled through the nfs.v4.acl.enable option, but this was changed in Data ONTAP 7.1 and later releases, where ACL is always inherited if parent has inheritable ACL. Data ONTAP 7.1 also has more robust pseudo file system implementation.
7.3	Supports nested exports rules, UNICODE (UTF8), and Kerberized NFSv4 call-backs.

For more details on the NFSv4 structural changes, refer to TR 3085.

2 MOUNT OPTIONS

Solaris™10 clients attempt to mount NFSv4 by default. The `rsize` and `wsize` default to 64k.

```
'/mnt/b1 from 172.17.44.43:/vol/test_vol'

Flags:vers=4,proto=tcp,sec=sys,hard,intr,link,symlink,acl,rsize=65536,w
size=65536, retrans=5,timeo=600
Attr cache:    acregmin=3,acregmax=60,acdirmin=30,acdirmax=60
```

RHEL4.4/5 clients mount NFSv4 with a `-t` option, and the `rsize` and `wsize` default to 64k.

```
'mount –t nfs4 172.17.44.43:/vol/test_vol on /mnt/b1'


172.17.44.43:/vol/test_vol on /mnt/b1 type nfs4
(rw,vers=4,rsize=65536,wsize=65536,hard,intr,proto=tcp,timeo=600,retran
s=3,sec=sys,addr=172.17.44.43)
```

Note: RHEL5 (2.6.18 kernel) is highly recommended for NFSv4 implementation for the following reasons:

- Performance:
    - Clients do more work with fewer operations and invalidate the cache less often.
    - Clients move more data in fewer bytes and fewer CPU cycles.
    - There is a more efficient use of CPU and memory resources.
- ACL: The nfsv4_acls are more compatible with 2.6.18 kernel (RHEL5).

The other mount options that do not work with NFSv4 file systems on a Linux® client include “noacl,” “nocto,” and “nolock.”

2.1 LS –L LISTS THE OWNER AND GROUP AS NOBODY/NOBODY! WHY?

In NFSv3, client-server communication happens using numeric UID/GID. The client is responsible for mapping it back to the string representation; that is, the source of the GID/UID mapping specified in the `/etc/nsswitch.conf` file on the client. For example, user `root` has `uid = 0` and `gid = 0`. Now the client sends a CREATE request to the NetApp storage system to create a file called `‘foo’`. The UID and the GID are contained in the RPC layer and parameters, such as filename, attributers, etc., for the CREATE procedure embedded in the NFS layer. When the NetApp storage system gets the CREATE request from the client, it uses the UID and GID numbers from the RPC header and stores them together with the new file `‘foo’`. After the file `‘foo’` is created, the NetApp storage system returns the numeric UID and GID during every GETATTR request for that file from the client. The client then maps the corresponding UID and GID numbers that the NetApp system returns to its respective owner and group names after consulting the `/etc/nsswitch.conf` file for appropriate sources. Therefore, even if there is no `/etc/passwd` or `/etc/group` file on the NetApp system, the mapping of owner and the group IDs to their respective string names will succeed.

However, these semantics change in NFSv4. Exchange of the owner and owner’s group between client and server happens in the string format, therefore introducing mapping on the NetApp storage system as well as on the client. Looking at the same example as before, when a client tries to create file `‘foo’` on the NetApp system, it sends an `OPEN` operation to the system, instead of `CREATE`. The RPC will still have the numeric UID and GID, and the NFS layer will have an `OPEN` operation with the CREATE flag set to ON to create the file, as well as the `FATTR` parameter, which will contain root@iop.eng.netapp.com as the owner and root@iop.eng.netapp.com as the owner’s group. @iop.eng.netapp.com is the nfsv4 id-domain.

On the Linux client, this domain is configured in the `/etc/idmapd.conf (/etc/nfs/default` on a Solaris client) file by specifying following line:

```
“Domain=iop.eng.netapp.com”
```

On the NetApp storage system, the same domain should be specified in the `nfs.v4.id.domain` option.

When the domains are matched, the NetApp storage system maps incoming strings into numeric UID/GID; otherwise, `UID=65535/GID=65535` (nobody/nobody) will be used. When the domain check is passed, NSDB is consulted for mapping of the received string to numeric UID and GID. NSDB does the mapping according to the `/etc/nsswitch.conf` file. Because id-domains are matching, it is assumed that client and NetApp system will use the same source for this mapping, the same NIS or LDAP server, or the same set of `passwd` and `group` files. For simplicity, suppose that we are dealing with files, `/etc/passwd` will be consulted for UID for root, and `/etc/group` will be consulted for GID for root group. Once numbers are obtained, the file `'foo'` is created, and the numerical UID/GID is stored on disk. The reverse mapping will be performed when the client issues the `GETATTR` operation. We need to fill the `FATTR` parameter with the string value, and `nfsv4.id.domain` will be added to the end of the string name before it is sent it on the wire.

The following is a short demonstration on a NetApp storage system that is not connected to any NIS. This NetApp system has only the local files. Initially the NetApp system did not have an `/etc/passwd` file, and the `/etc/group` file had just this entry: `'daemon:*:1:'`. In this scenario, whenever a file was created on the NetApp system, it was marked as nobody/nobody for owner and group respectively. The `/etc/passwd` file on the NetApp system was added, and the root entry in the passwd and group entries on the client and the NetApp system were matched. If you are logging in as root or any other user, then there should be an identical entry in `/etc/passwd` and `/etc/group` on the NetApp system as well as on the client. However, you should not do this if the NetApp system and the clients are connected the same NIS server.

On the NetApp storage system:

```
fas960c-svl04*> rdfile /etc/passwd
root::0:1::/:
pcuser::65534:65534::/:
nobody::65535:65535::/:
ftp::65533:65533:FTP Anonymous:/home/ftp:

fas960c-svl04*> rdfile /etc/group
root:*:0:
daemon:*:1:
```

Verification on the NetApp storage system:

```
fas960c-svl04*> getXXbyYY getpwbyuid_r 0
pw_name = root
pw_passwd =
pw_uid = 0, pw_gid = 1
pw_gecos =
pw_dir = /
```

```
pw_shell =

fas960c-svl04*> getXXbyYY getgrbygid 0
name = root
gid = 0

fas960c-svl04*> getXXbyYY getgrbygid 1
name = daemon
gid = 1
```

Now when a file is created on the client mounting the NetApp storage system as root (in this example), the correct owner and group are listed.

```
[root@ibmx335-svl14 b3]# touch foo
[root@ibmx335-svl14 b3]# ls -l
total 96964600
-rwxr-xr-x  1 root root     114462720 Mar 30 11:10 722_sysfiles_i.tar
drwxrwxrwx  4 root root          4096 Mar 30 14:27 demo
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.1
-rw-r--r--  1 root root    2344615936 Mar 21 18:36 f.10
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.2
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.3
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.4
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.5
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.6
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.7
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.8
-rw-r--r--  1 root root   10737418240 Mar 17 01:46 f.9
-rw-r--r--  1 root root             0 Apr  6 16:07 foo
-rw-r--r--  1 root root        364544 Mar 20 05:36 typescript1.old
```

Before the change was made, all of these files were listed as nobody/nobody.

3 PSEUDO FILE SYSTEM IN NSFV4

UNIX® and most other systems mount local disks or partitions on directories of the root file system. NFS exports are exported relative to root or /. Early versions of Data ONTAP had only one volume, so directories were exported relative to root just like any other NFS server. When disk capacities grew to the point that a single volume was no longer practical, the ability to create multiple volumes was added. Because users don't log directly into the NetApp storage system, there was no reason to mount volumes internally to the NetApp system. To distinguish between volumes, the /vol/volname syntax was created. To maintain compatibility, support was kept for directories within the root volume to be exported without any such prefix, so /home is equivalent to /vol/vol0/home, assuming that vol0 is the root volume. / was the physical root of the system, and /etc was for the configuration information.

NetApp storage systems are among the few implementations, possibly the only one, requiring such a prefix on an export. In some implementations, this means that they can't simply drop the NetApp system in place of an existing NFS server without changing the client mounts. It may sound trivial, but it depends on how things are implemented in /etc/vfstab or automounter. In NFSv3, if you do not use the complete path from /vol/vol0, and you mount NetApp storage: /, the mount point is NetApp storage: /vol/vol0. That is, if the path does not begin with /vol in NFSv3, then Data ONTAP adds /vol/vol0 to the beginning of the path. In NFSv4, if you do not use the complete path from /vol/vol0 and you mount <NetApp storage: />, you mount the root of the pseudo fs and not /vol/vol0. Data ONTAP does not add /vol/vol0 to the beginning of the path. Therefore, if you mount <NetApp storage: / /n/NetApp storage> using NFSv3 and try the same mount using NFSv4, you would mount a different file system.

Therefore DATA ONTAP has the “/vol” prefix in the exported global namespace, and that “feature” represents an instance of use of the NFSv4 pseudo fs namespace. The traversal from the pseudo fs namespace to those of actual exported volumes is marked by a change in “fsid”. In the Data ONTAP implementation of the NFSv4 pseudo fs, the nodes "/" and "/vol" are always present and form the common prefix of any reference into the pseudo fs. Any reference that does not begin with "/vol" is invalid.

The NFSv4 protocol eliminates the separate mount protocol. The namespace visible to a client is traversed from a global “ROOT” (/”) file handle. Any parts of the namespace paths that cannot be traversed by a specific (or any) client are considered as part of an NFSv4 pseudo fs. The server makes available pseudo fs, which contains non-exported directories that lead to the actual export points.

The NFSv4 server has a known root file handle for the server available exported file systems that are visible from this global server root, by means of ordinary NFSv4 operations (for example, LOOKUP, GETATTTR) used within pseudo fs.

Here is the mount compound RPC that obtains the pseudo file system root file handle, and that obviates the need for mountd.

PUTROOTFH	Places the root of the pseudo FS as the current FH
LOOKUP	Used only if the client is not mounting the pseudo FS root
GETATTR	Returns the current FH attributes
GETFH	Return the current FH

The idea is that any client can mount the pseudo FS (/). Users then browse the pseudo FS via LOOKUP. Access into exported subtrees is based on the user's RPC credentials and file system permissions. The pseudo file system pulls together disjoint parts of the exported local file system into a coherent, browsable package for the client.

3.1 EXAMPLE OF PSEUDO FS

```
fas960c-svl03*> exportfs

/vol/test_vol/demo/test -sec=sys,rw,anon=0,nosuid

/vol/vol0/home  -sec=sys,rw,nosuid
```

```
/vol/vol0       -sec=sys,rw,anon=0,nosuid
/vol/test_vol/demo/test1 -sec=sys,rw,anon=0, nosuid
```

COMPOND LOOKUP for “/”,”vol”, generates “special” (pseudofs) fsid. “vol0” has a different fsid.

```
fas960c-svl03*> showfh4 -v /

/ (really /): exp.fileid=0x00042a exp.snapgen=0x000001 flags=0x05
snapid=0x0 unused=0x0 fileid=0x00042a gen=0x000001 fsid=0x000002
handle_type=0x03


fas960c-svl03*> showfh4 -v /vol

/vol (really /vol): exp.fileid=0x00042b exp.snapgen=0x000001
flags=0x05 snapid=0x0 unused=0x0 fileid=0x00042b gen=0x000001
fsid=0x000002 handle_type=0x02


fas960c-svl03*> showfh4 -v /vol/vol0

/vol/vol0 (really /vol/vol0): exp.fileid=0x000040
exp.snapgen=0x1da07d flags=0x00 snapid=0x0 unused=0x0
fileid=0x000040 gen=0x1da07d fsid=0xe802e1 handle_type=0x00
```

There will be a COMPOUND LOOKUP for “/”, “vol”, “test_vol”, “demo” that generates “special” (pseudofs) fsid for each of them.

NFSv4 servers avoid this namespace inconsistency by presenting all the exports within the framework of a single server namespace. An NFS version 4 client uses LOOKUP and READDIR operations to browse seamlessly from one export to another. Portions of the server namespace that are not exported are bridged via a "pseudo file system" that provides a view of exported directories only. A pseudo file system has a unique fsid and behaves like a normal, read-only file system.

From the above example, based on the construction of the server's name space, it is possible that multiple pseudo file systems may exist.

```
/vol                    -pseudo filesystem

/vol/test_vol           -pseudo filesystem

/vol/test_vol/demo      -pseudo filesystem

/vol/test_vol/demo/test    -real filesystem
```

It is the server's responsibility to present the pseudo file system that is complete to the client. If the client sends a LOOKUP request for the path `/vol/test_vol/demo/test`, the NetApp storage system response is the file handle of the file system `/vol/test_vol/demo/test`. In previous versions of the NFS protocol, the server would respond with the file handle of directory `/vol/test_vol/demo/test` within the file system `/vol/test_vol`. NFSv4 clients can detect export point crossing by change of fsid attribute. For each of these operations, there will be either a LOOKUP or a GETATTR operation associated in order to check that every component of the path is either a file or a directory. There will be access checks for these components.

When an NFS4 client tries to mount one of the exports from the NetApp storage system and the export is a subdirectory or a qtree of another export that does not grant the client permission to mount it, the request fails even though the higher level of the pseudo fs does grant the access. In this case, the two exports are exported with the following permissions, where /vol/vol0 has more restricted permissions than /vol/vol0/home:

```
        /vol/vol0    -rw=foo:bar
        /vol/vol0/home   -rw=gonzo
```

OR

```
        /vol/vol0   -sec=krb5,rw
        /vol/vol0/home -sec=sys,rw
```

In versions earlier than DATA ONTAP 7.3, when a client tries to access /vol/vol0/home as per the export rules listed in the above example over NFSv4, it gets “permission denied." However, this is not the case with NFSv3. This failure was due to access checks, because the request crosses each component of the access path. ‘krb5’ is a stricter security rule applied to /vol/vol0 that takes precedence over the weaker security ‘sys’ that applies to /vol/vol0/home, thereby denying permission to the client. DATA ONTAP 7.3 pseudo fs enhancements have addressed this problem.

Note: As a best practice, NetApp does not recommend having clients originating with a stricter permission or security addressed in the pseudo fs or root path, compared to the descendant paths that are exported separately with weaker security. There will a performance overhead in case the client is denied permission at the top level for additional security checks while using nested exports with different or stricter exports with security policies at different sub-export levels. NetApp does not recommend the use of nested exports, because not all clients support this feature.

The Snapshot™ directory in Data ONTAP is a pseudo file system. Snapshot copies provide online read-only clones of the active file system. Unlike other NFSv4 clients, Solaris10 cannot cross fsid change boundaries, and the content of the Snapshot directory is not visible unless explicitly mounted. This is bug 503540 on the Solaris client. However, to work around this problem, a new option can help in this situation—DATA ONTAP 7.3 allows the NFSv4 clients to access nested exports even if the top level has restrictive permissions or security.

The `‘-actual’` option in `/etc/exports` on the NetApp storage system is not supported in NFSv4.

The `‘–nosuid’` option in `/etc/exports` on the NetApp storage system is currently not supported either.

4 ACL IMPLEMENTATION WITH NFSV4

Access control lists (ACLs) allow access permissions to be granted or denied on a per-user basis for every file object. ACLs provide much finer grained access control and a greater degree of selectivity than the traditional UNIX mode permissions bits. ACLs can be set over CIFS or NFSv4. NFSv4 ACLs are based on the NT model, but they do not contain owner/group information. NFSv4 ACLs are made up of an array of access control entries (ACEs), which contain information regarding access allowed/ denied, permission bits, user name/group name, and flags. Flags, access masks, and type mostly map 1-to-1 to an NT ACL.

When a client sets an NFSv4 ACL on a file during a SETATTR operation, the NetApp storage system sets that ACL on the object, replacing any existing ACL. If there is no ACL on a file, then the mode permissions on the file are calculated from OWNER@, GROUP@, and EVERYONE@. If there are any existing SUID/SGID/STICKY bits on the file, they are not affected.

When a client gets an NFSv4 ACL on a file doing a GETATTR operation, the NetApp system reads the NFSV4 ACL associated with the object and constructs a list of ACEs and returns to client. If the file has an NT-ACL or mode bits, then an ACL is constructed from mode bits and is returned to the client.

While doing a permission check for a CIFS user trying to access a file that has NFSv4 ACL, the NT SID is converted into a UNIX ID, which is checked against the NFSv4 ACL. Similarly, when a UNIX user tries to access a file with NT ACL, the UNIX ID is converted to an NT SID, which is checked against the NT-ACL.

Access is denied if a DENY ACE is present in the ACL, and access is granted if an ALLOW ACE exists. However, access is denied if neither of the ACEs is present in the ACL.

4.1 INTERACTION OF MODE BITS, NFS-V4 ACLS, AND NT ACLS

Permission Checking:

- If a CIFS user tries to access a file that has an NFSv4 ACL, the permission checking is done from the ACL, by mapping the SID to a UID.
- If a UNIX user tries to access a file that has an NT ACL, the permission checking is done from the ACL, by mapping the UID to a SID.
- In a mixed qtree, the /etc/usermap.cfg file has to be used to map the NFS and Windows users. Now considering the above scenario, when a windows user only get to see the displayed ACL of “r” for certain users” on the file where it has to perform a write operation, then the usermanp.cfg file will map windows user to the UNIX and then check if the UNIX user is part of the groups that that has the permission to write. Once the access check is confirmed, the Windows user can write to the file, or else an error is returned. Therefore just read/write operations can be handled just fine with the help of the /etc/usermap.cfg file, but there will be a problem when there is an attempt to SETACL.

UNIX qtree:

- Only mode bits and NFS-v4 ACLs are considered for permission checking. NT-ACLs are ignored.
- CIFS is not allowed to change permissions on a file.
- If a file has mode bits and NFS-v4 sets an ACL on the file, the ACL will be created and the mode bits changed to reflect the new ACL. The new mode bits do *not* affect the SUID/SGID/STICKY bits that might be set for that file. All subsequent permission checks will be done from the ACL. On a GETATTR for mode request, the modified mode bits will be returned to the client.

- If a file has an NFS-v4 ACL and a UNIX client does a CHMOD to change the permissions on the file, the ACL will be dropped and the mode bits set. All subsequent permission checks will be done from the mode bits. On a get ACL request, an NFSv4 ACL will be created from the mode bits and returned to the client. Note: This NFSv4 ACL will be for display only and will not actually be set on the file.This ACL will be referred to as the fake ACL in the rest of the document.
- If a file has an NFS-v4 ACL and a UNIX client does a CHMOD to do a SUID/SGID/STICKY operation, the ACL is not dropped and these special permission bits are added to the mode. Similarly, if a CHMOD is done to remove only these permissions, the ACL is not dropped and the mode bits are modified accordingly.
- If a file has an NFS-v4 ACL and a UNIX client does a CHOWN or CHGRP, the ACL will not be affected.
- If a file has an NT-ACL (this can happen if the qtree security has been changed from MIXED/NTFS to UNIX), mode bits will be returned upon a GETATTR request. Similarly, a fake NFS-v4 ACL will be created from the mode bits upon a get ACL request.
- If a file has mode bits and a get ACL request comes in, a fake NFS-v4 ACL will be created from the mode bits and returned to the client.

NTFS qtree:

- Only mode bits and NT-ACLs are considered for permission checking. NFS-v4 ACLs are ignored.
- NFS is not allowed to change permissions on group, owner, or ACL on a file, as per the NTFS semantics on the NetApp system.
- If a file has an NT ACL and NFSv4 reads the ACL, the mode bits will be converted to an NFSv4 ACL and returned to the client. At this time we do not support translating an NT ACL into an NFSv4 ACL.
- If a file has an NFSv4 ACL, and a get ACL request comes in from CIFS, the NFSv4 ACL will be mapped toan NT ACL will be returned. The ACL displayed on the NT side may be incorrect because there is no mapping from NT groups to UNIX groups. Any such ACEs will not be returned to the NT client as part of the ACL. Also, any user ACEs where the UID could not be mapped to an NT SID are not returned.
- If a file has an NT ACL and a UNIX client sends a GETATTR request to get the mode permissions, the mode permissions calculated at the time of setting the ACL are returned.
- If a file doesn’t have an ACL and a CIFS or NFSv4 client makes a get ACL request, the mode bits are converted into a fake ACL and returned to the client.
- If a CIFS user tries to access a file that does not have an ACL, the checking is done from the mode bits. This is done by mapping the SID to a UNIX UID and then authenticating the UID against the mode bits.

Mixed qtree:

- A file will have either mode bits, or mode bits and NFS-v4 ACL, or mode bits and NT ACL.
- If a file has mode bits and either NFS-v4 or CIFS comes in to set an ACL, a set of "display" mode bits is created from the ACL. These mode permissions are returned when a UNIX client does a GETATTR. However, the permission checking is done from the ACL.
- If a file has an NFSv4 ACL and a UNIX client tries to do a CHMOD, the ACL is dropped. The special bits (SUID/SGID/STICKY) on the file remain intact. Note that changing the SUID/SGID/STICKY bits does not cause the ACL to be dropped. If an NFSv4 or CIFS client then tries to read the ACL, a fake ACL will be created from the mode bits and returned.

- If a file has an NT-ACL and an NFSv4 client makes a GETATTR request, the NT-ACL is not converted to an NFSv4 ACL; an ACL is calculated from the mode bits on the file and returned to the client.
- If a file has an NFSv4 ACL, when a CIFS client tries to read the ACL, the NFSv4 ACL is converted to an NT ACL and returned. The ACL displayed on the NT side may be incorrect because there is no mapping from NT groups to a UNIX group. Any such ACEs will not be returned to the NT client as part of the ACL. Also, any user ACEs where the UID could not be mapped to an NT SID are not returned.

  To further clarify, if a file has an NFSv4 ACL e.g: "r" for certain users, "w" for certain groups and "x" to another set of groups. When a windows client accesses that file it only sees that this file has "r" for certain users, the rest of the ACL is not displayed. Note that now if an application attempts to set the ACL on another file based on this displayed ACL information, it would actually be setting an incomplete ACL and not the same one as on the original file.

- For any NT user, the user's SID is mapped to a UNIX ID and the ACL is then checked for access for that UNIX ID. Regardless of what permissions are displayed, the actual permissions set on the file take effect and are returned to client.
- If a file has an NT-ACL and a UNIX client does a CHMOD, CHGRP, or CHOWN, the ACL is dropped.
- To list the actual permissions on the file, the `fsecurity show` command on the NetApp storage system provides the accurate information.

As a best practice, NetApp recommends using either UNIX or NTFS security-style qtree, because if the application depends on being able to read an ACL accurately from a file when the ACL could have been set from the other protocol, then it is best to use either the UNIX or the NTFS qtree. Otherwise, a mixed qtree should be fine.

4.1.1 The following example illustrates how the NFSv4 ACL is set on a file residing in a mixed qtree. The example replaces the entire ACL for the file bar, which gives the user "blue" read access, the file owner all access, the file group owner read-only access, the ACL mask read-only access, and others no access.

```
4.1.2    setfacl -s user:blue:rwx,user::rwx,group::rw-
     ,mask:r--,other:--- bar
```

4.2 WHEN THE VOLUME HAS UNIX SECURITY STYLE

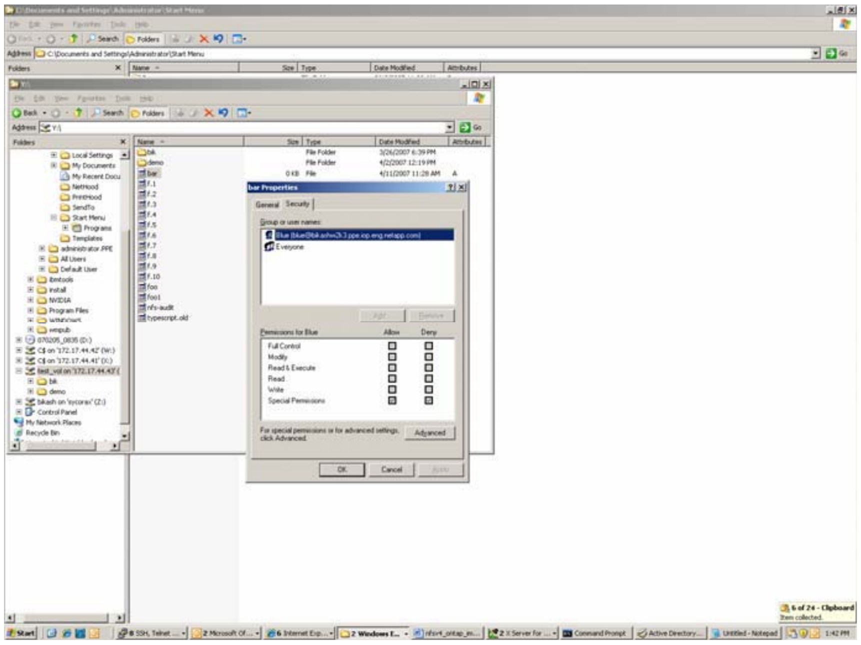
```
4.2.1    fas960c-svl03*> fsecurity show
     /vol/test_vol/bar -c

4.2.2    [/vol/test_vol/bar - File (inum 101)]

4.2.3      Security style: Unix

4.2.4      Effective style: Unix

4.2.5      DOS attributes: 0x0020 (---A----)

4.2.6      Unix security:

4.2.7        uid: 102 (blue)

4.2.8        gid: 100 (users)

4.2.9        mode: 0740 (rwxr-----)

4.2.10     NFSv4 security descriptor:

4.2.11       Control: 0x8004

4.2.12       DACL:

4.2.13         Allow - OWNER@ - 0x001601a7

4.2.14         Deny  - OWNER@ - 0x00000000

4.2.15         Deny  - uid: 102(blue) - 0x00040126

4.2.16         Allow - uid: 102(blue) - 0x001200a7

4.2.17         Deny  - uid: 102(blue) - 0x00040100

4.2.18         Deny  - GROUP@ - 0x00040126

4.2.19         Allow - GROUP@ - 0x00120087
```

```
4.2.20       Deny  - GROUP@ - 0x00040120
4.2.21       Allow - EVERYONE@ - 0x00120080
4.2.22       Deny  - EVERYONE@ - 0x00040127
```

4.3 *WHEN THE VOLUME HAS MIXED SECURITY STYLE*

```
4.3.1    fas960c-svl03*> qtree
4.3.2    qtree: This command is deprecated; using qtree
     status.
4.3.3    Volume   Tree     Style Oplocks  Status
4.3.4    -------- -------- ----- -------- ---------
4.3.5    vol0              unix  enabled  normal
4.3.6    vol0     test     unix  enabled  normal
4.3.7    test_vol          mixed enabled  normal
4.3.8    test_vol demo     unix  enabled  normal
4.3.9    bash-3.00$ pwd
4.3.10   /mnt/b1
4.3.11   bash-3.00$ getfacl bar
4.3.12   # file: bar
4.3.13   # owner: blue
4.3.14   # group: users
4.3.15   user::rwx
4.3.16   user:blue:rwx          #effective:r--
4.3.17   group::rw-             #effective:r--
4.3.18   mask:r--
4.3.19   other:---
```

4.3.20 On the Windows™ client, the file does not display the correct NFSv4 ACL.



Address: Y:\

Name	Size	Type	Date Modified	Attributes
bik		File Folder	3/26/2007 6:39 PM	
demo		File Folder	4/2/2007 12:19 PM	
bar	0 KB	File	4/11/2007 11:28 AM	A
f.1				
f.2				
f.3				
f.4				
f.5				
f.6				
f.7				
f.8				
f.9				
f.10				
foo				
foo1				
nfs-audit				
typescript.old				

bar Properties (Security tab)

Group or user names:
- Blue (blue@bik.ashw2k3.ppe.iop.eng.netapp.com)
- Everyone

Permissions for Blue	Allow	Deny
Full Control	☐	☐
Modify	☐	☐
Read & Execute	☐	☐
Read	☐	☐
Write	☐	☐
Special Permissions	▨	▨

For special permissions or for advanced settings, click Advanced.

4.3.21 The actual NFSv4 ACLs can be viewed by `fsecurity show` on the NetApp storage system.

```
4.3.22   fas960c-svl03*>fsecurity show
         /vol/test_vol/bar -c -d

4.3.23   [/vol/test_vol/bar - File (inum 101)]

4.3.24     Security style: Mixed

4.3.25     Effective style: Unix

4.3.26     DOS attributes: 0x0020 (---A----)

4.3.27     Unix security:

4.3.28       uid: 102 (blue)

4.3.29       gid: 100 (users)

4.3.30       mode: 0740 (rwxr-----)

4.3.31    NFSv4 security descriptor:

4.3.32       Control: 0x8004

4.3.33       DACL:

4.3.34         Allow - OWNER@ - 0x001601a7

4.3.35         Deny  - OWNER@ - 0x00000000

4.3.36         Deny  - uid: 102(blue) - 0x00040126

4.3.37         Allow - uid: 102(blue) - 0x001200a7

4.3.38         Deny  - uid: 102(blue) - 0x00040100

4.3.39         Deny  - GROUP@ - 0x00040126

4.3.40         Allow - GROUP@ - 0x00120087

4.3.41         Deny  - GROUP@ - 0x00040120

4.3.42         Allow - EVERYONE@ - 0x00120080

4.3.43         Deny  - EVERYONE@ - 0x00040127
```

4.3.44 While doing a `cp` and `mv` on file `bar` to different locations, it was noticed that `cp` changed the NFSv4 attributes for user ,`blue`. whereas `mv` did not.

```
4.3.45   bash-3.00$ getfacl bar
4.3.46   # file: bar
4.3.47   # owner: blue
4.3.48   # group: users
4.3.49   user::rwx
4.3.50   user:blue:rwx            #effective:r--
4.3.51   group::rw-               #effective:r--
4.3.52   mask:r--
4.3.53   other:---
```

4.3.54 File `bar` was copied from `/mnt/b1` to another location, `/mnt/b1/demo`. to here

```
4.3.55   bash-3.00$ cp bar /mnt/b1/demo/
4.3.56   bash-3.00$ cd demo
```

4.3.57 `getfacl` lists the new NFSv4 attributes, which do not match the attributes that the file `bar` had at its original location.

```
4.3.58   bash-3.00$ getfacl bar
4.3.59   # file: bar
4.3.60   # owner: blue
4.3.61   # group: users
4.3.62   user::rwx
4.3.63   group::r--               #effective:r--
4.3.64   mask:rwx
4.3.65   other:---
```

4.3.66 In the next step, file `bar` was moved from `/mnt/b1` to a new directory called `/mnt/b1/demo/testd/` using `mv`.

```
4.3.67 bash-3.00$ mkdir testd

4.3.68 bash-3.00$ mv ../bar /mnt/b1/demo/testd/

4.3.69 bash-3.00$ cd testd
```

4.3.70 `getfac'` lists all the NFSv4 for file `ba'` before and after the move.

```
4.3.71   bash-3.00$ getfacl bar
4.3.72   # file: bar
4.3.73   # owner: blue
4.3.74   # group: users
4.3.75   user::rwx
4.3.76   user:blue:rwx          #effective:r--
4.3.77   group::rw-             #effective:r--
4.3.78   mask:r--
4.3.79   other:---
```

4.4 ENABLING AND DISABLING NFSV4 ACLS

Use the `nfs.v4.acl.enable` option (disabled by default) to control the setting and viewing of NFSv4 ACLs.

```
options nfs.acl.enable on | off [off]
```

Note: The `nfs.v4.acl.enable` option does not affect whether an ACL is enforced and does not affect existing ACLs. The ACL will be enforced independent of the option value. The root user always has precedence over any ACL set on a file or directory.
In DATA ONTAP 7.3 the number of ACEs are set to 192 but can be increased to 400.

4.5 SETTING OR MODIFYING AN NFSV4 ACL

Use the `setfacl` command to set or modify an NFSv4 ACL.

To set an ACL granting the user "blue" read, write, and execute permission on file a, complete the following step.

On a Solaris10 update1 (Generic_118844-26 i86pc i386 i86pc) client:

```
bash-3.00$ touch drum
```

```
bash-3.00$ ls -l
total 231140464
-rw-r--r--   1 blue     users          0 Apr 11  2007 drum

bash-3.00$ setfacl -m u:blue:rwx /mnt/b1/drum
```

To view an NFSv4 ACL, use the `getfacl` command.

```
bash-3.00$ getfacl drum

# file: drum
# owner: blue
# group: users
user::rw-
user:blue:rwx           #effective:rwx
group::r--              #effective:r--
mask:rwx
other:r—

bash-3.00$ ls -l
total 231140464
-rw-r--r--+  1 blue     users          0 Apr 11 14:03 drum
```

On a Solaris10 update 3 (Sun® OS Generic_118855-33 i86pc i386 i86pc) client:

Starting Solaris10 update 3, chmod and "ls -v" (or "ls -V") are the preferred ways to set and get ACLs. These commands will handle both POSIX-draft and real NFSv4 ACLs. GETFACL and SETFACL will still be there, and they will still be limited to POSIX-draft, but in the future they will be removed.

The file permissions on file `foo2` before setting an ACL:

```
bash-3.00$ ls -v foo2

-rw-r--r--   1 green    users        0 Apr 19 13:33 foo2

     0:owner@:read_data/write_data/append_data/read_xattr/write_xattr

         /read_attributes/write_attributes/read_acl/write_acl/synchronize

         :allow

     1:owner@:execute/write_owner:deny
     2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
     3:group@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     4:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
     5:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
bash-3.00$ ls -V foo2
-rw-r--r--   1 green    users         0 Apr 19 13:33 foo2
            owner@:rw-p--aARWcC-s:------:allow
            owner@:--x---------o-:------:deny
            group@:r-----a-R-c--s:------:allow
            group@:-wxp---A-W-Co-:------:deny
         everyone@:r-----a-R-c--s:------:allow
         everyone@:-wxp---A-W-Co-:------:deny
```

Replacing ACLs entirely for user '`green`' on file `foo2`.

```
bash-3.00$ chmod A=owner@:read_data/write_data:allow,group@:read_data
/write_data:allow,user:green:read_data:allow foo2

bash-3.00$ ls -V foo2
-rw-rw----+  1 green    users         0 Apr 19 13:33 foo2
            owner@:rw------------:------:allow
            group@:rw------------:------:allow
        user:green:r-------------:------:allow

bash-3.00$ ls -v foo2
-rw-rw----+  1 green    users         0 Apr 19 13:33 foo2
     0:owner@:read_data/write_data:allow
     1:group@:read_data/write_data:allow
```

2:user:green:read_data:allow

On a Linux RHEL5 (2.6.18) client:

2.6.9-42 (RHEL 4.4) is not completely ready to support NFSv4 ACLs and delegations. Therefore we tested on a 2.6.18 RHEL5, and that release also does not work perfectly with NFSv4 ACLs because RHEL4.4/5.0 supports POSIX ACLs and the NetApp storage system does not. There is a separate package in the following link that must be downloaded. This package does a 1-to-1 mapping between the POSIX and NFSv4 ACLs.

"# `nfs4-acl-tools-0.3.0.tar.gz` has command line and GUI NFSv4 ACL tools, which deal directly with NFSv4 ACLs.

http://www.citi.umich.edu/projects/nfsv4/linux/

After downloading this package, follow the steps in this link to configure and install it.

http://www.citi.umich.edu/projects/nfsv4/linux/using-acls.html

However, this package does not work on RHEL4.4, which is based out of Fedora Core 3. After this package is installed on RHEL5, the ACLs work, but they have a different syntax.

You need to install the following package and configure it. You do not have to build the kernel, because this issue lies in the user space and not in the kernel. The correct syntax for setting the ACL on a file is `nfs4_setfacl -e` instead of just `setfacl`. This invokes the NFSv4 ACLs and not the POSIX ACLs.

After the package is installed, use –e to open the editor to modify the ACLs.

OR

Check the man pages for `nfs4_setfacl` (`nroff -man /usr/local/man/man1/nfs4_getfacl.1 |less`) for more examples of using this command.

```
[root@ibmx335-svl47 ~]# nfs4_setfacl -e /mnt/b3/testfile1

A::OWNER@:rwatTnNcCy
D::OWNER@:x
A:g:GROUP@:rtncy
```

```
D:g:GROUP@:waxTC

A::EVERYONE@:rtncy

D::EVERYONE@:waxTC
```

```
[root@ibmx335-svl47 b3]# nfs4_getfacl typescript.old
A::OWNER@:rwatTnNcCy
D::OWNER@:x
A:g:GROUP@:rtncy
D:g:GROUP@:waxTC
A::EVERYONE@:rtncy
D::EVERYONE@:waxTC
```

If clients are running on the latest kernel, an ACL set on a file from a Solaris client can be read from Linux and vice versa. The NFS clients depend on RPC for authentication using AUTH_SYS to identify users. There is a limitation on the NFS clients that any user can be part of 16 groups. However, the limit can be eliminated by using ACLs and authenticating using Kerberos. For more information, refer to the following link:

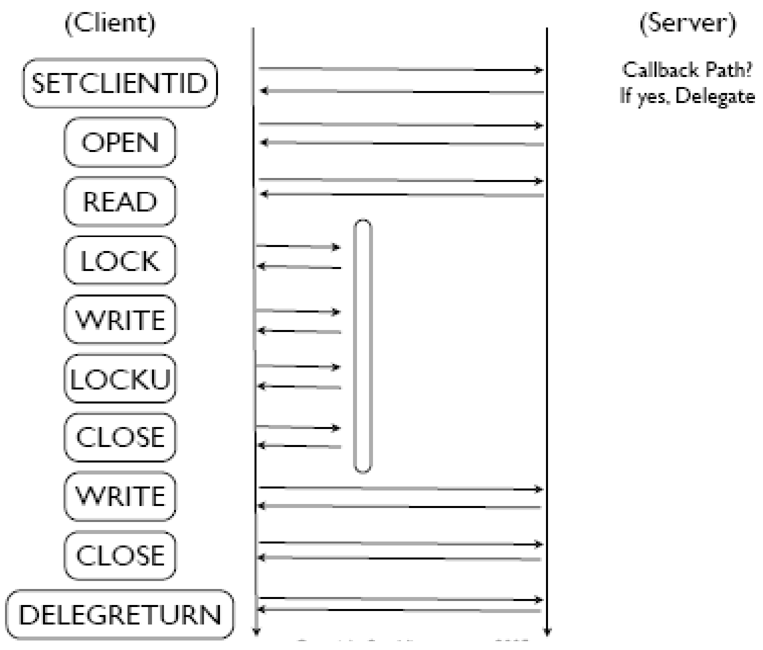
http://nfsworld.blogspot.com/2005/03/whats-deal-on-16-group-id-limitation.html

5 NFSV4 DELEGATIONS

NFS version 4 allows a server to *delegate* specific actions on a file to a client to enable more aggressive client caching of data and to allow caching of the locking state for the first time. A server cedes control of file updates and locking state to a client via a *delegation*. This reduces latency by allowing the client to perform various operations locally. Data ONTAP supports both read and write delegations. Directory delegations are not part of NFSv4. They are currently proposed for NFSv4.1.

Delegations can be recalled by the server. If another client (Snapshot with the Data ONTAP implementation; this includes clients using other versions of NFS and other protocols) requests access to the file in such a way that the access conflicts with the granted delegation, the server is able to notify the initial client and recall the delegation. This requires that a callback path exist between the server and client. If this callback path does not exist, then delegations cannot be granted. In Data ONTAP 7.3, the callbacks originate from ports greater than 1024; those are the nonprivileged ports. This release also supports Kerberos v4 callbacks as the same security flavor as incoming requests. If the RPCSEC_GSS fails, try doing AUTH_SYS. However, this option is turned OFF by default; you can do callbacks only using AUTH_SYS, because there is no client-side Kerberos callback support at this time.

Note: In RHEL5 the delegations seems to be broken. It is recommened to run on kernel 2.6.19-rc3 and later for the delegations to work properly.



5.1 GETTING A READ DELEGATION

The NFSv4 server in Data ONTAP will grant a read delegation in response to an OPEN for read, if no other client has the file open for read or denying read. This guarantees that no other client will be able to write to the file. If other clients open the same file for read-only access, they will be allowed to read the file. If another NFSv4 client supporting read delegations opens the file for read-only access, it will be granted a read delegation as well. For files being created, no delegation is returned on exclusive creates. This is because the client will issue a SETATTR request after the CREATE, which will cause the delegation to be recalled anyway, so as an optimization, we do not give out the delegation in the first place.

When using a read delegation, a client can do read-only opens and corresponding closes locally. It also doesn't need to poll the server to check modified times on the file, because no one will be allowed to modify the file. A lease is associated with a delegation. Upon lease expiration, the delegation's state goes away and any locks associated with the delegation are marked expired. If the client does not renew its lease within a certain time period (controlled by an option), these locks are revoked.

5.2 RECALLING A READ DELEGATION

A client can voluntarily return the delegation, or the NetApp storage system can recall the delegation in case of conflicting access to the file. This is done through a callback path established from the server to the client.

A read delegation is recalled when any of the following events occur:

1) OPEN for WRITE

2) OPEN denying READ

3) REMOVE, RENAME, SETATTR

4) WRITE

In Data ONTAP 7.3, a READ delegation can be recalled by the NetApp system when it is running low on resources, when a per-client limit on the locks is reached, or when the client has been idle for some time. When a delegation is recalled, there might be a number of opens that the client has done locally and now needs to propagate to the server. It will do that before returning the delegation. In Data ONTAP 7.3, when a client fails to return a delegation upon recall, the NetApp system will revoke the delegation and will not hand out any delegations to that client for 300 seconds by default.

When a conflicting request such as an OPEN for WRITE comes in for a file that has a read delegation, an NFSERR_DELAY/NFSERR_JUKEBOX error is returned if the request is coming from an NFSv4 client. If the request is coming over NFSv2/NFSv3/CIFS, the request is suspended waiting for the delegation to be recalled. When that is done, the suspended request is restarted and finally granted.

5.3 GETTING A WRITE DELEGATION

Write delegation gives exclusive access to a file to one client. The server will grant a write delegation if no other client has the file open. Once a client has a write delegation, it is guaranteed that no other client can access that file as long as the delegation remains valid.

When the client has a write delegation, it can be lazy about flushing the writes on file CLOSE, only if the server guarantees space for those writes. Clients choose not to flush the writes on CLOSE when the server space reservation information is not available. Currently the NFSv4 server in Data ONTAP does not make any such guarantees, so the client is required to flush everything on CLOSE.

The way delegations work, clients should do all OPEN, WRITE, LOCK, and UNLOCK requests locally. However, there are some clients, such as Solaris, that send LOCK and UNLOCK requests to the server even though they might be holding a write delegation for that file. The NetApp storage system associates such opens and locks with the delegation, and if the delegation is returned, these opens and locks are disassociated from the delegation state.

However, if a client decides to do locking/unlocking locally, it will have to send the lock state over when the delegation is being returned/recalled. That case is handled the same way as the one mentioned above.

When low on resources, the NetApp storage does not recall a WRITE delegation, unlike the READ delegation. The NetApp system reclaims the delegations as per RFC 3530.

5.4 PERFORMANCE IMPACT

Aggressive caching due to delegations can be a big help in environments exhibiting the following characteristics:

- Frequent opens and closes
- File locking
- Read-only sharing
- High-latency environment
- Fast client
- Heavily loaded server with many clients

5.5 AVAILABILITY IMPACT

NetApp storage system memory	# of NFSv4 client structures	Stateid - # of NFSv4 Open/Lock/Delegation structures	# of NFSv4 owners - PID
`= or > 16MB`	`16k`	`64k`	`16k`

On any NetApp system with 16MB or more of memory, the number of clients (the maximum number of NFSv4 client structures) is set to 16k, the number of stateids (the maximum number of NFSv4 Open/Lock/Delegation structures) is set to 64k, and the number of owners (the maximum number of NFSv4 Owner - PID of a particular client) is initially allocated 16K and then dynamically allocated up to a max of 64K in Data ONTAP 7.3. When the NetApp system reaches one of these limits and is not able to reclaim any state, the client will not be able to create a new state on this NetApp system.

5.6 TURNING ON READ/WRITE DELEGATIONS ON THE NETAPP STORAGE SYSTEM

The following two options enable the READ/WRITE delegations on the NetApp system:

```
Options nfs.v4.read_delegation       on
Options nfs.v4.write_delegation      on
```

When these options are turned ON, the NetApp system delegates READ or ACCESS to the client upon OPEN.

The following is a simple trace for a READ delegation where the client `172.28.2.82` is opening the file b. In the next line, the NetApp system issues a READ (DT=R) delegation to the client.

```
58   0.00030  172.28.2.82 -> 172.17.44.43 NFS C 4 (open) b PUTFH
     FH=5D51 OPEN b OT=NC SQ=1 CT=N AC=R DN=N OO=0045 GETFH
     GETATTR 10001a 30a23a
```

```
59    0.00337 172.17.44.43 -> 172.28.2.82  NFS R 4 (open) NFS4_OK
      PUTFH NFS4_OK OPEN NFS4_OK ST=0AA7:0 DT=R DST=0AA1:0 GETFH
      NFS4_OK FH=5E2D GETATTR NFS4_OK
```

The following is a sample trace where the NetApp system is providing a WRITE delegation (DT=W) to the client after the client attempts to open file b.

```
32    0.00036  172.28.2.82 -> 172.17.44.43 NFS C 4 (open) PUTFH
      FH=5D51 OPEN b OT=CR(U) SQ=9 CT=N AC=W DN=N OO=0017 GETFH
      GETATTR 10001a 30a23a

33    0.00725 172.17.44.43 -> 172.28.2.82  NFS R 4 (open) NFS4_OK
      PUTFH NFS4_OK OPEN NFS4_OK ST=0D03:0 DT=W DST=0D05:0 LB=SZ(0)
      GETFH NFS4_OK FH=5E2D GETATTR NFS4_OK
```

6 KERBEROS

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client-server applications by using secret-key cryptography. Kerberos was created at MIT as a solution to network security problems. Kerberos's fundamental approach is to create a service whose sole purpose is to authenticate. Data ONTAP uses RPCSEC_GSS with Kerberos for authentication, integrity, and privacy. For further details on Kerberos, refer to TR 3481.

Note: Make sure the Redhat client is running on at least nfs-util version 1.1 for Kerberos.

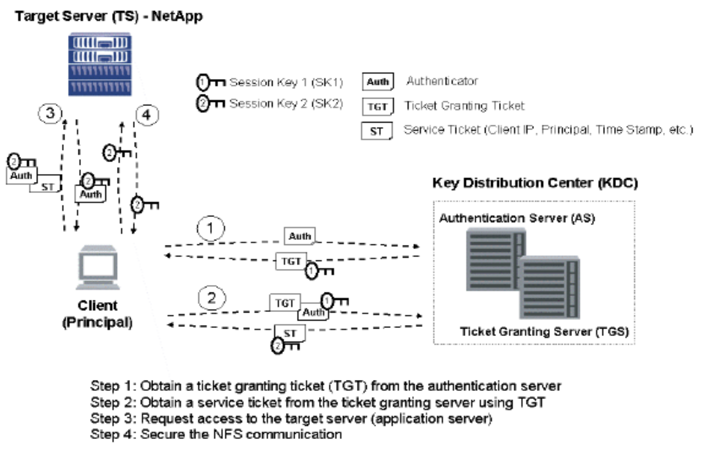
The following steps briefly describe the Kerberos authentication sequence between client and server using the Kerberos **Key Distribution Center (KDC).** (Terms in bold type are defined in the glossary.)

1. Client authenticates with KDC.

- Client contacts KDC (AS) in clear to get started (`kinit`).
- Gets back session key (encrypted with user key) for communicating with **TGS** or **TS**.
- Gets back **TGT** (encrypted with TGS key).
- Client contacts KDC (TGS) to get service **ticket**.
- Gets back session key for talking to the server (encrypted with session key that it got from AS).
- Gets back service ticket (encrypted with server's secret key).

2. Client sends service ticket to a server. (For NFS, RPCSEC_GSS is used.)

3. If it can decrypt the ticket, the server grants access to the client.


Target Server (TS) - NetApp
Session Key 1 (SK1)
Session Key 2 (SK2)
Auth Authenticator
TGT Ticket Granting Ticket
ST Service Ticket (Client IP, Principal, Time Stamp, etc.)
Key Distribution Center (KDC)
Authentication Server (AS)
Ticket Granting Server (TGS)
Client (Principal)
Step 1: Obtain a ticket granting ticket (TGT) from the authentication server
Step 2: Obtain a service ticket from the ticket granting server using TGT
Step 3: Request access to the target server (application server)
Step 4: Secure the NFS communication

The following two examples illustrate setting up Kerberos Key Distribution Center (KDC):

1) Using Active Directory as the KDC for NFS
2) Using SOLARIS10 as the KDC for NFS

1. Using Active Directory as the KDC for NFS

To set up a Key Distribution Center (KDC) for NFS using Windows Active Directory, you need:

1) Windows 2000/2003 AS with Active Directory and DNS configured
2) A Linux or a Solaris client to mount the file system and authenticate the hosts requesting it
3) A storage system that is exporting the file system with Kerberos security

The following example illustrates the steps to configure a Kerberos domain controller for NFS using Windows 2003 AS active directory.

1) The NetApp system is running on Data ONTAP 7.3, hostname fas960c-svl03.
2) The RHEL client is running on 2.6.18-8.el5 kernel (RHEL 5), hostname ibmx335-svl48.
3) Windows 2003 Active Directory and DNS are configured.
   i. Computer name: bikashw2k3as.bikashw2k3.ppe.iop.eng.netapp.com
   ii. Domain name:.bikashw2k3.ppe.iop.eng.netapp.com

On the Windows 2003 AD server the following changes need to be done.

1) On the Windows 2003 AD server, you need to create a "machine" credential for the Linux NFS client. Currently, Linux 2.6.kernels require a credential of the following form:
   `nfs/`*`hostname@REALM-NAME`*

A realm is a single Kerberos installation that defines namespaces for **`principal`**. This could be one or multiple DNS domain namespaces.

A **`principal`** defines a single entity with a Kerberos realm.

Use the Active Directory Management tool to create a new user account for the UNIX host:

- Select the **`Users`** folder, right-click, and select New, and then choose User**.**
- Type the name of the UNIX host.
- Select the Password Never Expires check box.

The principal **`nfs/ibmx335-svl48.iop.eng.netapp.com`** was created with the realm **`@bikashw2k3.ppe.iop.eng.netapp.com`** under USERS in Active Directory Users and Computers. A root user is also created in the same realm for the root to authenticate. Similar steps can be taken to create users.

Note: You must create this principal as type Users. Do *not* create the principal as type Computer. This Microsoft® document explains the reason.

http://www.microsoft.com/technet/prodtechnol/windows2000serv/howto/kerbstep.mspx

2) On the Linux client **`ibmx335-svl48`**, `/etc/krb.conf` was modified with the proper realm configuration to map with the credentials created in the Windows AD server. This file also maps the various DNSs with the KDC realm.

Originally, the `/etc/krb5.conf` file on the Linux client looks like the following:

```
[root@ibmx335-svl33 ~]# cat /etc/krb5.conf
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 forwardable = yes

[realms]
```

```
    EXAMPLE.COM = {
     kdc = kerberos.example.com:88
     admin_server = kerberos.example.com:749
     default_domain = example.com
    }

   [domain_realm]
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM

   [kdc]
    profile = /var/kerberos/krb5kdc/kdc.conf

   [appdefaults]
    pam = {
      debug = false
      ticket_lifetime = 36000
      renew_lifetime = 36000
      forwardable = true
      krb4_convert = false
   }
```

After editing, the `/etc/krb5.conf` file looks like the following:

```
[root@ibmx335-svl48 etc]# cat /etc/krb5.conf
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
 default_tkt_enctypes = des-cbc-md5
 default_tgs_enctypes = des-cbc-md5
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 forwardable = yes
```

```
[realms]
 BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM = {
  kdc = 10.35.8.57:88
  admin_server = 10.35.8.57:749
  default_domain = bikashw2k3.ppe.iop.eng.netapp.com
 }

[domain_realm]
 ppe.iop.eng.netapp.com = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
 .ppe.iop.eng.netapp.com = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
 .lab.netapp.com = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
 lab.netapp.com = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
 .iop.eng.netapp.com = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
 iop.eng.netapp.com = BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM

[appdefaults]
 pam = {
   debug = false
   ticket_lifetime = 36000
   renew_lifetime = 36000
   forwardable = true
   krb4_convert = false
 }
```

Note the following important points:

- The encryption type specifiers (*default_tkt_enctypes = des-cbc-md5 ;* or *des-cbc-crc* and *default_tgs_enctypes = des-cbc-md5 ;* or *des-cbc-crc*) cannot be omitted*.* Microsoft states that

  "Only DES-CBC-MD5 and DES-CBC-CRC encryption types are available for MIT interoperability."

- The [domain_realm] section that maps DNS domain names to the Active Directory realm is critical.
- Active Directory only supports uppercase realms. This is the case even though the screenshots of the Windows 2000 Active Directory tree show a lowercase domain name. Data ONTAP supports both lower and uppercase realms.

3) A Kerberos keytab file that includes an unencrypted list of principals and their keys needs to be created. Servers retrieve the keys they need from keytab files instead of

using `kinit`. A keytab named `linux_nfs_krb5.keytab` is created on the AD server using a `ktpass.exe` utility, which can be downloaded from http://www.microsoft.com/downloads/details.aspx?familyid=6EC50B78-8BE1-4E81-B3BE-4E7AC4F0912D&displaylang=en .

The NetApp storage system reads the keytab file when the first client tries to create an RPCSEC_GSS context. If this succeeds with GSS_C_ACCEPT, the credentials stay in the memory for 12 hours or less, depending on how `kinit` was done. The ticket can then be renewed, or a new ticket obtained if the current one expires. On Solairs10 and Linux clients, a message is logged on the console when the ticket is about to expire. Subsequent RPCSEC_GSS contexts creation gets serviced from the credential cache in memory.

```
C:\Documents and Settings\Administrator\Desktop>ktpass -princ
nfs/ibmx335-svl48.iop.eng.netapp.com@PPE.IOP.ENG.NETAPP.COM
mapuser BIKASHW2K3\nfs_ibmx335_svl48 -pass Wizard123 -crypto DES-
CBC-MD5 -out linux_nfs_krb5.keytab
```

```
Targeting domain controller:
bikashw2k3as.bikashw2k3.ppe.iop.eng.netapp.com

Using legacy password setting method

Successfully mapped nfs/ibmx335-svl48.iop.eng.netapp.com to
nfs_ibmx335_svl48.

WARNING: pType and account type do not match. This might cause
problems.

Key created.

Output keytab to linux_nfs_krb5.keytab:

Keytab version: 0x502

keysize 86 nfs/ibmx335-svl48.iop.eng.netapp.com@PPE.IOP.ENG.
NETAPP.COM ptype 0 (KRB5_NT_UNKNOWN) vno 4 etype 0x3 (DES-CBC-
MD5) keylength 8 (0xcb6279526b680245)
```

It is very important that the evaluation type (etype) is **`DES-CBC-MD5`** for the Linux clients to support.

4) The **`linux_nfs_krb5.keytab`** created on the Windows AD server is copied over to the Linux host as **`/etc/krb5.keytab`.**
5) On the Linux host, if the `/etc/sysconfig/nfs` file already exists, the following line needs to be included. If the file is does not exist, a new file must be created.

```
[root@ibmx335-svl48 /]# cat /etc/sysconfig/nfs

SECURE_NFS=yes
```

6) On the Linux host, check the following:

a. Make sure that `/etc/gssapi_mech.conf` exists. It should be installed by default.

```
 [root@ibmx335-svl48 /]# ls -l /etc/gssapi_mech.conf

-rw-r--r-- 1 root root 833 Jan 16 17:20
/etc/gssapi_mech.conf
```

b. The `rpcgssd` daemon needs to start and stop automatically. We use the `chkconfig` utility for this.

```
[root@ibmx335-svl48 /]# chkconfig --level 0123456
rpcgssd off

[root@ibmx335-svl48 /]# /etc/init.d/rpcgssd stop

Shutting down RPC gssd:
[  OK  ]

[root@ibmx335-svl48 /]# /etc/init.d/rpcgssd start

Starting RPC gssd:
[  OK  ]
```

c) Finally, check whether the `rpgssd` daemon is running.

```
[root@ibmx335-svl48 /]# ps -elf|grep rpc

5 S rpc       2806     1  0  75   0 -   445 -      Mar07
?        00:00:00 portmap

1 S root      3195     1  0  75   0 -  1233 322792 Mar07
?        00:00:00 rpc.idmapd

1 S root     11376    11  0  77  -5 -     0 worker 15:39
?        00:00:00 [rpciod/0]

1 S root     11377    11  0  70  -5 -     0 worker 15:39
?        00:00:00 [rpciod/1]

5 S root     12426     1  0  75   0 -   780 -      20:40
?        00:00:00 rpc.gssd -vvv

0 S root     12433 11203  0  78   0 -   972 pipe_w 20:42
pts/0    00:00:00 grep rpc
```

7) Use **`ktutil`** on the Linux host to display the contents of **`/etc/krb5.keytab`**. Use `rdate <time_server>` on the Linux host to sync the Linux host and the Windows AD server if there is a time skew. **`klist`** will display the ticket cache and proves to be a good diagnostic tool.

```
[root@ibmx335-svl48 etc]# ktutil

ktutil:  rkt krb5.keytab

ktutil:  l
```

```
      slot KVNO Principal
      ---- ---- -----------------------------------------------------
      1    5 nfs/ibmx335-
      svl48.iop.eng.netapp.com@BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM
      ktutil:q
      [root@ibmx335-svl48 etc]#

      [root@ibmx335-svl48 etc]# klist
      Ticket cache: FILE:/tmp/krb5cc_0
      Default principal: root@BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM

      Valid starting     Expires            Service principal
      03/15/07 17:11:37  03/16/07 03:11:53
krbtgt/BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM@BIKASHW2K3.PPE.IOP.ENG.N
ETAPP.COM
              renew until 03/16/07 17:11:37



      Kerberos 4 ticket cache: /tmp/tkt0
      klist: You have no tickets cached
```

8) On the NetApp storage system, configure Kerberos using the `nfs setup` command. There is no need for `krb5.conf` and `krb5.keytab` files on the NetApp system if option 2 is selected during the setup process. The NetApp system will obtain all the information from the Windows AD server and will keep it in the memory.

```
fas960c-svl03> nfs setup
Enable Kerberos for NFS? y
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):  2
        The default name for this NFS Kerberos server is
'FAS960C-SVL03'.
Would you like to change this name? [n]:
What is the name of the Active Directory domain?
[bikashw2k3.ppe.iop.eng.netapp.com]:
```

```
Fri Mar 16 01:28:49 GMT [kern.cli.cmd:debug]: Command line
input: the command is 'options'. The full command line is
'options kerberos.file_keytab.enable off'.

Fri Mar 16 01:28:49 GMT [kern.cli.cmd:debug]: Command line
input: the command is 'options'. The full command line is
'options nfs.kerberos.file_keytab.enable off'.

        In order to create an Active Directory machine account
for the filer,you must supply the name and password of a
Windows account with sufficient privileges to add computers to
the BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM domain.

Enter the name of the Windows user
[Administrator@BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM]:

Password for Administrator@BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM:

NFS Kerberos - Logged in as
Administrator@BIKASHW2K3.PPE.IOP.ENG.NETAPP.COM.

        An account that matches the name 'FAS960C-SVL03'
already exists in Active Directory: 'cn=fas960c-
svl03,cn=computers,dc=bikashw2k3,dc=ppe,dc=iop,dc=eng,dc=netap
p,dc=com'. This is normal if you are re-running NFS Kerberos
Setup. You may continue by using this account or changing the
name of this NFS Kerberos server.

Do you want to re-use this machine account? [y]: Fri Mar 16
01:28:49 GMT [kern.cli.cmd:debug]: Command line input: the
command is 'options'. The full command line is 'options
nfs.kerberos.file_keytab.enable off'.


Enter the NFS Kerberos server name for the filer [FAS960C-
SVL03]:

        An account that matches the name 'FAS960C-SVL03'
already exists in Active Directory: 'cn=fas960c-
svl03,cn=computers,dc=bikashw2k3,dc=ppe,dc=iop,dc=eng,dc=netap
p,dc=com'. This is normal if you are re-running NFS Kerberos
Setup. You may continue by using this account or changing the
name of this NFS Kerberos server.

Do you want to re-use this machine account? [y]: y

Fri Mar 16 01:29:48 GMT [cifs.kerberos.keytab:error]: CIFS:
Keytable information for Kerberos: Error during backup
restoration, could not find backup keytable.

Fri Mar 16 01:29:48 GMT [cifs.trace.GSS:error]: AUTH: Could
not restore old keytab after failed password change.

Fri Mar 16 01:29:48 GMT [kern.cli.cmd:debug]: Command line
input: the command is 'options'. The full command line is
'options nfs.kerberos.enable on'.

Fri Mar 16 01:29:48 GMT [kern.cli.cmd:debug]: Command line
input: the command is 'options'. The full command line is
'options nfs_kerberos.enable on'.

Kerberos now enabled for NFS.
```

```
NFS setup complete.
```

9) A `test_vol` volume is exported out from the NetApp system with Kerberos security.

```
fas960c-svl03> exportfs -io sec=krb5 /vol/test_vol
fas960c-svl03> exportfs
/vol/test_vol    -sec=krb5,rw
/vol/vol0/home   -sec=sys,rw,nosuid
/vol/vol0        -sec=sys,rw,anon=0,nosuid
```

The exported file system is mounted on the Linux client. The /proc/mounts list is as follows.

```
172.17.44.43:/vol/test_vol /mnt nfs
rw,vers=3,rsize=65536,wsize=65536,hard,proto=tcp,timeo=600,ret
rans=2,sec=krb5,addr=172.17.44.43 0 0
```

```
[root@ibmx335-svl48 etc]# ls /mnt
f1  f.1  f.10  f.2  f.3  f.4  f.5  f.6  f.7  f.8  f.9  foo
typescript.old
```

On the Solaris Key Distribution Center (KDC), the following configuration needs to be done.

1) Check whether the `krb5dc`, `kadmind`, and `ktkt_warnd` daemons are running on the KDC.

```
# ps   -elf | grep krb5
 0 S    root          254      1   0  40  20      ?    472    ?   Sep 28  ?        0:02
/usr/lib/krb5/krb5kdc
 0 S    root    249      1   0  40  20      ?   1066    ?   Sep 28  ?        1:16
/usr/lib/krb5/kadmind
 0 S    root   1578    232   0  40  20      ?    271    ?   Oct 03  ?        0:15
/usr/lib/krb5/ktkt_warnd
 0 R   root  10610  10593   0  50  20      ?    150   14:10:30   pts/1      0:00 grep krb5
```

2) Create a principal for the machine (storage system) using `kadmin.local` in the realm.

```
#  kadmin.local
Authenticating as principal root/admin@LAB.NETAPP.COM with password.
```

```
kadmin.local: addprinc -randkey -e des-cbc-crc:normal nfs/fas960c-
svl03-vif0.iop.eng.netapp.com@LAB.NETAPP.COM

WARNING: no policy specified for nfs/fas960c-svl03-
vif0.iop.eng.netapp.com@LAB.NETAPP.COM; defaulting to no policy

Principal "nfs/fas960c-svl03-vif0.iop.eng.netapp.com@LAB.NETAPP.COM"
created
```

3) Create the keytab using new principal from `kadmin.local`.

```
 kadmin.local:  ktadd -k fas960c-f sc vl03.krb4 5.keytab -e des-cbc-
-crc:normal nfs/fas960c-svl03-vif0.iop.eng.netapp.com@LAB.NETAPP.COM

Entry for principal nfs/fas960c-svl03-vif0.iop.eng.netapp.com
@LAB.NETAPP.COM with kvno 3, encryption type DES cbc mode with RSA-
CRC added to keytab WRFILE:fas960c-svl03.krb5.keytab.

kadmin.local:  quit
```

4) Create the root user to authenticate the realm.

```
kadmin.local:  ktadd -k fas960c-f sc vl03.krb4 5.keytab -e des-cbc-
crc:normal nfs/fas960c-svl03-vif0.iop.eng.netapp.com@LAB.NETAPP.COM

Entry for principal nfs/fas960c-svl03-vif0.iop.eng.netapp.com
@LAB.NETAPP.COM with kvno 3, encryption type DES cbc mode with RSA-
CRC added to keytab WRFILE: fas960c-svl03.krb5.keytab.

kadmin.local:  quit
```

5) Use `ktutil` to verify the keytab.

```
bash-2.05b# ktutil fas960c-svl03.krb5.keytab

ktutil:  rkt     l

slot KVNO Principal

---- ---- ---------------------------------------------------------

ktutil:  rkt fas960c-svl03.krb5.keytab

ktutil:  l

slot KVNO Principal

---- ---- ---------------------------------------------------------

   1    3 nfs/fas960c-svl03-vif0.iop.eng.netapp.com@LAB.NETAPP.COM
```

6) Edit the `/etc/krb/krb5.conf` file on the KDC.

```
/etc/krb5.conf

#
```

```
# Copyright 2004 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)krb5.conf  1.3     04/03/25 SMI"
#

# krb5.conf template
# In order to complete this configuration file
# you will need to replace the __<name>__ placeholders
# with appropriate values for your network.
#
[libdefaults]
        default_realm = LAB.NETAPP.COM
        default_tgs_enctypes = des-cbc-crc
        default_tkt_enctypes = des-cbc-crc
        permitted_enctypes = des-cbc-crc

[realms]
        LAB.NETAPP.COM = {
                kdc = kdcnfs1.lab.netapp.com
                kdc = kdcnfs2.lab.netapp.com
                admin_server = kdcnfs1.lab.netapp.com
        }

[domain_realm]
        .lab.netapp.com = LAB.NETAPP.COM
        lab.netapp.com = LAB.NETAPP.COM
        .eng.netapp.com = LAB.NETAPP.COM
        eng.netapp.com = LAB.NETAPP.COM
        .sim.netapp.com = LAB.NETAPP.COM
        sim.netapp.com = LAB.NETAPP.COM
        .hq.netapp.com = LAB.NETAPP.COM
        iop.eng.netapp.com = LAB.NETAPP.COM
        .iop.eng.netapp.com = LAB.NETAPP.COM

[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log
        kdc_rotate = {

# How often to rotate kdc.log. Logs will get rotated no more
# often than the period, and less often if the KDC is not used
# frequently.

                period = 1d

# how many versions of kdc.log to keep around (kdc.log.0, kdc.log.1,
...)

                versions = 10
        }

[appdefaults]
        kinit = {
                max_life = 2d
                max_renewable_life = 14d
                renewable = true
                forwardable= true
        }
        gkadmin = {
                help_url =
http://docs.sun.com:80/ab2/coll.384.1/SEAM/@AB2PageView/1195
```

7) Copy the `/etc/krb/krb5.conf` and the `/etc/krb/krb5.keytab` into the NetApp system.

On the NetApp storage system

1) Run `nfs setup` and complete configuring the NetApp system.

```
nfs setup

Enable Kerberos for NFS? y

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
There is no /etc/krb5.conf file yet. You will need to establish one.
It appears that CIFS has been set up to use
Kerberos. CIFS requires an Active Directory KDC.
If you want to use a UNIX KDC with NFS, you will
not be able to secure CIFS with a Kerberos KDC.

Do you wish to continue with setup for a UNIX KDC? y

Enter the Kerberos realm name: LAB.NETAPP.COM

Fri Mar 16 22:25:27 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
kerberos.file_keytab.realm LAB.NETAPP.COM'.

Fri Mar 16 22:25:27 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
nfs.kerberos.realm LAB.NETAPP.COM'.

Enter the host instance of the NFS server principal name [default:
fas960c-svl03.bikashw2k3.ppe.iop.eng.netapp.com]: fas

Fri Mar 16 22:25:57 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
kerberos.file_keytab.principal fas960c-
svl03.bikashw2k3.ppe.iop.eng.netapp.com'.

Fri Mar 16 22:25:57 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
nfs.kerberos.principal fas960c-
svl03.bikashw2k3.ppe.iop.eng.netapp.com'.

Fri Mar 16 22:25:57 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
kerberos.file_keytab.enable on'.

Fri Mar 16 22:25:57 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
nfs.kerberos.file_keytab.enable on'.

Fri Mar 16 22:25:58 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
nfs.kerberos.enable on'.

Fri Mar 16 22:25:58 GMT [kern.cli.cmd:debug]: Command line input:
the command is 'options'. The full command line is 'options
```

```
nfs_kerberos.enable on'.

NFS setup complete.
```

`nfs setup` fills in the values for these options automatically.

```
fas960c-svl03*> options nfs.kerberos
nfs.kerberos.enable          on
nfs.kerberos.file_keytab.enable on
nfs.kerberos.principal       fas960c-
svl03.bikashw2k3.ppe.iop.eng.netapp.com
nfs.kerberos.realm           LAB.NETAPP.COM
```

Note: Use caution in setting up the following options:

Options **nfs.rpcsec.ctx.high**

This option controls how many security contexts (RPCSEC_GSS sessions) you can have before they are reused. Setting this option too low can result in performance degradation and NFS outages. There were issues with the Solaris client when the option was set to 5, resulting in the client being unable to write to the volume.

The expected number for this option is in hundreds or thousands.

Options **nfs.rpcsec.ctx.idle**

This option controls how long to keep idle contexts around. The default of 6 minutes should be fine in most cases. Again, setting it too low will result in problems.

On the Solaris10 client, the following needs to be done:

1) Copy the `/etc/krb/krb5.conf` file from the KDC to this client.
2) Edit the `/etc/nfssec.conf` file. The following lines need to be uncommented:

```
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
default         1       -       -       -                       # default is AUTH_SYS
```

On the NetApp storage system, `/vol/test_vol` was exported with krb5 security.

```
fas960c-svl03*> exportfs
```

```
/vol/test_vol   -sec=krb5,rw
```

On the Solaris10 client, `/vol/test_vol` was mounted on NFSv4.

```
/mnt from fas960c-svl03-vif0.iop.eng.netapp.com:/vol/test_vol
 Flags:
vers=4,proto=tcp,sec=krb5,hard,intr,link,symlink,acl,rsize=65536,wsi
ze=65536,retrans=5,timeo=600
 Attr cache:    acregmin=3,acregmax=60,acdirmin=30,acdirmax=60


bash-3.00# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal:
root/ibmx335svl43.iop.eng.netapp.com@LAB.NETAPP.COM


Valid starting                Expires                Service
principal
03/16/07 14:51:36  03/18/07 14:51:31
krbtgt/LAB.NETAPP.COM@LAB.NETAPP.COM
        renew until 03/30/07 14:51:31
03/16/07 16:26:32  03/18/07 14:51:31  nfs/fas960c-svl03-
vif0.iop.eng.netapp.com@LAB.NETAPP.COM
        renew until 03/30/07 14:51:31
bash-3.00# ls /mnt
f.1             f.2             f.4             f.6             f.8
f1              typescript.old
f.10            f.3             f.5             f.7             f.9
foo
```

If there are any problems while configuring or in production regarding the functioning of Kerberos, a client-side trace along with the packet trace from the NetApp storage system will be helpful.

On RHEL4.4/5 clients:

`tethereal -i <interface> -f port 88`

`tcpdump -i <interface> -s 1517 port 88`

Add `-w <filename>` to save it to the file to `tcpdump` or `tethereal`.

On Solaris clients:

`snoop -d <interface> port 88`

Add `-o <filename>` to snoop to save it in a file.

7 GLOSSARY: KERBEROS TERMINOLOGY

- KDC: Key Distribution Center
- AS: Authentication Server, gets you going
- TGS: Ticket Granting Server, gets you your service tickets
- Database: Stores all of the information about your realm
- kadmin: Server for administrative tasks
- krb425d: Server for converting krb4 tickets to krb5 tickets (optional)
- Tickets: In general, your license that you are who you say you are:
  - TGT: Ticket granting ticket (main ticket that allows you to obtain other types of tickets)
  - Session ticket: Type of ticket that gives you access to a specific service
- Principal defines a single entity within a Kerberos realm:
  - Users: jdoe/admin@NETAPP.COM
  - Host: host/athyra.lab.netapp.com@NETAPP.COM
  - Services: nfs/athyra.lab.netapp.com @NETAPP.COM
- Realm: A single Kerberos installation that defines the namespace for principals

8 REFERENCES

http://www.ietf.org/rfc/rfc3530.txt

http://www.nluug.nl/events/sane2000/papers/pawlowski.pdf

http://web.netapp.com/engineering/design-depot/nfs/

http://ngs-web.corp.netapp.com/service_readiness/product_ready/products/toi/DataONTAP/Scrimshaw/NFS/delegations_toi.ppt

http://web.netapp.com/~saadia/dlg.ppt

http://web.netapp.com/~saadia/acls.ppt

http://ngs-web.corp.netapp.com/service_readiness/product_ready/products/Tsingtao/Tsingtao.0/toi/nfs/acls_toi.ppt

http://web.netapp.com/~burlyga/nfs_kerberos_setup.ppt

http://nfsworld.blogspot.com/2005/06/using-active-directory-as-your-kdc-for.html