Issue Date: IBM Edition for UNIX
August | September 2005, Posted On: 8/2/2005

Securing Access

Jaqui Lynch

When it debuted, AIX* 5.3 introduced many new features and functions. One of the more significant additions from a security perspective was the introduction of IBM's implementation of Network File Server (NFS) v4. NFS has been around for many years and is a proven, easy way to share files among multiple systems.

NFS v4 addresses many of the security concerns that were endemic to previous versions of NFS by adding better authentication methods (i.e., Kerberos), better user-name mapping and the ability to encrypt the transmissions taking place. Additionally, the requirement to use TCP rather than User Datagram Protocol (UDP) provides a more stable environment.

NFS is a distributed file system that allows access to files and directories on remote computers as if they were local. A simple NFS environment consists of a server and client system. The server defines and exports one or more directories, file systems or files, and the client system mounts them for local user access. For this to work, NFS make uses networking protocols, daemons on the client and the server, and extensions in the kernel.

Protocols
NFS takes advantage of two protocols at the OSI session layer—Sun Remote Procedure Call (RPC) and External Data Representation (XDR); and two protocols at the transport layer—User Datagram Protocol (UDP) and TCP. NFS v2 and v3 have the capability to communicate over UDP or TCP with the default for v2 being UDP and TCP for v3. Since UDP is a stateless protocol, all requests are treated as independent transactions but delivery isn't guaranteed and connections and their state aren't maintained. This caused many people to avoid using NFS because it was seen as unreliable and unstable. The migration to TCP, which is a stateful protocol, enables the server to obtain better performance as well as a more reliable connection. Since a connection is established and maintained between the server and client, it's unnecessary to go through the re-establishment procedures for every transaction. Also, with state being maintained, it's possible to send data in a continuous stream and guarantee its delivery. NFS v4 only works with TCP—UDP is no longer offered.

The key to the functionality within NFS is the use of remote procedure call (RPC), which is a library of procedures that can be used by the client to ask the server to execute other processes on its behalf. These processes can be simple list requests or more complicated move and copy requests.

Also keep in mind that the server and client don't need to be on the same platform. Since different platforms have different ways of representing data, it's critical to have a translation mechanism. This is performed using XDR. Data is converted into XDR format when it's sent to a remote system, and on receipt, it's re-converted into the local format.

Daemons
NFS depends on processes called daemons to function properly. Daemons run in the background to provide services to the system and its users and applications. NFS depends on several daemons, depending on whether the system is a server or client. By default, the server runs the following daemons:

nfsd—This daemon manages the server. It takes receipt of the requests coming in and returns the results to those requests. The actual requests aren't processed by the nfsd—they're handed off to a kernel process. Each nfsd is dedicated to a request from the time the request is received until the results are returned to the requestor. If there are no requests, the daemons sleep.

rpc.statd—This daemon works with the rpc.lockd daemon to provide locking functions.

rpc.lockd—This daemon processes lock requests and asks the rpc.statd daemon to monitor those requests.

rpc.mountd—This daemon handles mount requests coming in from the client. It also keeps track of currently mounted file systems and which systems have them mounted.

The client runs the following daemons:

biod—Also called the block I/O daemon, this daemon sends the client requests to the NFS server for reads and writes. One

biod is used for each read or write request.

rpc.statd—See definition above.

rpc.lockd—See definition above.

portmap—Run by the server and client, this daemon is used by RPC to register which ports the various servers and daemons are listening on and which RPC numbers each port services. If this daemon isn't started, NFS won't work. Also, portmap should be started before inetd because inetd starts several RPC servers.

With an NFS v4 implementation, there are some new
daemons to consider, including:

nfsrgyd—Provides name translation services for NFS servers and clients, translating between NFS string attributes and UNIX* numeric identities. This removes the requirement for user names to have the same user IDs (UIDs) on each system. As long as the user name is unique, the translations take place and security is maintained.
gssd—In AIX, Kerberos v5 security is provided using general security services (GSS). This comes on the AIX expansion pack and is required for Kerberos to work.

What's in NFS v4?
Key areas addressed by the NFS v4 specification include
performance enhancements, additional security schemas, locking support, more cross-platform support (specifically Windows*), continued improvement while maintaining backwards compatibility and a move toward NFS becoming an open standard. In a pure NFS v4 environment, the integration of locking into the main protocol means rpc.lockd, rpc.statd and rpc.mountd are no longer needed. Other key additions in NFS v4 include:

1. RPCSEC-GSS RPC and support for identity mapping of foreign domains
2. RPC-locking operations moved into the protocol
3. Support for TCP only
4. Access control language (ACL) support
5. Only file systems or directories can be exported. Files can't be exported by themselves.

IBM's Implementation in AIX v5.3
IBM made significant changes with AIX v5.3. However, full support is still maintained for NFS v2 and v3. In fact, NFS v3 is still the default when an NFS file system is defined. In the initial release of NFS v4, IBM included many of the core features including the RPCSEC-GSS RPC authentication flavor supporting the Kerberos v5 security mechanism.

IBM added support for most of the mandatory portions of the NFS v4 protocol. One notable exception is that LIPKEY and SPKM-3 security mechanisms aren't supported with RPCSEC-GSS authentication—only Kerberos v5 is supported in this initial release. Also, delegation isn't supported in the initial implementation. Other features not supported include diskless clients, Network Install Manager (NIM) and UDP.

Support for ACLs was provided for the client and server, and these can be managed using line commands (acledit, aclget and aclput). This support is provided for enhanced journaled file system (JFS2) with extended attribute version 2 file systems only.

An additional feature in NFS v4 is the use of a pseudo root. Prior versions of NFS exported the individual directories or file systems, and the client had to mount each one separately. With NFS v4, the server creates a pseudo file system that creates a single directory tree for all of the exported directories. The client then mounts the pseudo root and obtains access to the exported data that it's entitled to see.

In the AIX implementation, NFS v3 is the default so the vers= parameter must be used to switch to NFS v4. Also, the AIX implementation uses the rpc.mountd daemon on the server for some of the access handling.

Key Files
NFS depends on many configuration files. Files such as /etc/exports have been used for many years, however there are some new files that users must understand. The following is a list of the key files in an NFS environment.

Old files:

/etc/exports—Contains the list of exported directories and file systems and their export parameters.

/etc/xtab—This file lists the current status of exported directories and is updated whenever the exportfs command is run.

New files:

/etc/nfs/hostkey—This file is on the server and is used to define the Kerberos host principal and the location of the keytab file. This data is necessary when a client or server wants to communicate. To do so using NFS v4, they must acquire credentials so the host principal can accept requests. This file is created using the nfshostkey command.

/etc/nfs/local_domain—This file contains the name of the local NFS domain for this system. This file should always be changed or edited using the chnfsdom command so that the nfsrgyd daemon is notified of the changes.

/etc/nfs/realm.map—This file is key to the mapping functions used by NFS v4. It maps the incoming Kerberos domain principals, which are in the form of name@realm, to the local system's NFS domain, in the form of name@nfs-domain. Once the principals are remapped, they can be turned into local UNIX user names and access can then be provided accordingly.

/etc/nfs/princmap—This file is created and managed by the nfshostmap command and used to map host names to Kerberos principals if the principal isn't the fully qualified server name.

/etc/nfs/security_default—This file is created and managed by the chnfssec command and contains a list of the security protocols to be tried in the order they're to be used. Normally the first option would be krb5 and the last would be sys (for system).

Securing NFS

The new features of NFS, along with system enhancements, can be used to increase NFS security. Initially the system and ports should be secured. Port security and network security using TCP Wrappers should be implemented as a first start. Access to portmap can then be secured using TCP Wrappers, limiting those who can query portmap. Additionally, the variable NFS_PORT_RANGE= can be defined in /etc/environment to limit which ports can be used for NFS access, making it much easier to use firewalls.

By default, user nobody and group nobody is the ID assigned to any unauthenticated user. The variable NFS_NOBODY can be set to change this to a different UID.

When exporting directories, keep these options in mind:

1. What is the pseudo root?
2. Who will have access?
3. The use of fully qualified host names
4. Type of export (e.g., RO, RW, etc.)
5. Which version of NFS? (You need to code the vers=4 option if you want to use v4.)

Exports should also be done using fully qualified host names, especially if outside-the-network access will be allowed. For example, if you export a file system to a system named wombat, any system with wombat as its first qualifier can obtain access. If you export it to wombat.111.com, only that server can access it. In NFS v4, an additional level of security is added as the server now must authenticate using GSS, but this is still an important level of protection.

User Access

In earlier versions of NFS, security was based on the system's UID and group ID (GID), the numbers assigned by the system to the user name and group name. If these weren't unique across all systems (i.e., Jaqui is 123 on all systems), it was possible for an unauthorized person to get access to files. As an example, on wombat Jaqui is UID 123 and creates a file called Jaqui.txt with permissions rwx —- —-. On freddy Jaqui is UID 124 while Trish is UID 123. Trish can access the file, but Jaqui can't.

The new user name to UID-mapping model tries to take care of this problem. In NFS v4, users and groups are represented as user@nfs-domain or group@nfs-domain, not as UIDs and GIDs. If there's no entity that matches, the UID and GID are set to nobody. User authentication defaults to AUTH_SYS where the client does authentication, normally using logon name and password. The other option is to use RPCSEC_GSS (Kerberos v5) for authentication, which is the most secure option.

Kerberos
Kerberos authentication is a key element of NFS v4 enhanced security. It's a network-authentication service that allows users and hosts (called principals) to prove they are who they say they are. The intent is to provide authentication while protecting data. In an NFS environment, the client and server establish a "security context." That context is a structure that contains the information proving the client and server have mutually authenticated. Additionally, it contains encryption keys that are used for encrypting and decrypting the data being sent and received. Thus the use of NFS v4 with the RPCSEC_GSS expects that a Kerberos environment has been set up.

ACLs
In the NFS world, access was restricted using the standard permissions that control read, write and execute access by user, group and other. ACLs provide a more granular way to limit access, particularly in the group access where they allow different access rules for multiple groups on the same file or directory. In an AIX system, the JFS2 EAv2 and GPFS file systems support the use of ACLs with NFS v4. For JFS2, an ACL is limited to 64 KB. Administration of ACLs can be done via line commands, SMIT or the Web interface (WEBSM).

Take Another Look
With the introduction of NFS v4, many of its past security concerns have been addressed. If you haven't explored the use of NFS previously because of security concerns, now is the time to start testing NFS v4.

Jaqui Lynch, *a senior systems engineer focusing on pSeries* and Linux* at Mainline Information Systems, has worked in the IS industry for more than 28 years. She's been responsible for a wide variety of projects and OSs across multiple vendor platforms, including mainframes, UNIX systems, midrange systems and personal workstations. Jaqui can be reached at jaqui.lynch@mainline.com.*