CONF-9109276--3

NFS, Kerberos, and UNICOS

SAND--91-2115C

DE92 000246

Rena A. Haynes
Sandia National Laboratories, Albuquerque

OCT 07 1991

A presentation to Cray User Group, Inc.

September 25,1991

Abstract

The Network File System (NFS™) is used in UNIX®-based networks to provide transparent file sharing between heterogeneous systems. Although NFS is well-known for being weak in security, it is widely used and has become a *de facto* standard. This paper examines the user authentication shortcomings of NFS and the approach Sandia National Laboratories has taken to strengthen it with Kerberos™. The implementation on a Cray Y-MP8/864 running UNICOS® is described and resource/performance issues are discussed.

1.0 Why NFS?

The NFS service was designed by Sun Microsystems, Inc., to provide transparent access to remote files and directories in a heterogeneous networking environment. The primary goal of the NFS designers was to provide a way of making remote files available to local programs without requiring special parsing, libraries, or recompilations.[1] This goal was accomplished by providing a virtual file system interface on the local machine which would detect accesses to remote files and would utilize system independent networking protocols to request file services of the remote machine. Robustness was accomplished by requiring the local machine to continue issuing a request until a response was received or a predetermined retry count was exceeded. Performance equal to that of a local workstation disk was achieved by using networking protocols with low overhead.

Although NFS was designed to be system independent, it has been implemented primarily on UNIX-based platforms, where NFS supports UNIX filesystem semantics for UNIX client systems. The concept that all files can be dealt with as local files provides a powerful and irresistible tool for programmers. For this reason, NFS has become a *de facto* standard, and is expected to be supported in a UNIX networking environment.

In a supercomputer environment, NFS can be used to extend the supercomputer file space to scientists' desks. This can be done by the supercomputer acting as a client, mounting the scientists' local file space, or by the supercomputer acting as a server, exporting its file space to the scientists' local machines. Both ways are important—the first to provide additional cost effective storage for the supercomputer and the second to provide scientists with ready access to programs and data on the supercomputer.

Care must be taken when the supercomputer is the client to ensure that valuable cpu cycles are not wasted on retrying requests to low performance servers or servers with unreliable or slow communication paths. Sandia National Laboratories in Albuquerque (SNL,A) determined that until a high performance file server [2] with fast and reliable communications was acquired, the Y-MP8/864 would in general only support server NFS.

2.0 NFS and Security

Since NFS was designed to provide a simple mechanism for transparent remote file access, little emphasis was placed on security concerns. As NFS was originally designed, a trusted network was assumed. Every user and every machine were who they said they were. Or if they weren't, it was okay. The only "security" features provided were to protect one from oneself. The server could specify which machines could mount its filesystems and how these machines could access the files. Any user on a client system could access remote files as any effective user identifier by merely becoming superuser and then setting his user identifier accordingly. Only the superuser identifier itself could be restricted.

Although NFS assumes a trusted network with trusted users, the session level protocol RPC allows authentication of requests. RPC authentication is provided at the time of the request by the client system presenting the server with the user's credentials, for example, the user's name, and a verifier, for example, a driver's license. For NFS, the credentials provided are the user's effective user identifier (uid), the group identifier (gid), and groups list. Unfortunately, NFS provides no verifier for these credentials. In order to provide NFS in Sandia's secure environment, a user's identity must be authenticated before a request is serviced.

3.0 Kerberos and NFS

3.1 Project Athena and NFS

In the 1980s, M.I.T. created Project Athena, whose primary technical goal was to create a computing environment built around high-performance workstations, high-speed networking, and network servers.[3] In building this network environment, the Kerberos authentication service was developed. Kerberos provides user authentication in three phases as shown in Figure 1.

In the first phase, the user obtains a packet to be used to request access to other services. In the second phase, the user requests authentication for a specific service and receives a packet called a ticket. In the final phase, the user sends the ticket received in phase two to the server, who verifies its contents and grants or denies access.

By 1987, Project Athena was using Kerberos to authenticate 5000 users on 650 workstations for services on 65 network servers.[4] One of the most important services on the Project Athena network was transparent file access via NFS. In order to implement an efficient NFS which would not require kernel modifications to client systems, Project Athena developed a hybrid approach for Kerberizing NFS.

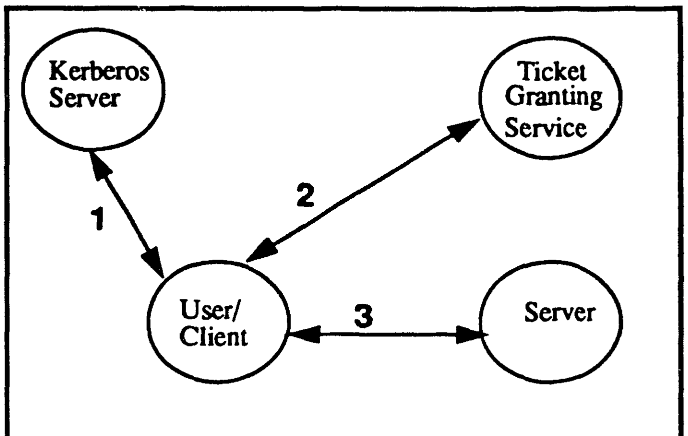

Figure 1. Kerberos Authentication Phases

Figure 2 shows the major components of the Project Athena Kerberized NFS design. The centerpiece of the design is the addition of a system call, *nfs_mapctl*, which allows users on client systems to set up credentials on the server for use in subsequent NFS requests. These credentials are cached in a map entry table which is managed by the *nfs_mapctl* system call code. Users authenticate themselves via the Kerberos authentication mechanism with the *nfsid* application program, which makes RPC calls to the server's mount daemon. After a user on a client system has created a map entry, any NFS request from that user on that client will utilize the credentials that were cached in kernel memory. If a user has not set up a map entry, he may be refused access or granted access of the "nobody" user.

The credentials that are cached on the server are not those that the user on the client provides. Rather, the Kerberized NFS server decrypts the Kerberos ticket and retrieves the user's credentials from a database of authorized users. This database can be built using the *mkcred* utility on the server. (At M.I.T., the Moira service is used to create this database.) If a user is not in the database, the credential will not be created.

In order to manage the cache of credentials on the server, the *nfsc* utility is provided. Using this utility, the system administrator on the server may retrieve, set, remove, dump, and load the server's map table.

When a user no longer requires access to the NFS server, cached credentials may be removed by running the *nfsid* utility with the unmap or purge option. At M.I.T. this is done automatically when a user logs out. In 1989, Sandia decided to evaluate the M.I.T. Kerberized NFS design for use in Sandia's secure network environment. SNL,A obtained version 1.2 of this code for use in the evaluation.

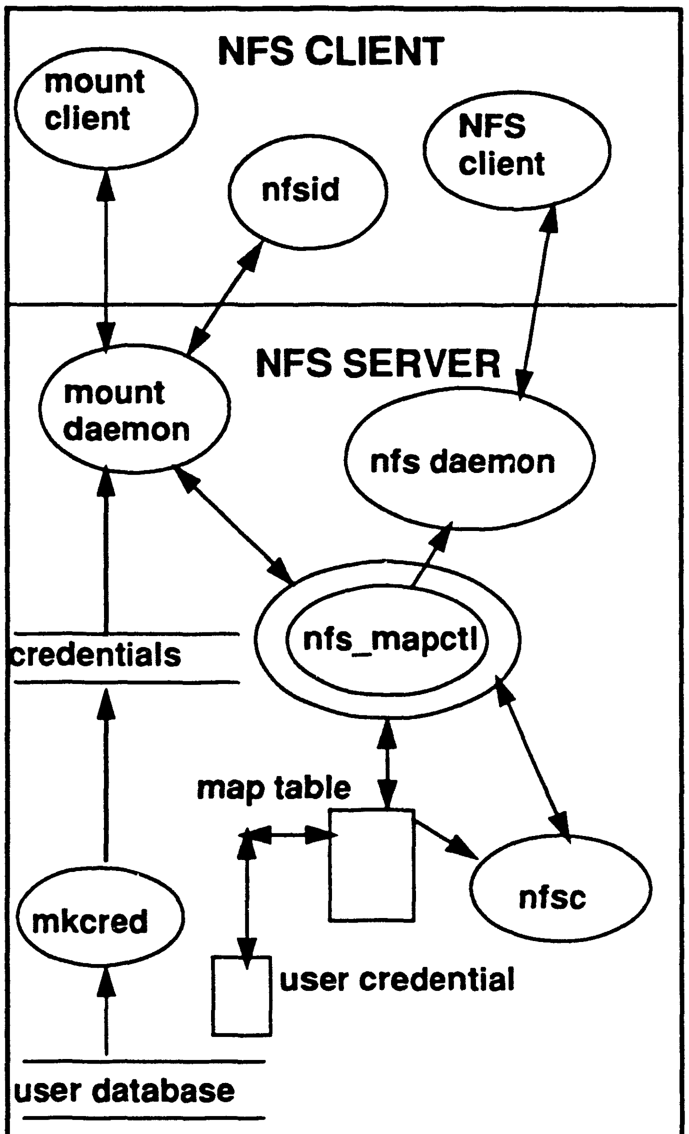

Figure 2. Kerberized NFS Components

3.2 Evaluation of the Project Athena NFS Design

Sandia ported the M.I.T. Kerberos NFS software to a Sun 3/80 workstation running SunOS 4.0.3 for use in the evaluation. The client application (*nfsid*) was ported to a variety of other workstation. Once these codes were ported, Kerberized NFS was tested using Kerberos Version 4. In testing the M.I.T. Kerberized NFS, Sandia identified several areas that would require modification before it could be used in Sandia's secure network environment. These areas are discussed in the following paragraphs.

The M.I.T. design allowed users on client systems to remove from the server's map table the credentials for all users on the client. While this option is valid for networks where the clients are not really multi-user systems, this was not valid for Sandia. Similarly, the M.I.T. design allowed users to remove their credentials without the Kerberos authentication. In Sandia's environment, where several users may be on a client system, positive user authentication is required.

The MIT definitions of the procedures to set up and remove user credentials give *void* as the data returned from the remote procedure call. Sandia determined in its testing that this could result in a user reusing another user's credential by merely calling the remote procedure with a *void* input parameter.

A major deficiency in the M.I.T. design was the requirement that users explicitly remove their credentials from the server. At M.I.T., this was overcome by tieing the delete credential function to a user logout. Obviously, if a system crashed, the user would be unable to logout. Sandia would need a way to remove inactive user credentials.

Lastly, when a user obtains a Kerberos ticket for a network service, the user's client hostname is encoded in the ticket. When this ticket is presented to the server, the server can check the client's internet address with the hostname in the ticket to verify the authenticity of the client. While this is done in the initial NFS authentication, subsequent NFS access requests rely on the client address being correct. In the SNL,A environment, network node authentication is provided by the network architecture. Consequently, this weakness was not considered significant.

The evaluation of the M.I.T. implementation of Kerberized NFS indicated that although weaknesses existed, the basic design could be used in the Sandia environment to provide NFS access. Sandia made modifications to overcome these weaknesses and implemented Kerberized NFS on the Y-MP. This software was placed into production in February, 1991.

4.0 Implementing Kerberized NFS on UNICOS

Implementing Kerberized NFS on the UNICOS platform proved to be an interesting yet sometimes frustrating experience. The basic M.I.T. design of implementing a new system call and adding procedures to the mount daemon were quite straight forward. The problems of kernel memory management and the gotchas of a word oriented machine proved to be the largest hurdles.

In the release software from M.I.T., the system call code dynamically allocated memory whenever a user set up a Kerberos NFS map entry. In fact, several small allocations were made for every new entry. Sandia added fields to the entry include security and entry activity information. The data structure for a map entry is shown in Figure 3. The MIT release software acquires memory for the map entry, the credential, the machine name, and the groups list.

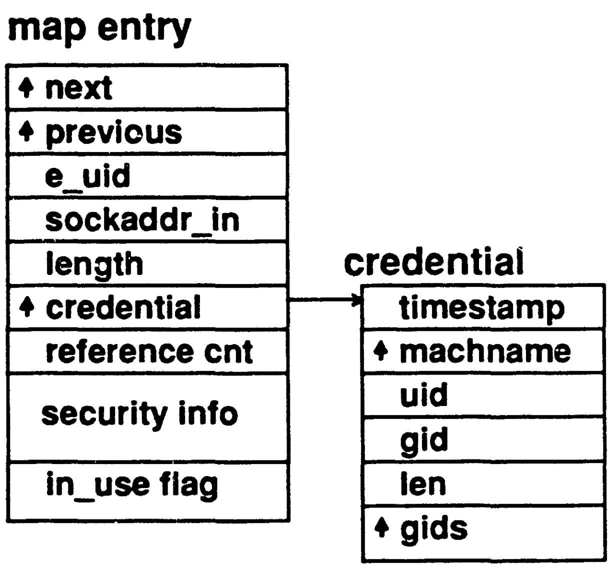

Figure 3. NFS Kerberos Map Entry

UNICOS provides no general kernel memory allocator. Networking software makes use of memory buffers that are allocated at system boot time for temporary (and some not so temporary) data structures. Every time one of these buffers is allocated a block of 1024 bytes is reserved. Sandia modified the system call code to use these buffers to cache user map entries. To conserve space, the code was modified to acquire memory only for the map entry and the credential. Additional memory in the credential was used to store the machine name and the groups list.

Problems associated with the fact that UNICOS is a word oriented system were more subtle. The first problem encountered was the definition of the Kerberos authentication ticket as an opaque XDR data structure. UNICOS could not determine the integer size field of the ticket. To overcome this, the Kerberos authentication ticket was defined according to the Kerberos header file. Other exciting adventures were experienced when zeroing data structures, copying bytes, and referencing the internet addresses.

To address the security weaknesses previously described, Sandia modified the *nfsid* protocol to remove the "purge host" option and to require a Kerberos authenticator with every *nfsid* request. The server mount daemon software was modified to clear the XDR data buffer before returning to the caller, and logic was added to set an in-use flag in a map entry whenever the entry was retrieved by NFS. Finally, a time-out function was added to the *nfsc* utility to detect and remove inactive user map entries.

One area where UNICOS provided an excellent facility for Kerberized NFS authentication was the user database (UDB). This allowed the removal of the *mkcred* utility and the associated system administration headaches. Authorized users are those that have logon accounts on UNICOS. Since Sandia uses a centrally administered global user name space, all the information needed for the user credentials could be easily obtained from the UDB.

5.0 Resource and Performance Issues

5.1 UNICOS Resource Utilization

Adding security functionality to a computing system requires utilization of system resources. Kerberized NFS authentication is no exception. One module was added to the UNICOS kernel and two modules were modified to support the Kerberos authentication. The UNICOS mount daemon was modified to add the Kerberos remote procedures and the *nfsc* utility was added.

Perhaps a more significant resource requirement is the memory storage for the user map entry credential cache. The alternative of forcing the client systems to provide Kerberos authentication would require significant modification of client system kernels and additional computation on both client and server. Sandia feels the benefits in terms of connectivity and functionality are worth the costs of the required resources.

5.2 UNICOS Performance

To measure the performance of the Kerberized NFS implementation at Sandia, the nhfsstone© benchmark was executed with Kerberos authentication enabled and disabled. Nhfsstone was developed by Legato Systems, Inc., to generate an artificial load with a particular mix of NFS operations. If a mix is not given, the default of 5000 calls with the following mix is used:

null 0%, getattr 13%, setattr 1%, root 0%, lookup 34%
readlink 8%, read 22%, wrcache 0%, write 15%,
create 2%, remove 1%, rename 0%, link 0%,
symlink 0%, mkdir 0%, rmdir 0%, readdir 3%, fsstat 1%

Since this mix represents a reasonable simulation of NFS activity on the Y-MP, the default mix was used. When nhfsstone completes, it reports the average response time of the server in milliseconds per call and the load in calls per second.

The client system for this test was a CONVEX C220i which was directly connected to a dedicated test-bed FDDI ring. This client system was selected because of the controlled environment on the client and the test-bed FDDI ring. Four *biod* processes were started on the client for buffering I/O requests. The Y-MP8/864, running UNICOS 6.1, was configured with eight *nfsd* processes.

Results are given in Figure 4. As can be seen, the results indicate a negligible difference in the time to process NFS requests when Kerberos authentication is enabled and disabled.

6.0 Conclusions

The NFS service can be provided in a secure supercomputer environment. User authentication can be effected using Kerberos authentication. The cost of authentication is primarily memory utilization with the benefit being the efficient NFS service to a wide variety of client systems.

References

1. R. Sandberg, D. Goldberg, S. Kleiman, D. Walch, and B. Lyon, "Design and Implementation of the Sun Network Filesystem" in Usenix Conference Proceedings, (Summer, 1985).

2. S. Kelly. "Requirements for a Network Storage Service in a Supercomputer Environment", in Proceedings of the 28th Semi-Annual Cray User Group Meeting, (Fall, 1991).

3. G. Treese. "Berkeley UNIX on 1000 Workstations: Athena Changes to 4.3BSD" in Usenix Conference Proceedings, (Winter, 1988).

4. J. Steiner, C. Neuman, J. Schiller, "Kerberos: An Authentication Service for Open Network Systems" in Usenix Conference Proceedings, (Winter, 1988).

NFS Operation		**Kerberos Authentication**	
		Enabled msec/call	**Disabled msec/call**
null	0	0.0	0.0
getattr	1112	1.45	2.22
setattr	117	37.62	23.4
root	0	0.0	0.0
lookup	1850	54.08	75.85
readlink	450	68.84	104.39
read	254	836.20	1105.05
wrcache	0	0.0	0.0
write	857	272.88	162.9
create	107	280.46	191.15
remove	51	229.58	139.38
rename	0	0.0	0.0
link	0	0.0	0.0
symlink	0	0.0	0.0
mkdir	0	0.0	0.0
rmdir	0	0.0	0.0
readdir	150	59.62	89.47
fsstat	52	65.98	85.44
Average Load	5000	127.48	119.8

Figure 4. Nhfsstone Results

Trademarks

UNIX is a registered trademark of AT&T.
NFS is a trademark of Sun Microsystems, Inc.
Kerberos is a trademark of M.I.T.
Nhfsstone is a copyrighted product of Legato Systems, Inc.
UNICOS is a registered trademark of Cray Research, Inc.

Acknowledgement

This work was supported by the United States Department of Energy under contract DE-AC04-76DP00789.

END

DATE
FILMED

10/30/91