Network Working Group
Request for Comments: 2623
Category: Standards Track

M. Eisler
Sun Microsystems, Inc.

June 1999

NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5

Status of this Memo

Copyright Notice

Abstract

This memorandum clarifies various security issues involving the NFS protocol (Version 2 and Version 3 only) and then describes how the Version 2 and Version 3 of the NFS protocol use the RPCSEC_GSS security flavor protocol and Kerberos V5. This memorandum is provided so that people can write compatible implementations.

Table of Contents

1. Introduction

The NFS protocol provides transparent remote access to shared file systems across networks. The NFS protocol is designed to be machine, operating system, network architecture, and security mechanism, and transport protocol independent. This independence is achieved through the use of ONC Remote Procedure Call (RPC) primitives built on top of an eXternal Data Representation (XDR). NFS protocol Version 2 is specified in the Network File System Protocol Specification [RFC1094]. A description of the initial implementation can be found in [Sandberg]. NFS protocol Version 3 is specified in the NFS Version 3 Protocol Specification [RFC1813]. A description of some initial implementations can be found in [Pawlowski].

For the remainder of this document, whenever it refers to the NFS protocol, it means NFS Version 2 and Version 3, unless otherwise stated.

The RPC protocol is specified in the Remote Procedure Call Protocol Specification Version 2 [RFC1831]. The XDR protocol is specified in External Data Representation Standard [RFC1832].

A new RPC security flavor, RPCSEC_GSS, has been specified [RFC2203]. This new flavor allows application protocols built on top of RPC to access security mechanisms that adhere to the GSS-API specification

[RFC2078].

The purpose of this document is to clarify NFS security issues and to specify how the NFS protocol uses RPCSEC_GSS. This document will also describe how NFS works over Kerberos V5, via RPCSEC_GSS.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.1. Overview of RPC Security Architecture

The RPC protocol includes a slot for security parameters (referred to as an authentication flavor in the RPC specification [RFC1831]) on every call. The contents of the security parameters are determined by the type of authentication used by the server and client. A server may support several different flavors of authentication at once. Some of the better known flavors are summarized as follows:

* The AUTH_NONE flavor provides null authentication, that is, no authentication information is passed.

* The AUTH_SYS flavor provides a UNIX-style user identifier, group identifier, and an array of supplemental group identifiers with each call.

* The AUTH_DH (sometimes referred to as AUTH_DES [RFC1057]) flavor provides DES-encrypted authentication parameters based on a network-wide string name, with session keys exchanged via the Diffie-Hellman public key scheme.

* The AUTH_KERB4 flavor provides DES encrypted authentication parameters based on a network-wide string name (the name is a Kerberos Version 4 principal identifier) with session keys exchanged via Kerberos Version 4 secret keys.

The NFS protocol is not limited to the above list of security flavors.

2. Overview of NFS Security

2.1. Port Monitoring

Many NFS servers will require that the client send its NFS requests from UDP or TCP source ports with values < 1024. The theory is that binding to ports < 1024 is a privileged operation on the client, and so the client is enforcing file access permissions on its end. The theory breaks down because:

* On many operating systems, there are no constraints on what port what user can bind to.

* Just because the client host enforces the privilege on binding to ports < 1024 does not necessarily mean that a non-privileged user cannot gain access to the port binding privilege. For example with a single-user desk-top host running a UNIX operating system, the user may have knowledge of the root user password. And even if he does not have that knowledge, with physical access to the desk-top machine, root privileges are trivially acquired.

In some rare cases, when the system administrator can be certain that the clients are trusted and under control (in particular, protected from physical attack), relying of trusted ports MAY be a reliable form of security.

In most cases, the use of privileged ports and port monitoring for security is at best an inconvenience to the attacker and SHOULD NOT be depended on.

To maximize interoperability:

* NFS clients SHOULD attempt to bind to ports < 1024. In some cases, if they fail to bind (because either the user does not have the privilege to do so, or there is no free port < 1024), the NFS client MAY wish to attempt the NFS operation over a port >= 1024.

* NFS servers that implement port monitoring SHOULD provide a method to turn it off.

* Whether port monitoring is enabled or not, NFS servers SHOULD NOT reject NFS requests to the NULL procedure (procedure number 0). See subsection 2.3.1, "NULL procedure" for a complete explanation.

2.1.1. MOUNT Protocol

The port monitoring issues and recommendations apply to the MOUNT protocol as well.

2.2. RPC Security Flavors

The NFS server checks permissions by taking the credentials from the RPC security information in each remote request. Each flavor packages credentials differently.

2.2.1. AUTH_SYS

Using the AUTH_SYS flavor of authentication, the server gets the client's effective user identifier, effective group identifier and supplemental group identifiers on each call, and uses them to check access. Using user identifiers and group identifiers implies that the client and server either share the same identifier name space or do local user and group identifier mapping.

For those sites that do not implement a consistent user identifier and group identifier space, NFS implementations must agree on the mapping of user and group identifiers between NFS clients and servers.

2.2.2. AUTH_DH and AUTH_KERB4

The AUTH_DH and AUTH_KERB4 styles of security are based on a network-wide name. They provide greater security through the use of DES encryption and public keys in the case of AUTH_DH, and DES encryption and Kerberos secret keys (and tickets) in the AUTH_KERB4 case. Again, the server and client must agree on the identity of a particular name on the network, but the name to identity mapping is more operating system independent than the user identifier and group identifier mapping in AUTH_SYS. Also, because the authentication parameters are encrypted, a malicious user must know another user's network password or private key to masquerade as that user. Similarly, the server returns a verifier that is also encrypted so that masquerading as a server requires knowing a network password.

2.2.3. RPCSEC_GSS

The RPCSEC_GSS style of security is based on a security-mechanism-specific principal name. GSS-API mechanisms provide security through the use of cryptography. The cryptographic protections are used in the construction of the credential on calls, and in the verifiers on replies. Optionally, cryptographic protections will be in the body of the calls and replies.

Note that the discussion of AUTH_NONE, AUTH_SYS, AUTH_DH, AUTH_KERB4, and RPCSEC_GSS does not imply that the NFS protocol is limited to using those five flavors.

2.3. Authentication for NFS Procedures

2.3.1. NULL Procedure

The NULL procedure is typically used by NFS clients to determine if an NFS server is operating and responding to requests (in other words, to "ping" the NFS server). Some NFS servers require that a client using the NULL procedure:

* send the request from TCP or UDP port < 1024. There does not seem to be any value in this because the NULL procedure is of very low overhead and certainly no more overhead than the cost of processing a NULL procedure and returning an authentication error. Moreover, by sending back an authentication error, the server has confirmed the information that the client was interested in: is the server operating?

* be authenticated with a flavor stronger than AUTH_SYS. This is a problem because the RPCSEC_GSS protocol uses NULL for control messages.

NFS servers SHOULD:

* accept the NULL procedure ping over AUTH_NONE and AUTH_SYS, in addition to other RPC security flavors, and

* NOT require that the source port be < 1024 on a NULL procedure ping.

2.3.2. NFS Procedures Used at Mount Time

Certain NFS procedures are used at the time the NFS client mounts a file system from the server. Some NFS server implementations will not require authentication for these NFS procedures. For NFS protocol Version 2, these procedures are GETATTR and STATFS. For Version 3, the procedure is FSINFO.

The reason for not requiring authentication is described as follows. When the NFS client mounts a NFS server’s file system, the identity of the caller on the client is typically an administrative entity (in UNIX operating systems, this is usually the "root" user). It is often the case that, for unattended operation in concert with an automounter [Callaghan], the AUTH_DH, AUTH_KERB4, or RPCSEC_GSS credentials for the administrative entity associated with an automounter are not available. If so, the NFS client will use AUTH_NONE or AUTH_SYS for the initial NFS operations used to mount a file system. While an attacker could exploit this implementation artifact, the exposure is limited to gaining the attributes of a file

or a file system's characteristics. This OPTIONAL trade off favors the opportunity for improved ease of use.

2.4. Binding Security Flavors to Exports

NFS servers MAY export file systems with specific security flavors bound to the export. In the event a client uses a security flavor that is not the one of the flavors the file system was exported with, NFS server implementations MAY:

* reject the request with an error (either an NFS error or an RPC level authentication error), or

* allow the request, but map the user's credentials to a user other than the one the client intended. Typically the user that is the result of this mapping is a user with limited access on the system, such as user "nobody" on UNIX systems.

If a client uses AUTH_NONE, the server's options are the same as the above, except that AUTH_NONE carries with it no user identity. In order to allow the request, on many operating systems the server will assign a user identity. Typically this assignment will be a user with limited access on the system, such as user "nobody" on UNIX systems.

2.5. Anonymous Mapping

The following passage is excerpted verbatim from RFC 1813, section 4.4 "Permission Issues" (except that "may" has been changed to "MAY"):

> In most operating systems, a particular user (on UNIX, the uid 0) has access to all files, no matter what permission and ownership they have. This superuser permission MAY not be allowed on the server, since anyone who can become superuser on their client could gain access to all remote files. A UNIX server by default maps uid 0 to a distinguished value (UID_NOBODY), as well as mapping the groups list, before doing its access checking. A server implementation MAY provide a mechanism to change this mapping. This works except for NFS version 3 protocol root file systems (required for diskless NFS version 3 protocol client support), where superuser access cannot be avoided. Export options are used, on the server, to restrict the set of clients allowed superuser access.

The issues identified as applying to NFS protocol Version 3 in the above passage also apply to Version 2.

2.6. Host-based Access Control

In some NFS server implementations, a host-based access control method is used whereby file systems can be exported to lists of clients. File systems may also be exported for read-only or read-write access. Several of these implementations will check access only at mount time, during the request for the file handle via the MOUNT protocol handshake. The lack of authorization checking during subsequent NFS requests has the following consequences:

* NFS servers are not able to repudiate access to the file system by an NFS client after the client has mounted the file system.

* An attacker can circumvent the MOUNT server's access control to gain access to a file system that the attacker is not authorized for. The circumvention is accomplished by either stealing a file handle (usually by snooping the network traffic between an legitimate client and server) or guessing a file handle. For this attack to succeed, the attacker must still be able impersonate a user's credentials, which is simple for AUTH_SYS, but harder for AUTH_DH, AUTH_KERB4, and RPCSEC_GSS.

* WebNFS clients that use the public file handle lookup [RFC2054] will not go through the MOUNT protocol to acquire initial file handle of the NFS file system. Enforcing access control via the MOUNT protocol is going to be a little use. Granted, some WebNFS server implementations cope with this by limiting the use of the public file handle to file systems exported to every client on the Internet.

Thus, NFS server implementations SHOULD check the client's authorization on each NFS request.

2.7. Security Flavor Negotiation

Any application protocol that supports multiple styles of security will have the issue of negotiating the security method to be used. NFS Version 2 had no support for security flavor negotiation. It was up to the client to guess, or depend on prior knowledge. Often the prior knowledge would be available in the form of security options specified in a directory service used for the purpose of automounting.

The MOUNT Version 3 protocol, associated with NFS Version 3, solves the problem by having the response to the MNT procedure include a list of flavors in the MNT procedure. Note that because some NFS servers will export file systems to specific lists of clients, with different access (read-only versus read-write), and with different

security flavors, it is possible a client might get back multiple security flavors in the list returned in the MNT response. The use of one flavor instead of another might imply read-only instead of read-write access, or perhaps some other degradation of access. For this reason, a NFS client SHOULD use the first flavor in the list that it supports, on the assumption that the best access is provided by the first flavor. NFS servers that support the ability to export file systems with multiple security flavors SHOULD either present the best accessing flavor first to the client, or leave the order under the control of the system administrator.

2.8. Registering Flavors

When one develops a new RPC security flavor, iana@iana.org MUST be contacted to get a unique flavor assignment. To simplify NFS client and server administration, having a simple ASCII string name for the flavor is useful. Currently, the following assignments exist:

flavor	string name
AUTH_NONE	none
AUTH_SYS	sys
AUTH_DH	dh
AUTH_KERB4	krb4

A string name for a new flavor SHOULD be assigned. String name assignments can be registered by contacting iana@iana.org.

3. The NFS Protocol's Use of RPCSEC_GSS

3.1. Server Principal

When using RPCSEC_GSS, the NFS server MUST identify itself in GSS-API via a GSS_C_NT_HOSTBASED_SERVICE name type. GSS_C_NT_HOSTBASED_SERVICE names are of the form:

```
service@hostname
```

For NFS, the "service" element is

```
nfs
```

3.2. Negotiation

RPCSEC_GSS is a single security flavor over which different security mechanisms can be multiplexed. Within a mechanism, GSS-API provides for the support of multiple quality of protections (QOPs), which are pairs of cryptographic algorithms. Each algorithm in the QOP consists

of an encryption algorithm for privacy and a checksum algorithm for integrity. RPCSEC_GSS lets one protect the RPC request/response pair with plain header authentication, message integrity, and message privacy. Thus RPCSEC_GSS effectively supports M * Q * 3 different styles of security, where M is the number of mechanisms supported, Q is the average number of QOPs supported for each mechanism, and 3 enumerates authentication, integrity, and privacy.

Because RPCSEC_GSS encodes many styles of security, just adding RPCSEC_GSS to the list of flavors returned in MOUNT Version 3's MNT response is not going to be of much use to the NFS client.

The solution is the creation of a concept called "pseudo flavors." Pseudo flavors are 32 bit integers that are allocated out of the same number space as regular RPC security flavors like AUTH_NONE, AUTH_SYS, AUTH_DH, AUTH_KERB4, and RPCSEC_GSS. The idea is that each pseudo flavor will map to a specific triple of security mechanism, quality of protection, and service. The service will be one of authentication, integrity, and privacy. Note that integrity includes authentication, and privacy includes integrity. RPCSEC_GSS uses constants named rpc_gss_svc_none, rpc_gss_svc_integrity, and rpc_gss_svc_privacy, for authentication, integrity, and privacy respectively.

Thus, instead of returning RPCSEC_GSS, a MOUNT Version 3 server will instead return one or more pseudo flavors if the NFS server supports RPCSEC_GSS and if the file system has been exported with one or more <mechanism, QOP, service> triples. See section 4, "The NFS Protocol over Kerberos V5" for an example of pseudo flavor to triple mapping.

3.3. Changing RPCSEC_GSS Parameters

Once an RPCSEC_GSS session or context has been set up (via the RPCSEC_GSS_INIT and RPCSEC_GSS_CONTINUE_INIT control procedures of RPCSEC_GSS), the NFS server MAY lock the <mechanism, QOP, service> triple for the duration of the session. While RPCSEC_GSS allows for the use of different QOPs and services on each message, it would be expensive for the NFS server to re-consult its table of exported file systems to see if the triple was allowed. Moreover, by the time the NFS server's dispatch routine was reached, the typical RPC subsystem would already have performed the appropriate GSS-API operation, GSS_VerifyMIC() or GSS_Unwrap(), if the respective integrity or privacy services were selected. If the file system being accessed were not exported with integrity or privacy, or with the particular QOP used to perform the integrity or privacy service, then it would be possible to execute a denial of service attack, whereby the objective of the caller is to deny CPU service to legitimate users of the NFS server's machine processors.

Thus, in general, clients SHOULD NOT assume that they will be permitted to alter the <mechanism, QOP, service> triple once the data exchange phase of RPCSEC_GSS has started.

3.4. Registering Pseudo Flavors and Mappings

Pseudo flavor numbers MUST be registered via same method as regular RPC security flavor numbers via iana@iana.org.

Once the pseudo flavor number has been assigned, registrants SHOULD register the mapping with iana@iana.org. The mapping registration MUST contain:

* the pseudo flavor number, an ASCII string name for the flavor (for example "none" has been assigned for AUTH_NONE), and

* the <mechanism, algorithm(s), service> triple. As per the GSS-API specification, the mechanism MUST be identified with a unique ISO object identifier (OID). The reason why the second component of the triple is not necessarily a QOP value is that GSS-API allows mechanisms much latitude in the mapping of the algorithm used in the default quality of protection (See subsection 4.1, "Issues with Kerberos V5 QOPs," for a detailed discussion). With some mechanisms, the second component of the triple will be a QOP. Internally, on the NFS implementation, it is expected that the triple would use a QOP for the second component.

The mapping registration SHOULD also contain:

* A reference to an RFC describing how the NFS protocol works over the pseudo flavor(s), including the pseudo flavor number(s), string name(s) for the flavor(s), and any other issues, including how the registrant is interpreting the GSS-API mechanism.

* A reference to the GSS-API mechanism used.

An example of a complete registration is provided in subsection 4.2, "The NFS Protocol over Kerberos V5 Pseudo Flavor Registration Entry."

4. The NFS Protocol over Kerberos V5

The NFS protocol uses Kerberos V5 security using the RPCSEC_GSS security flavor. The GSS-API security mechanism for Kerberos V5 that the NFS/RPCSEC_GSS protocol stack uses is described in the Kerberos V5 GSS-API description [RFC1964].

4.1. Issues with Kerberos V5 QOPs

The Kerberos V5 GSS-API description defines three algorithms for integrity:

* DES MAC MD5

* MD2.5

* DES-MAC

RFC 1964 states that MD2.5 "may be significantly weaker than DES MAC MD5." RFC 1964 also states that DES-MAC "may not be present in all implementations."

Thus the description of operation of NFS clients and servers over Kerberos V5 is limited to the DES MAC MD5 integrity algorithm.

NFS clients and servers operating over Kerberos V5 MUST support the DES MAC MD5 integrity algorithm. RFC 1964 lists a single algorithm for privacy: 56 bit DES. NFS clients and servers SHOULD support the 56 bit DES privacy algorithm.

GSS-API has the concept of a default QOP of zero which means different integrity and privacy algorithms to different GSS-API mechanisms. In Kerberos V5, the default QOP of zero means to use the 56 bit DES algorithm (when doing a GSS_Wrap() operation with the conf_req_flag set to 1).

For Kerberos V5, the default QOP of zero means different integrity algorithms to different implementations of Kerberos V5. Furthermore, during the processing of a token in GSS_Unwrap(), and GSS_VerifyMIC(), at least one reference implementation of the Kerberos V5 GSS-API mechanism [MIT], always returns a QOP of zero, regardless of integrity algorithm encoded in the token. For such implementations, it means that the caller of GSS_Unwrap() and GSS_VerifyMIC() cannot know the actual integrity algorithm used. Given that each integrity algorithm has a different degree of security, this situation may not be acceptable to the user of GSS-API. An implementation of Kerberos V5 under GSS-API for use under NFS MUST NOT do this.

For the purposes of NFS, as a simplification, some Kerberos V5 GSS-API mechanisms MAY map QOP 0 to always mean DES MAC MD5 integrity, and when using GSS_VerifyMIC() and GSS_Unwrap(), always map the DES MAC MD5 integrity that is specified to QOP 0.

4.2. The NFS Protocol over Kerberos V5 Pseudo Flavor Registration Entry

Here are the pseudo flavor mappings for the NFS protocol using

Kerberos V5 security:

columns:

1 == number of pseudo flavor
2 == name of pseudo flavor
3 == mechanism's OID
4 == mechanism's algorithm(s)
5 == RPCSEC_GSS service

1	2	3	4	5
390003	krb5	1.2.840.113554.1.2.2	DES MAC MD5	rpc_gss_svc_none
390004	krb5i	1.2.840.113554.1.2.2	DES MAC MD5	rpc_gss_svc_integrity
390005	krb5p	1.2.840.113554.1.2.2	DES MAC MD5 for integrity, and 56 bit DES for privacy.	rpc_gss_svc_privacy

An implementation of NFS over RPCSEC_GSS/GSS-API/Kerberos V5 that maps the default QOP to DES MAC MD5 (and vice versa), would implement a mapping of:

columns:

1 == number of pseudo flavor
2 == name of pseudo flavor
3 == mechanism's OID
4 == QOP
5 == RPCSEC_GSS service

1	2	3	4	5
390003	krb5	1.2.840.113554.1.2.2	0	rpc_gss_svc_none
390004	krb5i	1.2.840.113554.1.2.2	0	rpc_gss_svc_integrity
390005	krb5p	1.2.840.113554.1.2.2	0	rpc_gss_svc_privacy

The reference for the GSS-API mechanism with the above OID is [RFC1964].

The reference for how the NFS protocol MUST work over Kerberos V5 is this document.

5. Security Considerations

Version 3 of the MOUNT protocol is used to negotiate the security flavor to be used by the NFS Version 3 client. If the NFS client uses a weak security flavor like AUTH_SYS to query a Version 3 MOUNT server, then the following attacks are possible by an attacker in the middle:

* The attacker in the middle can coax the NFS client into using a weaker form of security than what the real NFS server requires. However, once the NFS client selects a security flavor when it sends a request to real NFS server, if the flavor is unacceptable, the NFS client's NFS request will be rejected. So at worst, a denial of service attack is possible. In theory, the NFS client could contact the MOUNT server using a stronger security flavor, but this would require that the client know in advance what security flavors the MOUNT server supports.

* If the client and server support a common set of security flavors, such that the client considers one preferable to the other (for example, one might have privacy and other not), unless the client uses a strong security flavor in the MOUNT protocol query, an attacker in the middle could cause the client to use the weaker form of security. Again, a client could contact the MOUNT server using a stronger form of security.

6. IANA Considerations [RFC2434]

This memorandum describes how NFS Version 2 and Version 3 work over RPC's RPCSEC_GSS security flavor. This memorandum requires that triples of { GSS-API mechanism OID, GSS-API mechanism algorithm, RPCSEC_GSS security service } be mapped to a unique RPC security flavor number, which is a pseudo flavor that does not appear in an RPC protocol header. This memorandum also encourages that an ASCII string name be registered with the triple.

Thus there are five different kinds of objects to consider guidelines for.

6.1. Pseudo Flavor Number

The considerations of assignment, allocation, and delegation of pseudo flavor numbers are no different than that the considerations for RPC security flavors, as both are assigned from the same number space. IANA is already responsible for the assigned of RPC security flavors, and because this memorandum does not specify the RPC protocol [RFC1831], it is beyond the scope of this memorandum to guide IANA in the assignment of flavor numbers.

6.2. String Name of Pseudo Flavor

This memorandum introduces the concept of a string name to be associated with the RPC pseudo flavor number, and so it is within the scope of this memorandum to provide guidance to IANA.

6.2.1. Name Space Size

There are no limits placed on the length of the unique string name by this memorandum, so the size of the name space is infinite. However, IANA may want to prevent the hoarding or reservation of names. The simplest way to do this is by requiring the registrant to provide the GSS-API mechanism OID, GSS-API quality of protection, the RPCSEC_GSS security service, and flavor number, with the request for a flavor name. If the registrant does not have a flavor number, then guidelines for flavor number assignments will indirectly limit the assignment of flavor names.

6.2.2. Delegation

The simplest way to handle delegation is to delegate portions of the RPC security flavor number space with the RPC flavor name space. The guidelines for delegation of the flavor name space are thus equivalent to guidelines for delegations of the flavor number space.

6.2.3. Outside Review

Because string names can be trademarks, IANA may want to seek legal counsel to review a proposed pseudo flavor name. Other than that, no outside review is necessary.

6.3. GSS-API Mechanism OID

This memorandum assumes that the mechanism OID associated with the pseudo flavor has already been allocated. OIDs are allocated by the International Standards Organization and the International Telecommunication Union. Both organizations have delegated assignment authority for subsets of the OID number space to other organizations. Presumably, IANA has received authority to assign OIDs to GSS-API mechanisms. Because this memorandum does not specify the GSS-API protocol (see [RFC2078]) it is beyond the scope of this memorandum to guide IANA in the assignment of GSS-API mechanism OIDs.

6.4. GSS-API Mechanism Algorithm Values

This memorandum assumes that the algorithm value for a given GSS-API mechanism has already been allocated. Algorithm values are controlled by the owner of the GSS-API mechanism, though the owner may delegate

assignment of algorithm values to a body such as IANA. Because this memorandum does not specify GSS-API mechanisms, such as [RFC1964], it is beyond the scope of this memorandum to guide IANA in the assignment of a mechanism's algorithm value(s).

6.5. RPCSEC_GSS Security Service

There are only three security services and they are enumerated and described in [RFC2203]. No guideline to IANA is necessary.

References

[RFC1094] Sun Microsystems, Inc., "NFS: Network File System Protocol Specification", RFC 1094, March 1989. http://www.ietf.org/rfc/rfc1094.txt

[Sandberg] Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B. (1985). "Design and Implementation of the Sun Network Filesystem," Proceedings of the 1985 Summer USENIX Technical Conference.

[RFC1813] Callaghan, B., Pawlowski, B. and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, June 1995. http://www.ietf.org/rfc/rfc1813.txt

[RFC1831] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1831, August 1995. http://www.ietf.org/rfc/rfc1831.txt

[RFC1832] Srinivasan, R., "XDR: External Data Representation Standard", RFC 1832, August 1995. http://www.ietf.org/rfc/rfc1832.txt

[Pawlowski] Pawlowski, B., Juszczak, C., Staubach, P., Smith, C., Lebel, D. and D. Hitz, "NFS Version 3 Design and Implementation", Proceedings of the USENIX Summer 1994 Technical Conference.

[RFC2203] Eisler, M., Chiu, A. and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997. http://www.ietf.org/rfc/rfc2203.txt

[RFC2078] Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997. http://www.ietf.org/rfc/rfc2078.txt

[RFC1057] Sun Microsystems, Inc., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1057, June 1988. This RFC is being referenced for its description of the AUTH_DH (AUTH_DES) RPC security flavor. http://www.ietf.org/rfc/rfc1057.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt

[Callaghan] Callaghan, B., Singh, S. (1993). "The Autofs Automounter," Proceedings of the 1993 Summer USENIX Technical Conference.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996. http://www.ietf.org/rfc/rfc1964.txt

[RFC2054] Callaghan, B., "WebNFS Client Specification", RFC 2054, October 1996. http://www.ietf.org/rfc/rfc2054.txt

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998. http://www.ietf.org/rfc/rfc2434.txt

[MIT] Massachusetts Institute of Technology (1998). "Kerberos: The Network Authentication Protocol." The Web site for downloading MIT's implementation of Kerberos V5, including implementations of RFC 1510 and RFC 1964. http://web.mit.edu/kerberos/www/index.html

Acknowledgments

The author thanks:

* Brent Callaghan, John Hawkinson, Jack Kabat, Lin Ling, Steve Nahm, Joyce Reynolds, and David Robinson for their review comments.

* John Linn, for his explanation of QOP handling in RFC 1964.

Author's Address

Address comments related to this memorandum to:

nfsv4-wg@sunroof.eng.sun.com

Mike Eisler
Sun Microsystems, Inc.
5565 Wilson Road
Colorado Springs, CO 80919

Phone: 1-719-599-9026
EMail: mre@eng.sun.com

14. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.