Workgroup: NFSv4
Updates: 8881, 7530 (if approved)
Published: 21 April 2024
Intended Status: Standards Track
Expires: 23 October 2024
Authors: D. Noveck, Ed.
NetApp

Security for the NFSv4 Protocols

Abstract

This document describes the core security features of the NFSv4 family of protocols, applying to all minor versions. The discussion includes the use of security features provided by RPC on a per-connection basis. Important aspects of the authorization model, related to the ACL feature, will be specified in a separate document.

The current version of the document is intended, in large part, to result in working group discussion regarding existing NFSv4 security issues and to provide a framework for addressing these issues and obtaining working group consensus regarding necessary changes.

When the resulting documents (i.e. this document and one derived from the separate ACL specification) are eventually published as RFCs, they will, by updating these documents, supersede the description of security appearing in existing minor version specification documents such as RFC 7530 and RFC 8881,

Table of Contents

1. Overview

These documents, including this document and the companion ACL document [I-D.dnoveck-nfsv4-acls], are intended to form the basis for a new description of NFSv4 security applying to all NFSv4 minor versions. The motivation for these new documents and the need for major improvements in the description of NFSv4 security are explained in Section 1.1.

Because these documents anticipate making major changes in material covered in previous standards-track RFCs, extensive working group discussion will be necessary to make sure that there is a working group consensus to make the changes being proposed. The changes needed include the major improvements mentioned in Section 1.1.1 and the changes necessary to suitably describe features currently described in a way that is inappropriate in Standards Track

documents, for reasons laid out in Section 3.4. A large part of the material necessary to accomplish this set of goals will appear in [I-D.dnoveck-nfsv4-acls] and its successors.

The need to make major changes in the security approach for three Proposed Standards ([RFC7530] for NFSv4.0, [RFC8881] for NFSv4.1, and [RFC7862] for NFSv4.2) raises troubling issues. These changes are necessary for reasons explained in Section 1.1. These troubling issues often concern compatibility and compliance issues as described in Section 1.3.

1.1. Document Motivation

1.1.1. Original Motivation

A new treatment of security is necessary because:

* Previous treatments paid insufficient attention to security issues regarding data in flight, assuming that security could reasonably be provided on an optional basis, to secure particular portions of the server namespace.

* The presentation of AUTH_SYS as an "**OPTIONAL** means of authentication" obscured the significant security problems that come with its use.

* The security considerations sections of existing minor version specifications contain no threat analyses and focus on particular security issues in a way that obscures, rather than clarifying, the issues that need to be addressed, while implying, often incorrectly, that the existing security features are adequate to the need.

* The availability of connection-oriented RPC security features, such as those provided by RPC-with-TLS (described in [RFC9289]) provides facilities that NFSv4 clients and servers will need to use to provide security for data in flight and mitigate the lack of user authentication when AUTH_SYS is used.

1.1.2. Need for Respecifcation of Acl-related Funtionality

Review of the existing specifications has made it apparent that the handling of ACLs has not been described in the detail normally necessary to make it possible to implement interoperating clients and servers. Because of the broad license granted by previous specifications to allow server implementations to choose how to behave, clients are forced to accept a broad range of server behaviors, with no way of reliably determining the server behavior actually implemented.

One important reason that such extensive changes are now necessary derives from a disagreement among working group partucipants as to the purpose of ACL support with one group of participants requiring new functionallity matching that of Windows ACLs with another major group uninterested in such features and unwilling to devote the effort involved in providing server-side support for them.

Within the NFSv4 architecture, such situations are normally dealt by defining one or more optional features, allowing different servers to provide different levels of support, with the client able to determine whether a selected server has the desired level of support.

Unfortunately, for reasons that remain unclear, this approach was not followed in writing RFCs 3530, 5661, 7530, and 8881. Instead, the full definition of the ACL was defined as an OPTIONAL attribute with no clear definition of useful support subsets or way of testing the support level. Essentially, each ACE mask bit was made its own optional feature. To further complicate things, the keyword "**SHOULD**" was frequently used, raising the possibility that the support might be other than described. This made it impossible for clients to determine the level of ACL suppport provided, or to choose whether to use a server based on the level of support provided.

As a result, we now need to provide a core ACL model that all clients can rely upon while providing the ability for server to implement useful extensions. This work will done within the companion document devoted to ACLs [I-D.dnoveck-nfsv4-acls]. Given the need to thorougly revise the discussion of ACLs while avoiding prohibited XDR changes and troublesome implementation incompatibilities, the working group will need information about implementations of the vast range of possibilities allowed, inadvertantly or not, by previous treatments of the matter.

In view of these difficulties in the existing specification of acl-related semantics a new approach toward the specfication will be adopted in [I-D.dnoveck-nfsv4-acls] and its successors, although the goal, of allowing a range of potential ACL implementations, will remain the same, as will the XDR used to represent the relevant protocol elments. This XDR encompasses many **OPTIONAL** extensions to the UNIX ACLs whose core semantics need to be supported by all servers supporting ACLs.

1.1.3. Further Issues that Neeed to be Addressed

As work has proceeded, additional important issues were discovered. Of prime importantce are following issues relateed to the classification of attributes:

*The attributes Owner, Owner_group, and Mode needed to be made **REQUIRED** since clients need this information and not supporting these attributes would create troublesome interoperabiity issues.

Previous specifications explicitly allowed servers to support none of these and even discussed a supposed need to support use of servers that supported none of the above attributes and none of the acl-related attributes either. To continue in this way would ovecomplicate the specification and create difficult interoperability issues to support implementations that have little practical purpose.

*The change in the format of specification of user ids and group ids, made as part of transition from NFSv3 to NFSv4, requires significant modification, in order to adddress a serious underspecification that creates the possbility of addditional security vulnerablities.

The change to a string-format representaion of thse ids was intended to provide a way to allow the protocol to escape the restrictions inherent in the previous representaion of these ids by 32-bit unsigned integers. However, as things have developed, for practical puposes, these restrictions remain in effect since the associated ids within POSIX are still 32-bit unsigned integers and the working group has no way of prompting changes necessary to implememt a more flexible approach..

While the use of ids of the form "name@domain" might help achieve the original goal if used together with multiple domains, that would require server support to sumultenously support multiple Kerberos realms, not yet available.

In the case in which AUTH_SYS is used, there is no reliable way to effect the mapping betweeen names and numeric ids. In the existing specfications, the provision of this mapping is treated as an implementation matter without protocol support. This has proved unacceptable becasue any implementation advice would require compatible implementationss on the client and server, and would allow attacks via interferance with the mapping. Although use of numerc ids instead is possible, it has been unfairly stigmatized in previous specifications, with the suggestion made that use of numeric strings somehow compromises the orignal intent of the shift, which nevertheless, is in no way undercut by the use of

strings having numeric values as long as the "name@domain" format is still available when useful and necessary.

In the new treatment (in Section 5.1), the mapping between name and numeric ids is the responsibility of Kerberos and the use of numeric strings is available for the AUTH_SYS case, avoiding any need for mapping between names and numeric ids in this common case. In order to accommodte pevious implemntations, such mapping is allowed, although it creates some unfortunate security vulnerabilities that are best avoided.

In addition to these two major issues, the following other issues were found and needed to be addressed in this document:

* Although involved in the inadequate speecification of ACL handling, a separate source of issues that need to be addressed involves use of the term "**SHOULD**", sometimes meaning "should" and sometimes meaning **MAY**, but never including discussion of the harm caued by not follwing the recommendation, or a discussion of what might be valid reasons to ignore the recommendation.

* The discussion of handling of modes is limited to forward-slope modes and ignores the possibility of reverse-slope modes.

  For definitions of these terms, see Section 4.1.

  For examlles of the necessary changes, see Sections 8.3 and 8.7 of [I-D.dnoveck-nfsv4-acls]

* There are a number of claarifications/corrections to the description of ACE mask bits appearing in Section 5.2 of [I-D.dnoveck-nfsv4-acls] and its cnstituent sub-sections.

1.2. Document Annotation

In order to make progress on difficult issues which will require that changes be made in the existing handling of security issues, including many whose resolution would potentially involve compatibility issues with existing implementations, the author has tried his best to resolve these issues, even though there is no assurance that the resolution adopted by consensus will match the author's current best efforts. To provide possible resolutions that might be the basis of discussion while not foreclosing other possibilities, proposed changes are organized into a series of consensus items, which are listed in Appendix B and in a corresponding Appendix in the acl document.

For such pending issues, the following annotations will be used:

*A paragraph headed "[Author Aside]:", provides the author's comments about possible changes and will probably not appear in an eventual RFC.

This paragraph can specify that certain changes within the current section are to be implicitly considered as part of a specific consensus item.

The paragraph can indicate that all unannotated material in the current section is to be considered either the previous treatment or the proposed replacement text for a specific consensus item.

*A paragraph headed "[Consensus Needed (Item #NNx)]:", provides the author's preferred treatment of the matter and will only appear in the eventual RFC if working group consensus on the matter is obtained, allowing the necessary changes to be made permanent, without being conditional on a future consensus.

The item id, represented above by "NNx" consists of a number identifying the specific consensus item and letter which is unique to appearance of that consensus item in a particular section. In cases in which a pending item is cited with no part of the discussion appearing in the current section, an item id of the form "#NN" is used.

*A paragraph headed "[Previous Treatment]:", indicates text that is provided for context but which the author believes, need not appear in the eventual RFC, because it is expected to be superseded by a corresponding consensus item.

The corresponding consensus item is often easily inferred, but can be specified explicitly, as it is for items associated with the consensus item itself.

Each of the annotations above can be modified by addition of the phrase, "Including List" to indicate that it applies to a following bulleted list as well as the current paragraph or the phase "Entire Bulleted Item" to indicate it applies to all paragraphs within a specific bulleted item.

1.3. Compatibility and Compliance Issues

Changes that need to be adopted in this document might need to eventually change the behavior of clients and servers that were written to conform to earlier protocol specifications. There are two important classes of such changes discussed in Sections 1.3.1 and 1.3.2 below.

As [RFC2119] was originally conceived, compliance and compatibility issues were tightly bound together so that a change to compliance specifications would inevitably give rise to compatibility issues. However, over time, behaviors have come to be denigrated by use of the terms "**SHOULD NOT**" and "**MUST NOT**" to warn implementors of a harm deriving from from insecure operation rather than peer incompatibility.

When making changes in compliance requirements/recommendations we need to deal with the possibility that, in changing the specification, we might cause a previously compliant implementation to become non-compliant. Some implementers take the view that the compliance status of their implementations is of less importance than other considerations such as compatibility with local file system semantics. Others feel it is important to maintain the compliance status of existing implementations. In any case, the working group has been reluctant, when making such necessary changes, to make previously compliant implementations non-compliant, and will try not to do in cases in which it is known or can reasonably be expected that such implementations exist.

Although there is no way to be sure about the non-existence of such implementations, the working group has made judgments about what implementations are likely to exist. For example, when internationalization for NFSv4.0 was changed in the transition to [RFC7530], many previously non-compliant server implementations became officially compliant and there was a potential for conflict with implementations compliant with RFC3530 [RFC3530], if any such implementations existed.

In that particular case, it was decided that no such RFC3530-compliant server implementations existed and there was no need to accommodate servers whose internationalization was written to conform to [RFC3530] since no such servers existed and there were no client implementations expecting such behavior. As a result, no actual implementations became non-compliant as a result of this necessary shift.

In the corresponding cases dealt with in this document, the situation is more difficult since it may be harder to determine the actual behavior of all existing implementations, since the authors might no longer be actively involved with implementation issues. However, it will be necessary to warn implementors of the negative consequences of certain behaviors without going so far as to declare these practices non-compliant. For the most part, this document will use the terms "**SHOULD**" and "**SHOULD NOT**" to draw attention to the problems with troublesome behaviors that previous specifications mentioned with no indication of the problems with them. In such contexts, it is made clear that the need to maintain existing patterns of

interoperation is a valid reason to bypass the normative term. The intention is to continue to allow previously acceptable implementations to be considered compliant while not placing the troublesome behaviors on the same levels as other alternatives as would happen if we used the terms "**MAY**" or "**OPTIONAL**" or continued to use some of the unfortunate practices discussed in Section 1.3.2.

This approach allows implementations to accept input from peers written in accord with previous specification while not obligating them to do so. The intent is to allow a transition to newer, better behaviors over time as client and server policies evolve. However, the specifics of this anticipated transition will vary:

* In the case of the issues dealt with in Section 1.3.1, the focus is on making implementers aware of the security problems with practices previously considered acceptable.

  It is specified that these practices **SHOULD NOT** be used by clients or allowed by servers in order to draw attention to the security problems with them. At the same time, the discussion of valid reasons to bypass these recommendations allows them to continue to be used while the infrastructure is developed to make their replacements easy to use.

  Servers are encouraged to adopt policies foreclosing client use but not obligated to do so.

* In the case of the issues dealt with in Section 1.3.2, the focus is different. In these cases, while we anticipate making changes in compliance specifications, there is no need to address a specific set of troublesome practices. Instead, the problem to be addressed is the vast range of allowable server behaviors previously defined as allowed, although not necessarily explicitly. This server-centered approach has made compatibility a hit-or-miss matter, requiring serious consideration of the question of what specific instances of multiple server behaviors need to be allowed and how clients can find out the choices that servers have made and possibly affect them, also making appropriate provision, as with other optional features, about how to adapt to the server's behavioral choices or to decide that they do not meet its needs.

  It appears that this approach was motivated, at least in large part, by the desire to fully support much of Windows ACL semantics while accommodating Unix servers incapable of providing much of that functionality. As a result, the working group will need to provide a way for the server to explicitly opt out of providing Windows functionality that it cannot provide and that Linux clients are not prepared to use.

In addition, the working group will have to restrict, or at least better organize, sever behavioral choices related to the handling of ACLs.

1.3.1. Dealing with Recognized Mistakes

As an example, we consider the handling of AUTH_SYS (presumably in the clear, without client peer authentication), described in previous specifications as "**OPTIONAL**". While the authors might have only been indicating that servers could choose not to support it, and that clients had to be prepared for it not to be supported by the server, the likely import of this designation, for clients, was to indicate to implementers that they could choose to use AUTH_SYS and that the authors of the spec were, by not recommending otherwise, as we might now feel they should have done, indicating there there were no issues whereby using AUTH_SYS in this form had the capacity to cause harm. As a result, clients using AUTH_SYS in this way were to be considered specification- compliant and were not warned of the real security issues created by this use.

As the protocol was implemented and further developed in subsequent minor versions, the specifications, which had Security Considerations sections that did not contain threat analyses, had no place to indicate to users and implementers of NFSv4 the security problems that come with AUTH_SYS use and it continued to be used heavily while encryption was only available to clients using RPCSEC_GSS and left as a choice that was not frequently used, despite the security issues that this raised.

We are now at a point at which we have to recognize that a mistake was made in this regard and have to be clear about the security issues present in many common implementations of the protocol. As we seek to do this, it is important to understand the compliance effects of doing so. This document, when adopted, will supersede previous specifications which took a different approach. Although it might, given the security issues with AUTH_SYS, make sense to say that it "**MUST NOT**" be used in that way, the working group is very reluctant to retroactively declare previously compliant behavior non-compliant, even in this case where there is good reason to do so. A more likely approach is to say that clients "**SHOULD NOT**" do this while making it clear that the difficulty of changing existing implementations and potential compatibility with existing peers are valid reasons to bypass the recommendation. Servers are, as before, allowed to support AUTH_SYS but "**SHOULD**" only do so when using additional security facilities that make this safe. The effect would be to create a clear set of recommendations to new implementations while providing for continued use of previously compliant implementations to continue as needed,

This approach gives rise to compatibility issues, but leaves them to implementors and users to resolve, while making clear the security issues with the old approach.

1.3.2. Dealing with Pervasive Uncertainty

Addressing the issues described in Section 3.4 raises similar issues. In this case as well, we will need to make changes in implementation behavior going forward and try to do so without declaring existing behavior non-compliant. However, we cannot, as we did in the example above, identify a specific set of bad choices, and try to come up with replacements, that reflect our new understanding of security issues.

In this case, it is necessary, as has been done in other cases in which NFSv4 tried to accommodate the needs of both UNIX and Windows, to decide what part of the non-UNIX semantics is required and which part is an optional extension, which UNIX-oriented clients would not use and UNIX-oriented servers might not support. When this sort of issue is not given the attention it needs, problems can result, although the nature and severity of the problems depend on the specifics of feature.

In the case of byte-range locks, severs were given a choice as to implement byte-range locks in an advisory or mandatory fashion. Although servers could choose to do either, there was no way for a client to determine which of these two incompatible semantic models the server implemented. As a result, unix-based clients and applications assumed the advisory model and could not interoperate successfully with servers implementing the mandatory model. Applications requiring mandatory semantics could only interoperate with a small set of servers which chose to support the mandatory model that has very few users. The normal way of dealing with situations like this is to make the server's behavioral choice available to the client as an attribute, as is provided for in [RFC8178]

In the case of ACLs, we have a difficult situation to resolve. Instead of having a small set of individual mistakes which can now be recognized as such, we have a situation in which the existing specifications have created an unacceptable interoperability situation in relation to ACL implementations. Existing specifications have not paid proper attention to the need to make decisions in the face of disagreements regarding proper server behavior and have in various ways avoided the need to compromise and reach a reasonable consensus but instead have made it the job of the specifications to consider valid any remotely similar server implementations as valid, leaving clients little that could do other than to accept a wide range of server behavior as valid, simply because it was chosen by

the server. How this set of issues is to be addressed is discussed in Section 1.2 of [I-D.dnoveck-nfsv4-acls].

2. Requirements Language

2.1. Keyword Definitions

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 specified in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Special Considerations

Because this document needs to revise previous treatments of its subject, it will need to cite previous treatments of issues that now need to be dealt with in a different way. This will take the form of quotations from documents whose treatment of the subject is being obsoleted, most often as direct quotation but sometimes as indirect ones as well.

Paragraphs headed "[Previous Treatment] or otherwise annotated as having that status, as described in Section 1, can be considered quotations in this context.

Such treatments in quotations will involve use of these BCP14-defined terms in two noteworthy ways:

* The term may have been used inappropriately (i.e not in accord with [RFC2119]), as has been the case for the "**RECOMMENDED**" attributes, which are in fact **OPTIONAL**.

  In such cases, the surrounding text will make clear that the quoted text does not have a normative effect.

  Some specific issues relating to this case are described below in Section 8.1.

* The term may been used in accord with [RFC2119], although the resulting normative statement is now felt to be inappropriate.

  In such cases, the surrounding text will need to make clear that the text quoted is no longer to be considered normative, often by providing new text that conflicts with the quoted, previously normative, text.

  An important instance of this situation is the description of AUTH_SYS as an "**OPTIONAL**" means of authentication". For detailed discussion of this case, see Sections 14 and 18.1.5

3. Introduction to this Update

There are a number of noteworthy aspects to the updated approach to NFSv4 security presented in this document:

* There is a major rework of the security framework to take advantage of work done in [RFC9289], as described in Section 1.1.

  NFSv4 security is still built on RPC, as had been done previously. However, it is now able to take advantage of security-related facilities provided on a per-connection basis. For more information about this transformation, see Section 3.1.

  For an overview of changes made so far as part of this rework, see Appendix A.1.

* This document deals with all minor versions together, although there is a need for exceptions to deal with, for example, pNFS security.

  For more detail about how minor version differences will be addressed, see Sections 3.2 and 3.3.

* There is a new Security Considerations section including a threat analysis.

* There has been extensive work to clarify the multiple types of authorization within NFSv4 and deal more completely with the co-ordination of ACL-based and mode-based file access authorization. this work is discussed in Section 3.4

3.1. Per-connection Security Features

There are a number of security-related facilities that can be provided on a per-connection basis, eliminating the need to provide such support on a per-request basis, based on the RPC auth-flavor used.

These will initially be provided, in most cases, by RPC-with-TLS but similar facilities might be provided by new versions of existing transports or new RPC transports.

* The transport or a layer above it might provide encryption of requests and replies, eliminating the need for privacy and integrity services to be negotiated later and applied on a per-request basis.

  While clients might choose to establish connections that provide such encryption, servers can establish policies allowing access to certain pieces of the namespace using such security facilities, or

limiting access to those providing privacy, allowing the use of either per-connection encryption or privacy services provided by RPCSEC_GSS.

*The transport or a layer above it might provide mutual authentication of the client and server peers as part of the establishment of the connection This authentication is distinct from the the mutual authentication of the client user and server peer, implemented within the RPCSEC_GSS framework.

This form of authentication is of particular importance when the server allows the use of the auth-flavors AUTH_SYS and AUTH_NONE, which have no provision for the authentication of the user requesting the operation.

While clients might choose, on their own,to establish connections without such peer authentication, servers can establish policies a limiting access to certain pieces of the namespace without such peer authentication or only allowing it when using RPCSEC_GSS.

To enable server policies to be effectively communicated to clients, the security negotiation framework now allows connection characteristics to be specified using pseudo-flavors returned as part of the response to SECINFO and SECINFO_NONAME. See Section 16 for details.

3.2. Handling of Multiple Minor Versions

In some cases, there are differences between minor versions in that there are security-related features, not present in all minor versions.

To deal with this issue, this document will focus on a few major areas listed below which are common to all minor versions.

*File access authorization (discussed in Section 8) is the same in all minor versions together with the identification/ authentication infrastructure supporting it (discussed in Section 14) provided by RPC and applying to all of NFS.

An exception is made regarding labelled NFS, an optional feature within NFSv4.2, described in [RFC7862]. This is discussed as a version-specific feature in this document in Section 11.

*Features to secure data in-flight, all provided by RPC, together with the negotiation infrastructure to support them are common to all NFSv4 minor versions, are discussed in Section 16.

However, the use of SECINFO_NONAME, together with changes needed for connection-based encryption, paralleling those proposed here

for SECINFO, is treated as a version-specific feature and, while mentioned here, will be fully documented in new NFSv4.1 specification documents.

* The protection of state data from unauthorized modification is discussed in Section 12) is the same in all minor versions together with the identification/ authentication infrastructure supporting it (discussed in Section 14 by security services such as those provided by RPC-with-TLS.

  It needs to be noted that state protection based on RPCSEC_GSS is treated as a version-specific feature and will continue to be described by [RFC8881] or its successors. Also, it needs to be noted that the use of state protection was not discussed in [RFC7530].

3.3. Handling of Minor-version-specific features

There are a number of areas in which security features differ among minor versions, as discussed below. In some cases, a new feature requires specific security support while in others one version will have a new feature related to enhancing the security infrastructure.

How such features are dealt with in this document depends on the specific feature.

* In addition to SECINFO, whose enhanced description appears in this document, NFSv4.1 added a new SECINFO_NONAME operation, useful for pNFS file as well as having some non-pNFS uses.

  While the enhanced description of SECINFO mentions SECINFO_NONAME, this is handled as one of a number of cases in which the description has to indicate that different actions need to be taken for different minor versions.

  The definitive description of SECINFO_NONAME, now appearing in [RFC8881] needs to be modified to match the description of SECINFO appearing in this document. It is expected that this will be done as part of the rfc5661bis process.

  The security implications of the security negotiation facilities as a whole will be addressed in the security considerations section of this document.

* The OPTIONAL pNFS feature added in NFSv4.1 has its own security needs which parallel closely those of non-pNFS access but are distinct, especially when the storage access protocol used are not RPC protocols. As a result, these needs and the means to satisfy them are not discussed in this document.

The definitive description of pNFS security will remain in [RFC8881] and its successors (i.e. the eventual rfc5661bis document). However, because pNFS security relies heavily on the infrastructure discussed here, it is anticipated that the new treatment of pNFS security will deal with many matters by referencing the overall NFS security document.

The security considerations section of rfc5661bis will deal with pNFS security issues.

*In addition to the state protection facilities described in this document, NFS has another set of such facilities that are only implemented in NFSv4.1.

While this document will discuss the security implications of protection against state modification, it will not discuss the details of the NFSv4.1-specific features to accomplish it.

*The additional NFSv4.1 acl attributes, sacl and dacl, are discussed in this document, together with the ACL inheritance features they enable.

As a result, the responsibility for the definitive description of these attributes will move to overall NFS security document, with the fact that they are not available in NFSv4.0 duly noted. While these attributes will continue to be mentioned in NFSv4.1 specification documents, the detailed description appearing in [RFC8881] will be removed in successor documents.

*Both NFSv4.0 and NFSv4.1 specifications discussed the coordination of the values the mode and ACL-related attributes. While the treatment in [RFC8881] is more detailed, the differences in the approaches are quite minor.

[Consensus Item #25a]: This document will provide a unified treatment of these issues, which will note any differences of treatment that apply to NFSv4.0. Changes applying to NFSv4.2 will also be noted.

As a result, this document will override the treatment within [RFC7530] and [RFC8881]. This material will be removed in the rfc5661bis document suite and replaced by a reference to the treatment in the NFSv4 security RFC.

*The protocol extension defined in [RFC8257], now part of NFSv4.2, is also related to the issue of co-ordination of acl and mode attributes and will be discussed in that context.

Nevertheless, the description in [RFC8257] will remain definitive.

*The NFSv4.1 attribute set-mode-masked attribute is mentioned together with the other attributes implementing the POSIX authorization model.

Because this attribute. while related to security, does not substantively modify the security properties of the protocol, the full description of this attribute, will continue to be the province of the NFSv4.1 specification proper.

*There is a brief description of the v4.2 Labelled NFS feature in Section 11. Part of that description discusses the limitations in the description of that feature within [RFC7862].

Because of some limitations in the description, it is not possible to provide an appropriate security considerations section for that feature in this document.

As a result, the responsibility for providing an appropriate Security Considerations section remains, unrealized for now, with the NFSv4.2 specification document and its possible successors.

3.4. Features Needing Extensive Clarification

For a number of authorization-related features, the existing descriptions are inadequate for various reasons:

*In the description of the use of the mode attribute in implementing the POSIX-based authorization model, critical pieces of the semantics are not mentioned, while, ironically, the corresponding semantics for ACL-based authorization are discussed.

This includes the authorization of file deletion and of modification of the mode, owner and owner-group attributes. For ACL-based authorization, there is an attempt to provide the corresponding description.

*The description of authorization for ACLs is more complete but it needs further work, because the previous specifications make extensive efforts, in my view misguided, to allow an enormous range of server behaviors, making it hard for a client to know what the effect of many actions, and the corresponding security-related consequences, might be.

Troublesome in this connection were the discussion of ACE mask bits which essentially treats every mask bit, as its own OPTIONAL feature, the use of "**SHOULD**" and "**SHOULD NOT**" in situations which it is unclear what valid reasons to ignore the recommendation might be, and cases in which it is is simply stated that some servers do some particular thing, leaving the unfortunate

implication that clients need to be prepared for a vast range of server behaviors.

This approach essentially treated ACLs in a manner appropriate to an experimental feature even though in appeaed in a Proposed Standard.

*Similar issues apply to descriptions related to the need to co-ordinate the values of the mode attribute and the ACL-related attributes.

Although the need for such coordination is recognized. There are multiple modes of mapping an ACL to a corresponding mode together with multiple sources of uncertainty about the reverse mapping.

In addition, certain of the mapping algorithms have flaws in that their behavior under unusual circumstances providing results that appear erroneous.

Dealing with these issues is not straightforward, because the appropriate resolution will depend on:

*The actual existence of server implementations with non-preferred semantics.

In some cases in which "**SHOULD**" was used, there may not have been any actual severs choosing to ignore the recommendation, eliminating the possibility of compatibility issues when changing the "**SHOULD**" to a formulation that restricts the server's choices.

*The difficulty of modifying server implementations to eliminate or narrow the effect of non-standard semantics.

One aspect of that difficulty might be client or application expectations based on existing server implementations, even if the existing specifications give the client no assurance that that server's behavior is mandated by the standard.

*Whether the existing flaw in some existing recommended actions to be performed by the server is sufficiently troublesome to justify changing the specification at this point.

This sort of information will be used in deciding whether to:

*Narrow the scope of allowable server behavior to those actually used by existing severs.

*Limiting the negative effects of unmotivated **SHOULD**s by limiting valid reasons to ignore the recommendation to the difficulty of changing existing implementations.

This would give significant guidance to future implementations, while forcing clients to live with the uncertainty about existing servers

*Tie a more restricted set of semantics to nominally unrelated OPTIONAL features such as implementation of dacl and sacl.

This would provide a way to allow the development of newer servers to proceed on a firmer basis, without requiring changes on older servers that do not support these SMB-oriented attributes.

*Provide means that clients could use to determine, experimentally, what semantics are provided by the server.

Would need to be supported by a requirement/assurance that a server behave uniformly, at least within the scope of a single file system.

*Allow the provision of other ways for the client to know the semantics choices made by the server or the file system.

Despite the difficulty of addressing these issues, if the protocol is to be secure and ACLs are to be widely available, these problems have to be addressed. While there has not been significant effort to provide client-side ACL APIs and there might not be for a while, we cannot have a situation in which the security specification makes that development essentially impossible.

3.5. Process Going Forward

Because of the scope of this document, and the fact that it is necessary to modify previous treatments of the subject previously published as Proposed Standards, it is necessary that the process of determining whether there is Working Group Consensus to submit it for publication be more structured than that used for the antecedent documents.

In order to facilitate this process, the necessary changes which need to be made, beyond those clearly editorial in nature, are listed in Appendix B. As working group review and discussion of this document and its successors proceeds, there will be occasion to discuss each of these changes, identified by the annotations described in Section 1.2.

Based on working group discussions, successive document versions will do one of the following for some set of consensus items:

*Deciding that the replacement text is now part of a new working group consensus.

When this happens, future drafts of the document will be modified to remove the previous treatment, treat the proposed text as adopted, and remove Author Asides or replace them by new text explaining why a new treatment of the matter has been adopted or pointing the reader to an explanation in Appendix A.

At this point, the consensus item will be removed from Appendix B and an explanation for the change will be added to Appendix A.

*Deciding that the general approach to the issue, if not necessarily the specific current text has reached the point of "general acceptance" as defined in Appendix B

In this case, to facilitate discussion of remaining issues, the text of the document proper will remain as it is.

At this point, the consensus item will be marked within the table in Appendix B as having reached general acceptance, indicating the need to prioritize discussion in the next document cycle, aimed at arriving at final text to address the issue.

In addition, an explanation for the change will be added to Appendix A.

*Deciding that modification of the existing text is necessary to facilitate eventual consensus, based on the working group's input.

In this case, there will be changes to the document proper in the next draft revision. In some cases, because of the need for a coherent description, text outside the consensus item may be affected.

The table in Appendix B will be updated to reflect the new item status while Appendix A is not expected to change.

*Deciding that the item is best dropped in the next draft.

In this case, the changes to the document proper will be the inverse of those when a change is accepted by consensus. The previous treatment will be restored as the current text while the proposed new text will vanish from the document at the next draft revision. The Author Aside will be the basis for an explanation of the consequences of not dealing with the issue.

At this point, the consensus item will be removed from Appendix B.

The changes that the working group will need to reach consensus on, either to accept (as-is or with significant modifications) or reject can be divided into three groups.

*A large set of changes, all addressing issues mentioned in Section 1.1, were already present in the initial I-D so that there has been the opportunity for working group discussion of them, although that discussion has been quite limited so far.

  As a result, a small set of these changes is marked, in Appendix B, as having reached general acceptance.

  That subset of these changes changes, together with the organizational changes to support them are described in Appendix A.1.

*Another large set of changes were made in draft -02. These mostly concern the issues mentioned in Section 3.4 None of these changes is yet considered to have reached general acceptance.

  The issues that need to be addressed are described in Appendix A.2 while the possible approaches that might be taken to resolve these issues are described in Appendix A.3.

*There remain a set of potential changes for which a need is expected but for which no text is yet available.

  Such changes have associated Author Asides and are listed in Appendix B.

  The text for these changes is expected to be made available in future document revisions and they will be processed then, in the same way as other changes will be processed now.

  If and when such changes reach general acceptance, they will be explained in the appropriate subsection of Appendix A.

4. Introduction to NFSv4 Security

Because the basic approach to security issues is so similar for all minor versions, this document applies to all NFSv4 minor versions. The details of the transition to an NFSv4-wide document are discussed in Sections 3.2 and 3.3.

NFSv4 security is built on facilities provided by the RPC layer, including various auth-flavors and other security-related services provided by RPC.

Support for multiple auth flavors can be provided. Not all of these actually provide authentication, as discussed in Section 14.

- Support for RPCSEC_GSS is **REQUIRED**, although use of other auth-flavors is provided for.

  This auth-flavor provides for mutual authentication of the principal making the request and the server performing it.

  This auth-flavor allows the client to request the provision of encryption-based services to provide privacy or integrity for specific requests. Although such services are often provided, on a per-connectio basis, by RPC, this support is useful, when such services are not supported or are otherwise unavailable.

- AUTH_SYS, provides identification of the principal making the request but **SHOULD NOT** be used unless the client peer sending the request can be authenticated and there is protection against the modification of the request in flight.

  Both of the above require specific RPC support such as that provided by RPC-with-TLS [RFC9289].

- AUTH_NONE does not provide identification of the principal making the request so would only be used for requests for which there is no such principal or for which it would irrelevant.

  The restrictions mentioned above for AUTH_SYS apply to AUTH_NONE as well.

There are important services that can be provided by RPC, when RPC-with-TLS or similar transport-level facilities are available.

- Such services can provide data security to all requests on the connection. This is to be preferred to data security provided by the RPC auth flavor because it provides protection to the request headers, because it applies to requests using all authentication flavors, and because it is more likely to be offloadable.

- These services can authenticate the server to the client peer. This is desirable since that authentication applies even when AUTH_SYS or AUTH_NONE is used.

- The client-peer can be authenticated to the server at the time the connection is set up. This is essential to allow AUTH_SYS to be used with a modicum of security, based on the server's level of trust with regard to the client peer.

Because important security-related services depend on the security services, rather than the auth flavor, the process of security

negotiation, described in Section 16, has been extended to provide for the negotiation of appropriate connection characteristics at connection time if the server's policy limits the range of transports being used and also when use of a particular auth flavor on a connection with inappropriate security characteristics causes NFS4ERR_WRONGSEC to be returned.

The authentication provided by RPC, is used to provide the basis of authorization, which is discussed in general in Section 7. This includes file access authorization, discussed in Sections 8 through 10 and state modification authorization, discussed in Section 12

File access is controlled by the server support for and client use of certain recommended attributes, as described in Section 8.1. Multiple file access model are provided for and the considerations discussed in Section 9 apply to all of them.

- The mode attribute provides a POSIX-based authorization model, as described in Section 8.3

- The ACL-related attributes acl, sacl, and dacl (the last two introduced in NFSv4.1) support a finer grained authorization model and provide additional security-related services. The structure of ACLs is described in new ACL document [I-D.dnoveck-nfsv4-acls].

  The ACL-based authorization model is described in Section 8.4

  The additional security-related services are described in Section 13. These also rely on the authentication provided by RPC.

- Because there are two different approaches to file-access authorization, servers might implement both, in which case the associated attributes need to be coordinated as described in Section 10.

- NFSv4.2 provides a file access authorization model oriented toward Mandatory Access Control. It is described in Section 11. For reasons described there, its security properties are hard to analyze in detail and this document will not consider it as part of the NFSv4 threat analysis.

Authorization of locking state modification is discussed in Section 12. This form of authorization relies on the authentication of the client peer as opposed to file access authorization, which relies on authentication of the client principal.

4.1. NFSv4 Security Terminology

In this section, we will define the security-related terminology used in this document. This is particularly important for NFSv4 because

many of the terms terms related to security in previous specification may be hard to understand because their meanings have changed or have been used inconsistently, resulting in confusion.

The following terms are listed in alphabetical order:

* "Access Control" denotes any control implemented by a server peer to limit or regulate file system access to file system objects. It includes but is not limited to authorization decisions. Access control features can be divided into those which are "Discretionary" or "Mandatory" as described below.

* "ACL" or "Access Control List" denotes a structure used, like the mode (see below), to defines the privileges that individual users have with respect to a given file. These structures provide more options than modes with regard to the association of privileges with specific users or group and often provide a finer-graned privilege structure as well. This specification will have need to refer to two types of ACLs.

  The ACLs intended to be presented in the acl, sacl, and dacl attributes are called "NFSv4 ACLs". This ACL format, was modeled on the the semantics of the SMB ACL format which provide a privilege model substantially finer-grained than that provided by POSIX modes.

  [Consensus needed (Item #56a)]: Another ACL type derives from an attempt to define, within POSIX, a UNIX-oriented approach to ACLs which was published as a draft (POSIX 1003.1e draft 17), but subsequently withdrawn. Despite the withdrawal of this draft and the working group's decision to adopt a native NFsv4 ACL format based on SMB ACLs, this document will have to discuss these ACLs, which we will term "UNIX ACLs" because many server file systems do not support the finer-grained privilege model needed by the the NFSv4 ACL model and because many clients are built on systems whose only ACL-related API is based on the UNIX ACL model.

* "authentication" refers to a reliable determination that one making a request is in fact who he purports to be. Often this involves cryptographic means of demonstrating identity.

  This is to be distinguished from "identification" which simply provides a specified identity without any evidence to verify that the identification is accurate.

  In the past, these terms have been confused, most likely because of confusion engendered by th use of the term "authentication flavor" including flavors for which only identification is provided or which do not provide even identification.

*"authorization" refers to the process of determining whether a request is authorized, depending on the resources (e.g. files) to be accessed, the identity of the entity on whose behalf the request was issued, and the particular action to be performed.

Depending on the type of request, the entity whose identity is referenced can be a user, a peer, or a combination of both.

Authorization is distinct from authentication. However, performing authorization based on identities which have not been authenticated makes secure operation impossible since use of unauthenticated identities allows acceptance of requests that are not properly authorized if the sender has the ability, as it typically does, to pretend to be an authorized user/peer.

*"client" refers to the entity responsible for setting up a connection. In most cases the client and the requester reside on the same node but this not always the case for NFSv4 because of the possibility of callback requests in which the server makes some request of the client.

*"confidentiality" refers to the assurance provided, typically through encryption, that the contents of requests and responses are not inadvertently disclosed to unauthorized parties.

*"Discretionary Access Control" denotes forms of access control, that rely on a user, such as the owner, specifying the privileges that various users are to have.

*"Mandatory Access Control" denotes forms of access control that reflect choices made by the server peer and based on its policy and that are typically based on the identity of the client peer rather than the specific user making a request. While such access control is discussed in this document, it is important to note that many forms of mandatory access control are discussed by other NFsv4 documents and that there are forms that are not standardized.

*[Consensus Needed, Entire Bulleted Item (Items #21a, #57a)]: "Mode" designates a set of twelve flag bits used by POSIX-based systems to control access to the file with which it is associated. In NFSv4, there are represented by the **REQUIRED** attribute Mode.

The three high-order flags are generally accessed only by the client while low-order bits are divided into three three-bit fields, which give, in order of decreasing numeric value, the privileges to be associated with, the owner of the file, other users in the group owning the file, and users not in the above two categories.

In most cases, the privileges associated with each successive group are no greater than those for the previous group. Modes whose privileges are of this form are referred to as "forward-slope modes" because the privilege level proceeds downward as successive groups of users are specified. Cases in which the contrary possibility is realized are referred to as "reverse-slope modes".

*"peer" refer to the entity which is charged with requesting or performing a specified request as opposed to the entity on whose behalf the request is requested or performed, the principal;

*"principal" refers to the specific entity (e.g. user) on whose behalf a request is being made.

*"privacy", has in the past been used to refer, to what is now referred to as "confidentiality".

over time, this usage has changed so that the word most often refers to applicability of data to a single individual and person's right to prevent its unauthorized disclosure

As a result, many references to "privacy" in previous are no longer appropriate and really refer to confidentiality.

The NFSv4 protocol has no way to determine whether particular data items raise privacy concerns (In the new sense). NFSv4 provides confidentiality whatever type of data is being accessed so that private data is kept private.

*"integrity" refers to the assurance that data in a request has not been modified in the process of transmission. Such an assurance is generally provided b means of a cryptographic hash of the requests or response.

*"requester" is the entity making a request, whether that entity is on the client-side, as it most often is (forward-direction request) or the server side, in th case of callback (reverse-direction requests)

*"responder" is the entity performing a request, whether that entity is on the server side, as it most often is (forward-direction request) or the client side, in the case of callbacks (reverse-direction requests.

*"server" refers to the entity to which the client connects. In most cases the client and the responder reside on the same node but this not always the case for NFSv4 because of the possibility of callback requests in which the server makes some request of the client.

4.2. NFSv4 Security Scope Limitations

This document describes the security features of the NFSv4 protocol and is unable to address security threats that are inherently outside the control of the protocol implementors. Such matters as out of this document's scope.

As a way of clarifying the threats that this document, and the threat analysis in Section 18.4 can and cannot deal with, we list below the potential threats discussed Section 3.1 of [nist-209] and review how, if at all, it is discussed in the current document. In cases in which the threat is dealt with in this document, distinctions are to be made between cases in which the issues have been dealt with directly or have been delegated to a lower layer on which the protocol is built and whether the issue has been addressed by the changes to NFSv4 security made by this document.

*Regarding the possibility of "Credential Theft or Compromise", this is not a matter that the NFSv4 protocols concern themselves with or can address directly, despite its importance for security. Depending on the auth flavor chosen, either the client (for AUTH_SYS) or a third-party (for RPCSEC_GSS), usually Kerberos, will be responsible for credential verification.

  Since experience has shown that credential compromise (e.g. through "phishing" attacks) is a common occurrence, this problem cannot be ignored, even though NFSv4's reliance on RPC facilities for authentication might be thought to make it out-of-scope as it would be RPC if had an effective solution to the issue. However, the urgency of the situation this issue is such that will be discussed in Section 18.4.2, even though no definitive solutions to this issue are likely before this document is completed and published.

  Regardless of such issues, the likelihood of such compromise has had a role in decisions made regarding the acceptance and use of "superuser" credentials. The possibility of such compromise is also relevant to implementation of means to synchronize credentials when they are managed by the client, as described in Section 18.4.6.1

*Regarding the possibility of "Cracking Encryption", prevention of this is responsibility of the NFSv4 protocols but it is one which has been delegated to RPC, so that its discussion in Security Considerations will rely on RPCSEC_GSS and RPC-with-TLS impelentations to manage the selectio and replacement of kets for encryption so as to limit the possibility of such unwanted encryption key discovery.

*Regarding the possibility of "Infection of Malware and Ransomware", NFSv4 has no direct role in preventing such infection, but does have an important role in limiting its consequences, by limiting the the ability of Malware to access or modify data, through the file access authorization model supported by NFSv4 to limit access to authorized users. Of course, malware will be able to execute on behalf of the user mistakenly invoking it but the authorization model will server to limit the potential damage.

  The possibility of vertical privilege escalation is of concern as regard the possible elevation to "superuser" privileges. For this reason, this document recommends that any such escalation not be effective on the server, even if it happens on local clients for which NFSv4 has no role.

  Execution of a ransomeware-based attack requires the attacker to have the ability to read existing data and replacing it with an encrypted version together with the ability to temporarily hide the encryption from ongoing operations by intercepting requests to read encrypted data and substitute the unencrypted data.

*Regarding the possibility of "Backdoors and Unpatched Vulnerabilities", it needs to be noted that the NFSv4 protocols do not specify any backdoors even though it is possible that might choose to provide such backdoors. Since it is not practical to specifically prohibit the existence of such backdoors nor would they be enforceable if written, this document will not attempt to do so. Instead, Section 18.2.3 will note the possibility of such backdoors and recommend against any such implementation, and include implementations containing backdoors in the category of insecure use that will not be dealt with in Section 18.4.

  Although it is expected that vulnerabilities will be due to incorrect implementations and thus outside the scope of this document, the possibility of a protocol design errors cannot be excluded. In dealing with such eventualities, it is likely that complete remediation would require co-ordinated changes on the client and server

*Regarding the possibility of "Privilege Escalation", NFSv4 has dealt with the possibility of vertical escalation by not allowing a client-local escalation to superuser privileges to be effective on the server.

  With regard to horizontal "escalation", NFSv4 provides for the use of various means RPC authentication of principals but relies on the client operating system to make sure that one user principal cannot masquerade as another.

*Regarding the possibility of "Human Error and Deliberate Misconfiguration", the approach taken is to limit the need for the server to make complicated decisions regarding the security requirements of each section of its namespace, with many opportunities for misconfiguration, if the chosen security requirements are insufficiently restrictive. This is in contrast to previous specifications which made such configuration the centerpiece of the security approach.

Although it is possible to create configurations where certain data, generally publicly accessible, are to be made available without encryption, this is expected to be a rarely used option with the possibility of in-transit modification kept in mind before adopting such use.

*Regarding the possibility of "Physical Theft of Storage Media", this a matter which, while of concern to those deploying NFSv4 server, will be considered out-of-scope since there is nothing that the protocol could do to deal with this threat.

*Regarding the possibility of "Network Eavesdropping", when the protocol implementation follows the recommendations in this document, the protocol's use of RPC facilities is designed, through the consistent use of encryption to make it difficult for an attacker to have access to the data being transmitted, to modify it, or inject requests into an existing data stream.

The possibility of an attacker with access to the network creating a new connection is best considered as a case of the attacker pretending to be a client and is addressed in Section 18.4.3.

*Regarding the possibility of "Insecure Images, Software and Firmware", while attention to such matters is important for those deploying NFSv4, it is important to note that these are matters outside the control the NFSv4, which has to assume that the infrastructure it is built is working properly. As a result, this document will not deal with the possibility of such threats.

5. Authorization-related Attributes

NFSv4 operations are authorized (or not) based on the entity requesting the operation and the values of the authorization-related attributes documented in this section. The table in Section 5.2 lists all such attributes with the actual descirptions appearing in three subsections based on the specfic authorization models supported by various NFSv4 protocols.

*There are a number of attributes derived from file characteristics defined by POSIX. They are similar to the corresponding NFSv3

attributes, although they are different in form. These are described in Section 5.3.

* In order to provide finer-grained control of authorization decisions, a number of acl-related attibutes are defined. These are described in Section 5.4.

  [Consensus Required (Item #57b)]: Although it might have been intended that the acl-related and POSIX-derived attributes would serve as two alternate modes authorization, with each **OPTIONAL**, that has not been possible, with the POSIX-derived model becoming **REQUIRED** while the acl-related one remains **OPTIONAL**.

  As the acl-related model has evolved, it was constrained to to work well with the POSIX-based model, as there were many clients who required support for POSIX authorization semantics.

* In addition, an addional authoization model was made availabe in NFSv4.2. It is described in Section 5.5.

  This provides a form of Mandatory Access Control in which authorization decisions derived from the identity of the client making the request rather than the identifty of the specfic user/principal. When enabled, it serves as an additional authorization step in addition to that specified by POSIX-related attributes or ACLs.

The NFSv4 protocols have integrated a set authorization-related attributes within the extensible attribute model, introduced in [RFC7530]. This extensibility has been the basis of the introduction of additional **OPTIONAL** attriutes provided for in [RFC8178]

As a result, different sets of attributes are valid in different minor versions. Although attribubes are described in the specfications for the minor version in which they are introduced, and, in some cases, in specifications for later minor versions, the description in this document is definitive and overrides any other such specfication. This includes the following attributes:

* The attributes owner, group, mode, aclsupport and acl were introduced in NFSv4.0 and are valid in all minor versions.

* The attribute mode_set_masked, dacl, and sacl were introduced in NFSv4.1 and are valid in all minor versions except minor version zero.

* The attribute sec_label was introduced in NFSv4.2 and is only valid in minor version two.

[Consensus Needed, Including List (Items #57b, #58a)]:", Although previous specifications have treated all of these as **OPTIONAL**, sometimes using the incorrect designation "RECOMMENDED", it has to be understood that there important exceptions that need to be noted:

- In the interest of providng better assurances of meaningful interoperaability to compliant servers and clients, the attributes described in Section 5.3 (i.e. owner, group, mode) are to be considered **REQUIRED**.

- While it is reasonable to say, as [RFC8881] does, that **OPTIONAL** need to be "understood well enough to warrant support", it should be the case that this understanding is documented sufficiently to enable clients and servers to interoperate and new implementations of each to be implemented. Unfortunately, that condition is not currently met for some attribues which we need to realize are under-specified, and thus essentially experimental, even though formally **OPTIONAL**. The details are discussed in Sections 3.4, 3.7, and 3.8 of [I-D.dnoveck-nfsv4-acls] and Section 5.5.1) ofthis document. The intention, not yet fully realized, is that new features described in the ACL document might, when adopted, allow the Acl, Dacl and Sacl attributes to be designated as **OPTIONAL** in fact, i.e. not under-specified.

  [Author Aside (Item #58a)]: It could be that, as discussed elsewhere, that the inadequate semantic description referred to in the paragraph above is the result of a flawed approach to the description of the features, rather than to the attempt to support two types of ACLs within the same set of attributes.

5.1. Format of Id Strings in Authorization-related Attributes

Unlike the case in NFSv3, users and groups are represented in NFSv4 by UTF-8-encoded Unicode strings. These strings include:

- [Consensus needed (Item #57c)]: The values of the **REQUIRED** attribute Owner, as described in Section 5.3.3.

- [Consensus needed (Item #57c)]: The values of the **REQUIRED** attribute Owner_group, as described in Section 5.3.4

- Values within the "who" field within Access Control Entries as described in Section 5.4 of [I-D.dnoveck-nfsv4-acls]. These entries appear in the Acl, Dacl, and Sacl attributes.

[Author Aside, Including List (Item #59a)]: This section had to be extensively revised in order to be clearer and to allow more uses of string representations of numeric ids. The previous treatment, which

is not reproduced here, can be found in Section 5.9 of [RFC8881]. In the author's opinion, major revisions were necessary because:

* The existing treatment, by leaving the methods of mapping between strings and local user ids unspecified, created an interoperability issue. Since providing this mapping in a secure way is difficult, the gap is likely to be filled in an a way vulnerable to attack.

  While the need for flexibility might make it impossible to specify mapping fully, the requirements for agreement between client and server needed to be specified more clearly and avoided where this can be done without impeding protool operation.

* There is inadequate attention to the additional implementation needs to support multiple domain values.

  The difficulties inherent in supporting all possible domains are obscured, with the very real security issues essentially ignored.

* The description of the domain value as "meant to be a DNS domain" is not clear and makes it harder to focus on the real difficulties involved in providing coherent secure means of providing necessary mappings for multiple domains.

* Some negarive characterizations of the use of numeric ids appear to be unjustified, leading to their non-use in situations in which artificially mapping to the name@domain format creates difficult issus while providing no benefit.

[Consensus Needed, Including Rest of Section (Items #52a, #59a)]:

These identifiers are represented by strings which have two possible formats:

* Strings of the format "name@domain" can be used with a number of important benefits. They are to be preferred in situations in which the mapping between names and the 32-bit numeric ids preferred by typical file systems is securely provided by means of the RPC auth flavor being used, as is the case when Kerberos services are made available through the use of RPCSECGSS.

  Servers would be able support multiple clients each with users from one or more domains, without requiring construction of a single list of users together with associated authentication info.

  The number of users would not be limited by existing hard-to-extend limits such as the POSIX/NFSv3 use of a 32-bit field to represent a user or group id.

* The string can be the ASCII representation o the 32-bit user or group id.

  This format is to preferred when the auth flavor AUTH_SYS is being used and in other cases in which the translation between user names and numeric ids is not available or creates security vulnerabilities.

These strings are assigned the xdr type utf8str_mixed because the name and domain portions are treated differently:

* The name portion is a utf8 string that is matched in a case-sensitive manner without any attempt to treat distinct canonically equivalent strings as the same.

* The handling of the domain portion, including recognition of equivalent string as the same is specified by the server. However, when multple domains are supported, case-insensitivity of ASCII characters is mandated by DNS, as well as translations to and from punycode-encoded forms.

It is expected that the client and server file systen will have their own local representation of users and user groups that is used for local storage or presentation to the end user. In addition, it is expected that when these attributes are transferred between the client and server and the format used is of the form name@domain, the local representation is translated to that orm

Although this allows for a client and server that both use the name@domain format and do not use the same local representation the ability to translate to a common format, there will often be cases in which both client and sever use the same internal format. This often happens with servers that only support a single domain. However, regrdless of whether these forms are, in fact, different, it is inecessary, when such translation is necessary, that client and server agree on the following:

* The set of valid strings that can appear as the domain portion of these identifiers. This set must have at least a single member but when this not the case, the set known to each client must be a subset of the set recognized by the server.

* For each valid domain string passed by a client and accepted by a server, the server and client must agree on a means by which the user designated by each user and group name can be mapped to a user or group so that each interprets all name@domain strings identically.

  The above applies whether the client and server use the same internal name repreentation or not.

When this translation is used, security principals and the groups they belong to need to be integrated within this arrangement since these identities need to be presented to the server in a form compatible with this new format. The specfics depend on the auth flavor providing the principal identfication:

- When auth flavors based on RPCSEC_GSS are used, the prinipal is generally identified in a form easily converted to the name@domain fornat. For example, with Kerberos, the kerberos user name can provide the name ortion while kerberos realm can serve as the domain portion.

- When the AUTH_SYS auth flavor is used, principals are identified by a 32-bit numeric identifier which can be generally teated in the same manner as numeric id's used locally within the file system to arive at a name with the domain being the one assigned that filesysten.

  The idenification of principals **MAY** be subject to filtering to eliminate users with a high level of privilege.

  In the case of servers with filesystems that support the use of multiple domains, clients for which use of AUTH_SYS is supported need to be assigned to a specific domain so that the principal identfications received are properly dealt with.

The translation used to interpret owner and group strings is not specified as part of the protocol. This allows various solutions to be employed, as long as the requirement mentioned above are satisfied and additionl security vulnerabilities are not created. For example, a local translation table may be consulted that maps a numeric identifier to the user@domain syntax. A name service may also be used to accomplish the translation. A server may provide a more general service, not limited by any particular translation (which would only translate a limited set of possible strings) by storing the owner and owner_group attributes in local storage without any translation or it may augment a translation method by storing the entire string for attributes for which no translation is available while using the local representation for those cases in which a translation is available.

[Author Aside:] The next sentence previously used the word "**SHOULD**. Since the author has been unable to determine valid reasons to do otherwise and has no reason believe that any exist, the new text uses "**MUST**". It is possible that the original author might have been thinking of "nobody" for this but that case needs to be addressed separately and will need to be clearly distinguished from "nobody@domain".

Servers that do not provide support for all possible values of the owner and owner_group attributes **MUST** return an error (NFS4ERR_BADOWNER) when a string is presented that has no translation, as the value to be set for a SETATTR of the owner, owner_group, or as the who value in an ACE within the acl, sacl, or dacl attributes. When a server does accept an owner or owner_group value as valid on a SETATTR (and similarly for the owner and group strings in an ACL), it is promising to return that same string when a corresponding GETATTR is done. Configuration changes (including changes from the mapping of the string to the local representation) and ill-constructed name translations (those that include aliasing) may make that promise impossible to honor. Servers need to make appropriate efforts to avoid a situation in which these attributes have their values changed when no real change to ownership has occurred.

The "domain" portion of the owner string will often be a DNS domain name, for example, user@example.org. Servers should accept as valid a set of users for at least one value of the domain portion. A server may treat other domains as having no valid translations. A more general service is provided when a server is capable of accepting users for multiple domains values but the following will need to be attended to:

- When use of AUTH_SYS is allowed, the translation of numeric ids to the form name@domain becomes problematic.

  The server will need to determine the domain value to be used based on the identity of the client peer.

- When numeric ids are used in the file system the handling of those ids needs to be modified to support multiple sets of users with one set for each of the various domain values supported.

  One option is to add a small (i.e. 8- or 16-bit) field to represent the chosen domain value will leaving the 32-bit id field as it is.

  It is also possible to map individual 32-bit id spaces into a single 32-bit id space. However, this involves eastablishing for each component id space a restricted mange, so that the mapping can be done without resulting in conflicting reverse mappings.

In cases in which translation is being used and there is no translation available to the client or server, the attribute value will be constructed without the "@". The absence of the @ from the user or group string signifies that no translation was available at the sender and that the receiver of the attribute should not use that string as a basis for translation into its own internal format. When

the string consists of a numeric value with no leading zeroes it can be interpreted as represeting the corresponding 32-bit numeric id. Even though such attribute values cannot be translated, they are still likely to be useful. In the case of a client, the attribute string may be used for local display of ownership. However, in the case in which the server receives such a string, it is less likely to be useful and might be harmful in that its use by the server might undercut the value of the translation to the form name@domain. In order to avoid these negative effects:

- When a numeric value is received in an attribute being set where an RPCSECGSS-based auth flavor is being used, the SETATTR **MUST** return an error (NFS4ERR_BADOWNER).

- When the auth flavor AUTH_SYS is being used, the seerver **MAY** accept user and groups identified by ids using the string numeric form and return them in that form. However, when it does so it still **MUST** return these values, in the name@domain form to those using other auth flavors.

[Author Aside, Including List (Items #52a, #60a)]: Something needs to be done about the failure of the corresponding text in RFC8881 to deal approriately with root@domain and nobody@domain, either to exclude these or include them with appropriate explanations. Specifically:

- With regard to the case of "nobody", the primary issue is that "nobody@domain" is not addressed, although "nobody", which should not occur, is.

  A further difficulty is the use of the term "anonymous" with the intention to include unidentified and unauthentcated users as well as those which are not approrpriately authenticated.

- With regard to the case of "root", the issue is not mentioned at all, leaving it unclear as to whether this sort of special handling is gone, or is dependent now on the name "root" or the mumeric id zero despite the geneneral deprecation of numeric ids.

  The possibility of support for multiple domains, creates additional complexity whih needs to be addressed.

[Consensus Needed (Item #60a)]: An owner string of the form "nobody@domain" may be used to designate an anonymous or unauthenticated user, which will be associated with a file created by a security principal whose identity is not determinable or cannot be mapped through normal means to the Owner attribute. Users and implementations of NFSv4.1 should avoid the use of the string "nobody" to identify an actual user.

[Consensus Needed (Item #52a)]: Certain id strings that map to known numeric values such as zero **MAY** be assigned special privileges with regard to operation authorization, allowing operations to be authoized that POSIX or ACL-based authentication might disallow. The granting of such privileges **MUST NOT** be based on the user name (e.g. "root"). Instead, the server **MUST** only use the principal id or information returned by a secure id mapping facility in assigning such privileges, which **MAY** be assigned differently based on the reliability of the authentication method used. When multiple domains are supported, the privilege assignments might be diffeent for diffeent domains, but the mapping of ids into a value used internally by the file systen is not to be considered in deciding about granting such privileges.

5.2. Table of Authorization-related Attributes

The list of authorization-related attributes appears in Table 1.

The meaning of the columns of the table are:

Name: The name of the attribute.

Id: The number assigned to the attribute. In the event of conflicts between the assigned number and minor version specification documents, the former is authoritative, but conflicts should be resolved with Errata to this document and/or the minor version specifcation document. See [errata] for the Errata process.

Data Type: The XDR data type of the attribute.

Acc: Access allowed to the attribute. R means read-only (GETATTR may retrieve, SETATTR may not set). W means write-only (SETATTR may set, GETATTR may not retrieve). R W means read/write (GETATTR may retrieve, SETATTR may set).

Defined in: The section of this specification that describes the attribute.

Name	Id	Data Type	Acc	Defined in:
acl	12	nfsace4<>	R W	ACL document
aclsupport	13	uint32_t	R	ACL document
dacl	58	nfsace4<>	R W	ACL dicument
mode	33	uint32_t	R W	Section 5.3.1
mode_set_masked	74	mode_masked4	__W	Section 5.3.2
owner	36	utf8str_mixed	R W	Section 5.3.3
owner_group	37	utf8str_mixed	R W	Section 5.3.4
sacl	59	nfsace4<>	R W	ACL document
sec_label	80	sec_label4	R W	Section 5.5.1

Table 1

5.3. POSIX-oriented Authorization-related Attributes

[Consensus Needed (Item #57d)]: The **REQUIRED** attributes Owner, Owning_group, and Mode enable use of a POSIX-based authorization model, as described in Section 8.3. Given that all of these attributes **MUST** be supported, this authorization model is always available.

These attributes are also of use when ACL-based authorization is in effect.

- *The values of the Owner and Owner_group attributes affect the interpretation of the special "who" values "@OWNER" AND "@GROUP">

- *Changes to the acl and dacl attributes will often result in corresponding changes to the mode attribute.

- *Setting the mode attribute will often override the values in acl and dacl attributes, even if tere has been no changes to the authorzation-related bits.

  The set_mode_masked attribute can be set when changing bits in the mode, if the authorization-related bits are not to be changed and there is a need not to update authorization-related ACEs.

5.3.1. The Mode Attribute (v4.0)

[Consensus needed (Item #6a)]: This field is limited to twelve bits, of which only the low-order ten bits are authorization-related. Of the lowest-order nine bit, each set of three bits controls the actions authorized for the file object by a particular set of users. Proceding from higher-order bits to lower, these sets are:

- *The user that is owner of the file.

- *Any user within the owning user group other than the owner of the file.

- *All other users.

Once the appropriate set is determned, based on the principal making the request, the corrsponding three bits determine operation authorization as follows:

- *Authorization for reading of data from the file is controlled by the highest-order bits, as is reading the contents from a directory.

- *Authorization for writing of data to the file is controlled by the second-highest-order bit, as is modifying the contents of a

directory using REMOVE, RENAME, CREATE and OPEN of file specfiying creation of a new file.

*Authorization for read of data from the file for the purposes of executing it as code (eiter via loading/execution of machine instructions or by interpretation of scripts is controlled by the lowest-order bit. Similarly, in the case of a directory, this bit controls the searching of the directory to open the file, do a LOOKUP, or do ny other operation that involves searchin a direcory for a specific name

The specfication of the individulal mode bits appears below:

```
const MODE4_SUID = 0x800;  /* set user id on execution */
const MODE4_SGID = 0x400;  /* set group id on execution */
const MODE4_SVTX = 0x200;  /* save text even after use */
const MODE4_RUSR = 0x100;  /* read permission: owner */
const MODE4_WUSR = 0x080;  /* write permission: owner */
const MODE4_XUSR = 0x040;  /* execute permission: owner */
const MODE4_RGRP = 0x020;  /* read permission: group */
const MODE4_WGRP = 0x010;  /* write permission: group */
const MODE4_XGRP = 0x008;  /* execute permission: group */
const MODE4_ROTH = 0x004;  /* read permission: other */
const MODE4_WOTH = 0x002;  /* write permission: other */
const MODE4_XOTH = 0x001;  /* execute permission: other */
```

Bits MODE4_RUSR, MODE4_WUSR, and MODE4_XUSR apply to the principal identified by the owner attribute. Bits MODE4_RGRP, MODE4_WGRP, and MODE4_XGRP apply to principals belonging to the group identified in the owner_group attribute but who are not identified by the owner attribute. Bits MODE4_ROTH, MODE4_WOTH, and MODE4_XOTH apply to any principal that does not match that in the owner attribute and does not belong to a group matching that of the owner_group attribute. These nine bits are used in providing authorization information.

[Previous Treatment]: The bits MODE4_SUID, MODE4_SGID, and MODE4_SVTX do not provide authorization information and do not affect server behavior. Instead, they are acted on by the client just as they would be for corresponding mode bits obtained from local file systems.

[Consensus needed (Item #6a)]: For objects which are not directories, the bits MODE4_SUID, MODE4_SGID, and MODE4_SVTX do not provide authorization information and do not affect server behavior. Instead, they are acted on by the client just as they would be for corresponding mode bits obtained from local file systems.

[Consensus needed (Item #6a)]: For directories, the bits MODE4_SUID and MODE4_SGID, do not provide authorization information and do not affect server behavior. Instead, they are acted on by the client just

as they would be for corresponding mode bits obtained from local file systems. The mode bit MODE_SVTX does have an authorization-related role as described later in this section

[Consensus Needed, Including List (Item #6a]): When handling RENAME and REMOVE operations the check for authorization depends on the setting of MODE_SVTX for the directory.

- When MODE_SVTX is not set on the directory, authorization requires write permission on both the file being renamed and the source directory.

- When MODE_SVTX is set on the directory, authorization requires, in addition, that the requesting principal be the owner of the file to be renamed or removed.

[Consensus needed (Item #6a)]: It needs to be noted that this approach is similar to the ACL-based approach documented in Section 5.2.9 of [I-D.dnoveck-nfsv4-acls]. However there are some semantic differences whose motivation remains unclear and the specification does not mention RENAME, as it needs to.

[Author Aside]: Bringing the above into more alignment with the ACL-based semantics is certainly desirable but the necessary work has not been done yet. For tracking purposes, that realignment will be considered as Consensus Item #20.

Bits within a mode other than those specified above are not defined by this protocol. A server **MUST NOT** return bits other than those defined above in a GETATTR or READDIR operation, and it **MUST** return NFS4ERR_INVAL if bits other than those defined above are set in a SETATTR, CREATE, OPEN, VERIFY, or NVERIFY operation.

[Consensus Needed (Item #21b)]: As will be seen in Sections 8.3, 8.4, and 8.7 of [I-D.dnoveck-nfsv4-acls]. many straightforward ways of dealing with mode that work well with forward-slope modes need adjustment to properly deal with reverse-slope modes, as defined in Section 4.1

5.3.2. The Mode_set_masked Attribute (v4.1)

The mode_set_masked attribute is an **OPTIONAL** write-only attribute that allows individual bits in the mode attribute to be set or reset, without changing others. It allows, for example, the bits MODE4_SUID, MODE4_SGID, and MODE4_SVTX to be modified while leaving unmodified any of the nine low-order mode bits devoted to permissions.

When this attribute is not supported, clients' only option in setting the mode attribute is to set all bits of that attribute using

SETATTR, even if the motivation is only to modify bits which are not authorization-related.

This has the unfortunate result that some the effects of any assciated ACL attribute are negated as it is assumed that by changing the mode, the intention is to override all ACL-related authorization, whereever its effect is different from that specified by the mode alone.

5.3.3. The Owner Attribute (v4.0)

Defines the user who owns the file, which affects which set of three bits from the mode attribute controls the bits controlling authorizetio.

The interpretation of this string value is described in Section 5.1

5.3.4. The Owner_group Attribute (v4.0)

Defines the group that wns the file, which affects which set of three bits from the mode attribute controls the bits controlling authorizetio.

The interpretation of this string value is described in Section 5.1

5.3.5. Issues with Named Attribute Directories

[Previous Treatment (Item #66a]): Note that the hidden directory returned by OPENATTR is a convenience for protocol processing. The client should not make any assumptions about the server's implementation of named attributes and whether or not the underlying file system at the server has a named attribute directory. Therefore, operations such as SETATTR and GETATTR on the named attribute directory are undefined.

[Author Aside (Item #66a), Through end of bulleted list]: Despite the dubious logic of the preceding pargraph, it is probably too late to significantly revise this decision, which first appeared in [RFC3530]. Nevertheless, it might be helpful to understand how this decision was made, why it was not caught in document review, and why it has not been addressed in the many years since it was first incorporated in NFSv4. The following factors should be noted:

* Whether the underlying file system at the server has a named directtory attribute or not is an implementation matter and out of scope for this sort of specfication.

  Despite this fact, a named attribute directory is required by the specfication because LOOKUP and OPEN operations are done on it and

that remains the case whether it is dismissively characterized as a "convenience" or not.

It is a fundamental mistake to conflate the protocol choice made here, to deprive these directories of the need/ability to store attributes, with the basic architectural point that it is up to the server to decide on the means by which the protool's requirements are met.

*While it is hard to conceive of a situation in which the above mistake led to this decision, it is much more likely that this paragraph was assembled at the last minute to try to justify a decision already arrived at, for other reasons.

It could well be that this decision was made because the working group was not prepared to resolve issues that might arise from supporting attributes on these directories in the time available or feared that many proposed implementations of named attributes might be unwilling to provide appropriate support for named attributes if support for attributes of named attribute directories were included.

It was probably felt that honestly citing the real difficulties cited above might result in IESG criticism from those prefacing their remarks by disclainimg NFS expertise, while feeling that, given the complexity of the explanation actually provided, it was more protected from such non-expert criticism.

*The difficulties cited above might not have been sufficient to foreclose these attributes if proper notice was taken of the need for authorization-related attributes to support the authorization of actions related to named attributes.

As things turned out, this need was not appropreately recognized. This lack of recognition arose mostly because it had never been important previously, given that POSIX semantics could relied upon without separate effort from NFSv4. It was not noticed that with the non-POSIX semantics implied by named attributes, that situation had fundamentally changed.

*Given that the lack of support for such attributes might be more approprotaely be dealt with by deferral without essentially foreclosing future implementation of such a feature, it needs to be understood why such deferral was never considred.

Given this context, it is worth noting that, at the time this choice was made, ther was no extension model for NFSv4 except minor versioning and that made what would be, in retrospect, the best choice, i.e deferrring the named attribute feature to a later time, inconceivable.

[Consensus Needed (Item #66a]): Since there is no way to set the attributess associated with a newly-created named attribuute directory and because operations such as SETATTR and GETATTR are undefined when applied to named attribute directories, normal approaches to authorization of oprations on named attribute directories are not available. As a result, the necessary authorization semantics need to be specified somehow, most likely by deriving values to be used for Mode, Owner, and Owner_group attributes for the named directory from the corresponding attributes for the base object.

[Author Aside (Item #66a), Through end of section}: For the most part, the NFSv4 specfifications have avoided the need to describe authorization semantics, by relying on the POSIX definition and making it the responsibility of the protocol, as it should be, to duplicate local semantics when used remotely.

In the case of the **OPTIONAL** named attributes feature which is ouside the scope of POSIX, that approach is no longer viable and needs to supplemented by a semantic description. In formulating such description, the following issued need to be addressed:

* We don't have information about authorization semantics for implememententations of this feature or even if such implementations exist.

  This applies to both potential implementations that do and do not support ACLs.

  The lack of previous focused discussion of this issue strongly suggests that implementations are either non-existent, or pay no serious attention to authorizaton semantics.

* The privilege structure provided by the **REQUIRED** attributes Mode, Owner, and Owner_group seems like it could be adapted to control access to the named attribute, but there are some troublesome gaps.

  OPENATTR provides no way to set the attributes for the mamed attribute directory it creates. Furthermore, GETTATTR and SETATTR on these directories are undefined, according to [RFC8881]) and most likely will remain so.

* There are corresponding issues relating ACL-based authorization that are part of the associated Consensus Item #100 which is discussed within [I-D.dnoveck-nfsv4-acls].

  There will need to be extensive co-ordination in addressing these two related issues.

5.3.6. Posix Authorization for Named Attribute Directories

[Consensus Needed (Item #66b), through end of section]: Because named attribute directories do not have attributes, the values to be used in place of those those attributes in order to support authorization decisions for operations on named attribute directoies are as follows:

- *The value to be used in place of mode attribute is based on the value of the mode attribute of the base object with the following modifications:

  For each of owner, group, and others, the read and execute permissions are set iff either the read or execute bit is set in the corresponding set of three bits in the mode attribute of the base object.

- *The value to be used in place of owner attribute is identical to the owner attribute for the base object.

- *The value to be used in place of owner_group attribute is identical to the owner_group attribute for the base object.

5.4. ACL-based Authorization-related Attributes

The following ACL-related attributes (all **OPTIONAL** are described in detail in the new document devoted to ACL handling [I-D.dnoveck-nfsv4-acls].

- *The per-fs Aclsupport attribute.

- *The per-fs Aclfeatures attribute being added as an extension to NFSv4.2..

- *The per-object Acl attribute.

- *The per-object Dacl attribute added in NFSv4.1..

- *The per-object Sacl attribute added in NFSv4.1

5.5. Authorization-related Attributes for MAC

5.5.1. The Seclabel Attribute (v4.2)

This opaque attribute provides authorization-related information to support Mandatory Access Control. Given the lack of specfic docuentation about the contents and the uncertainty regarding idenification of the actors making the requests to be authorized, client-server interoperability is not available and any any authorization decision are the responsibility of the client itself.

[Consensus needed (Item #58b)]: Although this was intended to be an OPTIONAL attribute, it is now more appropriate to describe it is an under-specfied one, and thus essentially experimental, although still formally. **OPTIONAL**. This is due to the absence of specifications for the content of the attribute, or descriptions of the way in which it is to be used to govern authorization.

6. Introduction to ACLs

The ACL-related atributes Acl, Sacl, and Dacl, introduced above in Section 5.4 and in more detail in the ACL specification document [I-D.dnoveck-nfsv4-acls], each contain an array of Access Control Entries. These ACEs define the operations that are authorized for paticular users or user groups

ACL-related attributes and the structure of ACEs are described in detail in a companion document [I-D.dnoveck-nfsv4-acls].

6.1. Previous Treatment of ACLs

The description of ACL-related attributes and ACEs that appeared in previous documents embodied a flawed appoach to protocol description that makes in unsuitable for use as a basis for a descrption intended to appear and in a standards-track document. As the goal of this effort is to produce a helpful standards-track document, these item will be described using a new approach, as described in Section 6.2.

This same approach was used in a number of documents published as Proposed Standards, despite its flaws. These included RFCs now obsoleted ([RFC3010], [RFC3530], and [RFC5661]) and a number RFCs to be obsoleted when this document is published as an RFC ([RFC7530] and [RFC8881]). Although some changes were made in the transition between minor versions, the essence of the approach remained the same.

The goal that led to this flawed approach remains as it was. There is need to support an extended ACL model, with a fine-grained permission model and other helpful extensions togther with providing support for a more limited ACL model with a more direct connection to POSIX semantics, very similar to that defined in the withdrawn POSIX ACL draft implemented by a number of file sytems implemented on UNIX systems.

The approach actually taken to this need had the folloing elements:

* The definition of the Acl attribute was based on the extended ACL model, rather than simpler, more POSIX-oriented core.

* To allow servers implement the UNIX ACL subset to be considered compliant, the specification was written providing an unacceptable

degree of leeway for the server to implement many elements in pretty much whatever way it chose.

This approach allowed UNIX ACls, NFSv4 ACLs and hybrids of the two to be considerd compliant, as well as many many unantcipated vaiants. In any case, the client had no way of determining the ACL semantics being implemented.

6.2. New Approach to Treatment of ACLs

The new approach, which will be explained in more detail in the companion document [I-D.dnoveck-nfsv4-acls] takes a very different approach, with the following chacteristics:

- The UNIX ACL subset is the basis of the canonical description which can asssume is available, if the Acl attribute is supported.

- Additions to this core are treated as **OPTIONAL** extensions. each of which might or might not be suported by a server implementing the Acl atribute.

- A new per-fs **OPTIONAL** attribute is defined to allow clients to determine which ACL model extensions are supported, if any.

This new approach is embodied in the companion document [I-D.dnoveck-nfsv4-acls]. As a result, that document has two functions:

- Together with this document it defines security for all minor versions, updating [RFC7530] and, together with the other rfc5661bis documents, onsoleteing [RFC8881]. This applies to all NFSv4 minor versions.

- Defining an extension that allows clients to determine the ACL extensions supported by any given file system. This applies to nFSv4.2 and subsequent minor versions.

7. Authorization in General

There are three distinct methods of checking whether NFSv4 requests are authorized:

- The most important methods of authorization is used to effect user-based file access control, as described in Section 8. These methods are often termed "Discretionary access control" because they rely on attributes set by particular users, to control acceptable file access.

  This requires the identification of the user making the request. Because of the central role of such access control in providing

NFSv4 security, server implementations **SHOULD NOT** use such identifications when they are not authenticated. In this context, valid reasons to do otherwise are limited to the compatibility and maturity issues discussed in Section 18.1.4

*NFSv4.2, via the labelled NFS feature, provides an additional potential requirement for request authorization. The labelled NFS provides "Mandatory access control" not under the control of individual users.

For reasons made clear in Section 11, there is no realistic possibility of the server using the data defined by existing specifications of this feature to effect request authorization. While it is possible for clients to provide this authorization, the lack of detailed specifications makes it impossible to determine the nature of the identification used and whether it can appropriately be described as "authentication".

*Since undesired changes to server-maintained locking state (and, for NFSv4.1, session state) can result in denial of service attacks (see Section 18.4.7), server implementations **SHOULD** take steps to prevent unauthorized state changes. This can be done by implementing the state authorization restrictions discussed in Section 12. Because these restrictions apply on a per-peer basis rather than being affected by the identity of the user making the request, it is better to consider them as part of "Mandatory access control".

8. User-based File Access Authorization

8.1. Attributes for User-based File Access Authorization

NFSv4.1 provides for multiple authentication models, controlled by the support for particular recommended attributes implemented by the server, as discussed below:

*Consensus Needed (Item #57e)]: The **REQUIRED** attributes owner, owning_group, and mode enable use of a POSIX-based authorization model, as described in Section 8.3. Since these attributes are always supported, this authorization model is aways available.

*[Consensus Needed (Item #17a)]: The acl attribute (or the attribute dacl in NFSv4.1) can provide an ACL-based authorization model as described in Section 8.4 as long as support for ALLOW and DENY ACEs is provided.

[Consensus Needed (Items #17a)]: When some of these ACE types are not supported, this authorization modelis, to a degree, incompatible with the mode-based one, since there are some modes that cannot be represented as a corresponding NFSv4 ACL, when

using only a single ACE type. See Sections 8.2 and 8.7 of [I-D.dnoveck-nfsv4-acls] for details

8.2. Handling of Multiple Parallel File Access Authorization Models

NFSv4 ACLs and modes represent two well-established models for specifying user-based file access permissions. NFSv4 provides support for either or both depending on the attributes supported by the server and, in cases in which both NFSv4 ACLs and the mode attribute are supported, the actual attributes set for a particular object.

- *[Consensus Needed (item #18b)]: When the attributes mode, owner, owner group are all supported, the posix-based authorization model, described in Section 8.3 can be used.

- *[Consensus Needed (Items #17b, #18b)]: When the acl (or dacl) attribute is supported together with both of the ACE types ALLOW and DENY, the acl based authorization model, described in Section 8.4 can be used as long as the attributes owner and owner_group are also supported.

[Consensus Needed (item #18b)]: While formally recommended (essentially **OPTIONAL**) attributes, it appears that the owner and owner_group attributes need to be available to support any file access authorization model. As a result, this document will not discuss the possibility of servers that do not support both of these attributes and clients have no need to support such servers.

When both authorization models can be used, there are difficulties that can arise because the ACL-based model provides finer-grained access control than the POSIX model. The ways of dealing with these difficulties appear later in this section while more detail on the appropriate handling of this situation, which might depend on the minor version used, appears in Section 10.

The following describe NFSv4's handling in supporting multiple authorization models for file access.

- *If a server supports the mode attribute, it needs to provide the appropriate POSIX semantics if no ACL-based attributes have ever been assigned to object. These semantics include the restriction of the ability to modify the mode, owner and owner-group to the current owner of the file.

- *If a server supports ACL attributes, it needs to provide NFSv4 ACL semantics as described in this document for all objects for which the ACL attributes have actually been set. This includes the ACL-based restrictions on the authorization to modify the mode, owner and owner_group attributes.

*On servers that support the mode attribute, if ACL attributes have never been set on an object, via inheritance or explicitly, the behavior is to be the behavior mandated by POSIX, including tthose provisions that restrict the setting of authorization-related attributes.

*On servers that support the mode attribute, if the ACL attributes have been previously set on an object, either explicitly or via inheritance:

  -[Previous Treatment]: Setting only the mode attribute should effectively control the traditional UNIX-like permissions of read, write, and execute on owner, owner_group, and other.

  [Author Aside]: It isn't really clear what the above paragraph means, especially as it governs the handling of aces designating specific users and groups which are not the owner and have no overlap with the owning group

  {Consensus Needed (Item #19a)]: Setting only the mode attribute, will result in the access of the file being controlled just it would be if the existing acl did not exist, with file access decisions as to read made in accordance with the mode set. The ALLOW and DENY aces in the ACL will reflect the modified security although there is no need to modify AUDIT and ALARM aces or mask bits not affected by the mode bits, such as SYNCHRONIZE.

  [Author Aside]: the above may need to modified to reflect the resolution of Consensus Item #??.

  -[Previous Treatment]: Setting only the mode attribute should provide reasonable security. For example, setting a mode of 000 should be enough to ensure that future OPEN operations for OPEN4_SHARE_ACCESS_READ or OPEN4_SHARE_ACCESS_WRITE by any principal fail, regardless of a previously existing or inherited ACL.

  [Author Aside]: We need to get rid of or provide some some replacement for the subjective first sentence. While the specific example give is unexceptionable, it raises questions in other cases as to what would constitutes "reasonable semantics". While the resolution of such questions would be subject to dispute, the author believes that consensus item #19a deals with the matter adequately. As a result he proposes, that the that this bullet be removed and that the second-level list be collapsed to a single paragraph.

*Although [RFC7530] and [RFC8881] present different descriptions of the specific semantic requirements relating to the interaction of

mode and ACL attributes, the difference are quite small, with the most important ones deriving from the absence of the set_mode_masked attribute. The unified treatment in Section 10 will indicate where version-specific differences exist.

8.3. Posix Authorization Model

Uses the **REQUIRED** attributes Mode, Owner, amd Owner_group to authorize actions applying to files and directories. Aside from the conversion of Owner and Owner_group to the forms of strings, this mirrors authorization in NFSv3, in being based on semantics defined by POSIX, originally used to control local finle access.

There are some potential differences to be notesd:

*The use of "owner-override" semanctics is allowed. This vaiant of autorization semantics was necessary in earlier NFS vesions, to allow continued access to open files even when the mode had been changed subequent to the open.

  Even though it is no longer necessary because NFSv4 is a stateful protocol, it is still allowed.

*[Consensus needed (Item #6b)]: The bit MODE_SVTX within the Mode attrbute on directories affects the authorization of REMOVE and RENAME.

8.4. ACL-based Authorization Model

While the relevant details appear in [I-D.dnoveck-nfsv4-acls], this model can be helpfully smmarize as follows:

The use of ACLs, via setting the **OPTIONAL** attributes Acl and Dacl, allows a more flexible approch to authorization in that the individual entries (ACEs) within the ACL can assign different privileges to different users or groups of users.

The ACEs within an ACL are examined in sequence with those designating the user requesting or a group to which that user belongs affecting the authorization decision to see if a needed authorzation for the requested operation is granted, or, depending the ACE type, denied.

Once all requested autnorization are granted, the operation is authorized and can proceed. If any requested authorzation is denied, the operation is not authotized. In either case, this terminates the processing of ACEs.

9. Common Considerations for Both File access Models

[Author Aside, Including List]: The subsections within this section are derived from Section 6.3 of 8881, entitled "Common Methods. However, its content is different because it has been rewritten to deal with issues common to both file access models, which now appears to have not been the original intention. Nevertheless, the following changes have been made:

*The section "Server Considerations" has been revised to deal with both the mode and acl attributes, since the points being made apply, in almost all cases, to both attributes.

*The section "Client Considerations" has been heavily revised, since what had been there did not make any sense to me.

*The section "Computing a Mode Attribute from an ACL" has been moved to Section 8.3 of [I-D.dnoveck-nfsv4-acls] since it deals with the co-ordination of the POSIX and ACL authorization models.

9.1. Handling of ACCESS and OPEN Operations

The primary means by which client-side users find out whether particular operations are authorized is to attempt those operations and have them executed successfully or rejected with an error. However, clients can use the ACCESS operation to determine in advance whether operations are permitted done without actually attempting those operations.

The use of the ACCESS operation is of particular importance when ACLs exist. This is due in part to the complexity of the ACLs but also derives from cases in which the client is incapable of determining whether a specific ACE applies to a particular request, as when some of the special who-field values are used in an ACE.

An additional difficulty precluding the client performing its own interpretation of an ACL concerns the existing specifications' lack of clear requirements for server support, making it difficult for the client to determine how a server might choose to behave in some cases. Even if this lack of specificity were remedied by clarifying the specification servers would still exist that reflected the previously valid scope of acceptable server behavior. As a result, it is impossible for the client to predict determine the choices a server might make without using ACCESS.

Similarly, the OPEN operation can determine acces rights in advance of actual access. There are two differences from ACCESS that clients need to be aware of, since they might require use of ACCESS in

addition to OPEN or make the results of OPEN incompatible with the results of attempting IO operations.

* The permission model of OPEN is coarser-grained than that provided for by ACCESS or the ACE masks defined as part of the definition of NFSv4 ACLs.

  When reading files, the fact that OPEN treats requests to open a file to read it and to execute it together means that the client will need to do an ACCESS operation to determine whether particular reads are to be allowed.

  When writing files, the fact OPEN that does not distinguish between WRITEs which extend the file and those that modify bytes already written means that the client will need to get the permission information using ACCESS or be prepared to have some set of WRITES on a file open for Write rejected as unauthorized.

* [Consensus needed, Entire Bulleted Item (Item #23a)]: Just as with ACCESS, the granting of permission does not foreclose subsequent permission changes, there are good reasons for servers to provide ways of allowing IO allowed at OPEN time to continue even in the face of permission changes that would normally be expected to make the IO operation invalid. These reasons derive from the fact that local operations work this way and that it is is often necessary for requests over NFSv4 to behave just as they would if done locally. As a result, it is desirable for servers to provide this expected behavior in some way since application programs have come to depend on it.

  One way of handling this situation is described in Section 9.2. It deals with the most common of these situations in which the owner of the is both writing a file and seeking to make it subsequently unwritable, and while it does not deal with all such situations, it has proven satisfactory for many NFS protocols over a long time period

  Another way of dealing with this situation involves the server explicitly using the stateid to reference a particular open made by a user, and avoiding reverification of access for READs and WRITEs made by that user since that verification was made at OPEN time. To do this safely, the server needs to have authenticated principals and client peers and, in order to prevent man-in-middle attacks, it s necessary for all connections on which stateids are sent to provide encryption.

9.2. Server Considerations

The server uses the mode attribute or the acl attribute applying the algorithm described in Section 7 of [I-D.dnoveck-nfsv4-acls] to determine whether an ACL allows access to an object.

[Author Aside, Including List]: The list previously in this section (now described as "Previous Treatment" combines two related issues in a way which obscures the very different security-related consequences of two distinct issues:

- In some cases an operation will be authorized but is not allowed for reasons unrelated to authorization.

  This has no negative effect on security.

- The converse case can have troubling effects on security which are mentioned in this section and discussed in more detail in Section 18

[Author Aside, Including List]: The items in that list have been dealt with as follows:

- The first and sixth items fit under the first (i.e. less troublesome) of these issues. They have have been transferred into an appropriate replacement list.

- The third item is to be deleted since it does not manifest either of these issues. In fact, it refers to the semantics described in Section 5.2. of [I-D.dnoveck-nfsv4-acls].

- The second, fourth and fifth items need to be addressed in a new list dealing with the potentially troublesome issues arising from occasions in which the access semantics previously described are relaxed, for various reasons.

  Included are cases in which previous specifications explicitly allowed this by using the term "**MAY**" and others in which the existence of servers manifesting such behavior was reported, with the implication that clients need to be prepared for such behavior.

[Previous Treatment, Including List (Items #22a, #41a, #52b)]: However, these attributes might not be the sole determiner of access. For example:

- In the case of a file system exported as read-only, the server will deny write access even though an object's file access attributes would grant it.

*Server implementations **MAY** grant ACE4_WRITE_ACL and ACE4_READ_ACL permissions to prevent a situation from arising in which there is no valid way to ever modify the ACL.

*All servers will allow a user the ability to read the data of the file when only the execute permission is granted (e.g., if the ACL denies the user the ACE4_READ_DATA access and allows the user ACE4_EXECUTE, the server will allow the user to read the data of the file).

*Many servers implement "owner-override semantics" in which the owner of the object is allowed to perform accesses that are denied by the ACL or mode bits This may be helpful, for example, to allow users continued access to open files on which the permissions have changed.

*Many servers provide for the existence of a "superuser" that has privileges beyond an ordinary user. The superuser may be able to read or write data or metadata in ways that would not be permitted by the ACL or mode attributes.

*A retention attribute might also block access otherwise allowed by ACLs (see Section 5.13 of [RFC8881]).

[Consensus Needed, Including List (Item #22a)]: It needs to be noted that, even when an operation is authorized, it may be denied for reasons unrelated to authorization. For example:

*In the case of a file system exported as read-only, the server will deny write access even though an object's file access attributes would authorize it.

*A retention attribute might also block access otherwise allowed by ACLs (see Section 5.13 of [RFC8881]).

[Author aside, including List, (Item #22a)]: Unlike other cases in which previous specs have granted permission to the server to expand allowed behavior, in the case of "owner-override semantics", the need for supercession does not arise from security issues but instead derives from:

*The unacceptable implication in the superseded text that the fact that a server chooses to do something, means that the client needs to accept that behavior.

*The lack of any definition of the term "owner-override semantics. Subsequent investigation has led to the conclusion that the

semantics being referred to derive from the following material in [RFC1813]:

> Another problem arises due to the usually stateful open operation. Most operating systems check permission at open time, and then check that the file is open on each read and write request. With stateless servers, the server cannot detect that the file is open and must do permission checking on each read and write call. UNIX client semantics of access permission checking on open can be provided with the ACCESS procedure call in this revision, which allows a client to explicitly check access permissions without resorting to trying the operation. On a local file system, a user can open a file and then change the permissions so that no one is allowed to touch it, but will still be able to write to the file because it is open. On a remote file system, by contrast, the write would fail. To get around this problem, the server's permission checking algorithm should allow the owner of a file to access it regardless of the permission setting. This is needed in a practical NFS version 3 protocol server implementation, but it does depart from correct local file system semantics. This should not affect the return result of access permissions as returned by the ACCESS

As a result, it appears that NFSv4 servers have implemented semantics different from that provided for by POSIX that were defined in order to deal with lack of an open operation in NFsv3 in the context of NFSv4, which does have an open operation.

*The need to make it clear that an alternate approach to the issue of permission change after open is to allow the OPEN's permission check to be dispositive as long as the appropriate security infrastructure is in place to allow client stateids to be trusted.

[Consensus Needed, (Item #22a)]: There are also cases in which the converse issue arises, so that an operation which is not authorized as specified by the mode and ACL attributes is, nevertheless, executed as if it were authorized. Because previous NFSv4 specifications have cited the cases listed below without reference to the security problems that they create, it is necessary to discuss them here to provide clarification of the security implications of following this guidance, which might now be superseded. These cases are listed below and discussed in more detail in Section 18.1.3.

[Consensus Needed, Including List (Item #22a, #41a, #52b)]: In the following list, the treatment used in [RFC8881] is quoted, while the corresponding text in [RFC7530] is essentially identical.

*[RFC8881] contains the following, which is now superseded:

> Server implementations **MAY** grant ACE4_WRITE_ACL and ACE4_READ_ACL permissions to prevent a situation from arising in which there is no valid way to ever modify the ACL.

While, as a practical matter, there do need to be provisions to deal with this issue, the "**MAY**" above is too broad,in that it describes the motivation without any limits providing appropriate restriction on the steps that might be taken to deal with the issue. See Section 18.1.3 for the updated treatment of this issue.

*[RFC8881] contains the following, which is now superseded:

> Many servers implement owner-override semantics in which the owner of the object is allowed to override accesses that are denied by the ACL. This may be helpful, for example, to allow users continued access to open files on which the permissions have changed.

The principal problem with the above statement is the lack of a clear definition of the term "owner-override semantics". Also, it needs to be made clear that the fact that a server manifests a particular behavior does not imply that it is valid according to the protocol specification.

In this case, investigation has led to the conclusion that the semantics being referred to derive from the discussion of NFSv3 semantics appearing Section 4.4 of [RFC1813] and that this handling has been implemented in a number of NFSv4 servers, despite the fact that NFSv4, unlike NFSv3, does have an open operation.

With regard to the second sentence of the quotation above, it is not clear whether it is helpful or hurtful to allow continued access to open files which have become inaccessible due to changes in security and it is not clear that the working group will make a decision on the matter in this document, despite the obvious security implications. In any case, the resolution is unlikely to depend on whether the owner is involved.

Since this divergence from POSIX semantics is unlikely to result in security issues, we can clarify the above by saying that a server **MAY** diverge from POSIX semantics by always allowing READ or WRITE to be done by the file owner but that this divergence **MUST NOT** affect the handling of OPEN and ACCESS. It also needs to be

noted that servers could address the issue of preventing permission change affecting that handling of READs and WRITEs of open files as described in Section 9.1.

*[RFC8881] contains the following, which is now superseded:

> Many servers have the notion of a "superuser" that has privileges beyond an ordinary user. The superuser may be able to read or write data or metadata in ways that would not be permitted by the ACL or mode attributes.

While many (or almost all) systems in which NFSv4 servers are embedded, have provisions for such privileged access to be provided, it does not follow that NFSv4 servers, as such, need to have provision for such access.

Providing such access as part of the NFSv4 protocols, would necessitate a major revision of the semantics of ACL including such troublesome matters as the proper handling of AUDIT and ALARM ACEs in the face of such privileged access.

Because of the effect such unrestricted access might have in facilitating and perpetuating attacks, Section 18.1.3 will the new approach to this issue, while Section 18.4.1, will explain how such access is addressed in the threat analysis.

9.3. Client Considerations

[Previous Treatment]: Clients **SHOULD NOT** do their own access checks based on their interpretation of the ACL, but rather use the OPEN and ACCESS operations to do access checks. This allows the client to act on the results of having the server determine whether or not access is to be granted based on its interpretation of the ACL.

[Author Aside]: With regard to the use of "**SHOULD NOT**" in the paragraph above, it is not clear what might be valid reasons to bypass this recommendation. Perhaps "**MUST NOT**" or "are not advised to" would be more appropriate.

[Consensus Needed (Item #23b)]: Clients are not expected to do their own access checks based on their interpretation of the ACL, but instead use the OPEN and ACCESS operations to do access checks, where this is possible. This allows the client to act on the results of having the server determine whether or not access is to be granted based on its interpretation of the ACL, rather then the client's which might be different. For a full discussion of limitations on the use of ACCESS and appropriate client approaches to deal these limitations, see Section 9.1.

[Previous Treatment]: Clients must be aware of situations in which an object's ACL will define a certain access even though the server will not enforce it. In general, but especially in these situations, the client needs to do its part in the enforcement of access as defined by the ACL.

[Author Aside]: Despite what is said later, the only such case I know of is the use of READ and EXECUTE where the client, but not the server, has any means of distinguishing these. I don't know of any others. If there were, how could ACCESS or OPEN be used to verify access?

[Consensus Needed (Item #23b)]; Clients need to be aware of situations in which an object's ACL will define a certain access even though the server is not in position to enforce it because the server does not have the relevant information, such as knowing whether a READ is for the purpose of executing a file. Because of such situations, the client needs to do be prepared to do its part in the enforcement of access as defined by the ACL.

To do this, the client will send the appropriate ACCESS operation prior to servicing the request of the user or application in order to determine whether the user or application is to be granted the access requested.

[Previous Treatment (Item #24a)]: For examples in which the ACL may define accesses that the server doesn't enforce, see Section 9.2.

[Author Aside]: The sentence above is clearly wrong since that section is about enforcement the server does do. The expectation is that it will be deleted as part of Consensus Item #24a.

10. Combining Authorization Models

The existence of multiple authorization models where ACLs are supported raises a number of issue regarding how these two models are to interact that are dealt with in [I-D.dnoveck-nfsv4-acls]. These include:

*How the value of the mode attribute is to computed from the ACL.

*How the case of the mode and ACL both being set in the same SETATTR are to be dealw with.

*How ACLs are modified in response to changes in the mode attribute.

*How the ACL mode is to be modified when the mode attribute is set.

11. Labelled NFS Authorization Model

The attribute sec_label was intended to enable an authorization model focused on Mandatory Access Contro.

Not much can be said about this feature because the specification, in the interest of flexibility, has left important features undefined in order to allow future extension. As a result, we have something that is a framework to allow Mandatory Access Control rather than one to provide it. In particular,

- *The sec_label attribute, which provides the objects label has no existing specification.
- *There is no specification of the format of the subject labels or way to authenticate them.
- *As a result, all authorization takes place on the client, and the server simply accepts the client's determination.

This arrangements shares important similarities with AUTH_SYS. As such it makes sense:

- *To require/recommend that an encrypted connection be used.
- *To require/recommend that client and server peers mutually authenticate as part of connection establishment.
- *That work be devoted to providing a replacement without the above issues.

12. State Modification Authorization

Modification of locking and session state data are not be done by a client other than the one that created the lock. For this form of authorization, the server needs to identify and authenticate client peers rather than client users.

Such authentication is not directly provided by any RPC auth flavor. However, RPC can, when suitably configured, provide this authentication on a per-connection basis.

NFSv4.1 defines a number of ways to provide appropriate authorization facilities. These will not be discussed in detail here but the following points need to be noted:

- *NFSv4.1 defines the MACHCRED mechanism which uses the RPCSEC_GSS infrastructure to provide authentication of the clients peer. However, this is of no value when AUTH_SYS is being used.

*NFSv4.1 also defines the SSV mechanism which uses the RPCSEC_GSS infrastructure to enable it to be reliably determined whether two different client connections are connected to the same client. It is unclear whether the word "authentication" is appropriate in this case. As with MACHCRED, this is of no value when AUTH_SYS is being used.

*Because of the lack of support for AUTH_SYS and for NFSv4.0, it is quite desirable for clients to use and for servers to require the use of client-peer authentication as part of connection establishment.

When unauthenticated clients are allowed, their state is exposed to unwanted modification as part of disruption or denial-of-service attacks. As a result, the potential burdens of such attacks are felt principally by clients who choose not to provide such authentication.

13. Other Uses of Access Control Lists

Whether the Acl or Sacl attributes are used, AUDIT and ALARM ACEs provide security-related facilities separate from the the file access authorization provided by ALLOW and DENY ACEs

*AUDIT ACEs provide a means to audit attempts to access a specified file by specified sets of principals.

*ALARM ACEs provide a means to draw special attention to attempts to access specified files by specified sets of principals.

Whichever attribute is used to store the relevant ACEs, only these ACE types are processed while other types are ignored. While the processsing of successive ACEs is simillar to that described in Section 8.4, these two scans cannot be combined. The authoriation-related scan needs to be done first and the result of that checks, success or failue is needed to govern the deletion (or not) of AUDIT and ALARM ACEs.

14. Identification and Authentication

Various objects and subjects need to be identified for a protocol to function. For it to be secure, many of these need to be authenticated so that incorrect identification is not the basis for attacks.

14.1. Identification vs. Authentication

It is necessary to be clear about this distinction which has been obscured in the past, by the use of the term "RPC Authentication Flavor" in connection with situation in which identification without authentication occurred or in which there was neither identification

nor authentication involved. As a result, we will use the term "Auth Flavors" instead

14.2. Items to be Identified

Some identifier are not security-relevant and can used be used without authentication, given that, in the authorization decision, the object acted upon needs only to be properly identified

- *File names are of this type.

  Unlike the case for some other protocols, confusion of names that result from internationalization issues, while an annoyance, are not relevant to security. If the confusion between LATIN CAPITAL LETTER O and CYRILLIC CAPITAL LETTER O, results in the wrong file being accessed, the mechanisms described in Section 8 prevent inappropriate access being granted.

  Despite the above, it is desirable if file names together with similar are not transferred in the clear as the information exposed may give attackers useful information helpful in planning and executing attacks.

- *The case of file handles is similar.

Identifiers that refer to state shared between client and server can be the basis of disruption attacks since clients and server necessarily assume that neither side will change the state corpus without appropriate notice.

While these identifiers do not need to be authenticated, they are associated with higher-level entities for which change of the state represented by those entities is subject to peer authentication.

- *Unexpected closure of stateids or changes in state sequence values can disrupt client access as no clients have provision to deal with this source of interference.

  While encryption may make it more difficult to execute such attacks attackers can often guess stateid's since server generally not randomize them.

- *Similarly, modification to NFSv4.1 session state information can result in confusion if an attacker changes the slot sequence by assuring spurious requests. Even if the request is rejected, the slot sequence is changed and clients may a difficult time getting back in sync with the server.

While encryption may make it more difficult to execute such attacks attackers can often guess slot id's and obtain sessinid's since server generally do not randomize them.

*

it is necessary that modification of the higher-levell entities be restricted to the client that created them.

*For NFSv4.0, the relevant entity is the clientid.

*for NFSv4.1, the relevant entity is the sessionid.

Identifiers describing the issuer of the request, whether in numeric or string form always require authentication.

14.3. Authentication Provided by specific RPC Auth Flavors

Different auth flavors differ quite considerably, as discussed below;

*When AUTH_NONE is used, the user making the request is neither authenticated nor identified to the server.

Also, the server is not authenticated to the client and has no way to determine whether the server it is communicating with is an imposter.

*When AUTH_SYS is used, the user making is the request identified but there no authentication of that identification.

As in the previous case, the server is not authenticated to the client and has no way to determine whether the server it is communicating with is an imposter.

*When RPCSEC_GSS is used, the user making the request is authenticated as is the server peer responding.

14.4. Authentication Provided by other RPC Security Services

Depending on the connection type used, RPC may provide additional means of authentication. In contrast with the case of RPC auth flavors, any authentication happens once, at connection establishment, rather than on each RPC request. As a result, it is the client and server peers, rather than individual users that are authenticated.

*For many types of connections such as those created TCP without RPC-with-TLS and RPC-over-RDMA version 1, there is no provision for peer authentication.

As a result use of AUTH_SYS using such connections is inherently problematic.

*Some connection types provide for the possibility of mutual peer authentication. These currently include only those established by RPC-with-TLS. However, given the value of peer authentication, there is reason to believe further means of providing such services will be defined.

15. Security of Data in Flight

15.1. Data Security Provided by Services Associated with Auth Flavors

The only auth flavor providing these facilities is RPCSEC_GSS. When this auth flavor is used, data security can be negotiated between client and server as described in Section 16. However, when data security is provided for the connection level, as described in Section 15.2, the negotiation of privacy and integrity support, provided by the auth flavor, is unnecessary,

Other auth flavors, such as AUTH_SYS and AUTH_NONE have no such data security facilities. When these auth flavors are used, the only data security is provided on a per-connection basis.

15.2. Data Security Provided for a Connection by RPC

RPC, in many case, provide data security for all transactions performed on a connection, eliminating the need for that security to be provided or negotiated by the selection of particular auth flavors, mechanisms, or auth-flavor-associated services.

16. Security Negotiation

[Author Aside]: All unannotated paragraphs in this section are considered part of Consensus Item #32a.

Because of the availability of security-related services associated with the transport layer, the security negotiation process needs to be enhanced so that the server can indicate the connection-level services needed. Without the addition of pseudo-flavors defining connection-level services needed, use of SECINFO would be limited to selection of a particular auth flavor with connection characteristics selected by a process of trial and error.

When the SECINFO response does not indicate the connection chacteristics needed, either because it does not contain any of the pseudoflavors described below or because the only such pseudo-flavor is PFL_ANY, the client only knows the available auth flavors together with the auth-flavor-associated services and needs to use trial and

error to determines the server requirments for connection-provided services, as described below:

- For each SECINFO entry the client attempts to make a connection with whatever connection characteristics that its (i.e. the client's) security policy requires. When that connection is successfully established, it can be used to access the current portion of the server, as indicated by SECINFO or SECINFO_NONAME.

  In many cases, this will be a connection without rpc-with-tls, while in other cases the client will require rpc-with-tls to be assured of encryption or of entryption with mutual peer authentication.

- If that connection cannot be established, because the server does no provide support for the requested characteristics, then the client moves on to the next auth-flavor entry. If there are no further entry, this portion ofthe server namespace cannot be accessed.

- If the connection cannot be established because it is rejected by the server because of its security policy, the client takes note and proceeds to request addutional connection-level services on the next connection request.

  For example, if the client attempts an RPC connection of the server's security policy does not allow that, the connection rejection will indicate to the client that use of rpc-with-tls is required. If the server's policy allows non-tls connection but restricts the auth-flavors that may use such connections, the requests made with those flavors may be rejected and the slient will also try to establish an rpc-with-tls connection.

  Similarly, when an rpc-tls-connection is attempted without mutual peer authentication, its rejection will resut in the client attempting to establish a connection with mutual peer authentication.

  When the client and server are unabble to estanlish a mutually accptable connection to use with a specific auth-flavor, the client moves on to the next auth-flavor in the SECINFO response.

In order for the server to better communicate its security policies to the client and limit the cases in which connections are established only to be rejected due ti inapproprriate connection

characteristics, the followin pseudo-flavor can be used in the response to SECINFO and SECINFO_NONAME requests.

- *PFL_CRYPT indicates that encryption, such as thar provided by rpc-with-tls is to be required on any connection to support access to the current portion of the server namespace.

- *PFL_CRYPTMPA indicates that encryption such as that provided by rpc-with-tls is to be required together with mutual peer authentication on any connection to support access to the current portion of the server namespace.

- *PFL_ANY indicates that, for subsequent auth-flavors, there are no connection-based restritios to be assumed by the client, allowing connections that do not use encryption to be used as well as those that do.

When any of these flavors are incuded in a SECINFO or SECINFO_NONAME response, the procEdure described above is modified as follows:

- *When scanning the response, if pseudo-flavor entries are encountered, the client takes note of them, keeping track of the last such entry encountered.

- *When processing a flavor entry in the response, the most recent pseudo-flavor is used in establishing the connection to be used, to avoid using connection that the server;s scurity policy will not allow to be used. The initial connection characteristics used will be a rhose that meets the requiremnts specified by the pseudo-flavor and is consistent with the client's security policy.

As an example of how these pseudoflavors can be used, consider a situation in which it is the server's security to only accept encrypted requests, whether the encryption is provide by RPCSEC_GSS or by rpc-with-tls. A response consisting of the following entries will specify this behavior so the client can connect appropriately.

- *PFL_CRYPT to indicate that the following auth-flavor entry, in this case AUTH_SYS, is to be imited to use on connection that provide encryption of requests and responses.

- *AUTH_SYS indicates that use of the auth-flavor AUTH_SYS is acceptable to access the current potiono of the server bamespace. Because of the preceeding PFL_CRYPT, the client knows that this is only accepted on encrypted connections.

  If the server supports rpc-with-tls and allows its use together with AUTH_SYS, this connection is used. Otherwise processing proceeds to use subsequent entries.

*PFL_ANY to indicate that the subsequent auth-flavor entry, in this case RPCSEC_GSS, is not limited to any particular connection type.

*RPC_SECGSS indicating the requirement for encryption provided by rpcsec_gss.

16.1. Dealing with Multiple Connections

[Author Aside]: All unannotated paragraphs in this section are considered part of Consensus Item #32b.

Because effective security will require both an appropriate auth flavor (and possibly services provide by the auth flavor) together with appropriate connection characteristics, it is often necessary that clients and server be aware of connection characteristics:

*When multiple connections with different security-related characteristics, are used to access a server, the clients needs to ensure that each request is issued on an appropriate connection.

*Similarly, in such situations, the server needs to be aware of the security-related characteristics for the connection on which each request is received, in order to enforce its security policy.

Depending on how the client and server implementations are structured, implementations may have to be changed to accomplish the above.

In the case of NFSv4.1 and above, the protocol requires that requests associated with a given session only be issued on connections bound to that session and accepted by the server only when that binding is present. This makes it likely that clients or servers will be able to correctly associate requests with the appropriate connections although additional work might be necessary to enable them to determine, for any given connection, its security characteristics.

In the case of NFSv4.0, no such binding is present in the protocol so that, depending on existing implementations' layering, channel binding functionality might have to be added.

17. Future Security Needs

This section deals with weaknesses in the security of the existing protocol which might be dealt with in future minor versions or in extensions to existing minor versions. Whle potential improvements to NFSv4 security are discussed in Section 17.1 the details of these extnsions are not specified and will, when the workin group decdes to implement them, in new standrards-track extension documents.

Not discussed in this section are the extensions provided to inform of the specfics of the ACL semantic implemented by particular file system

17.1. Desirable Additional Security Facilities

[Author Aside]: All unannotated paragraphs in this section are to be considered part of Consensus Item #35a.

[Author Aside]: This section is basically an outline for now, to be filled out later based on Working Group input, particularly from Chuck Lever who suggested this section and has ideas about many of the items in it.

*Security for data-at-rest, most probably based on facilities defined within SAN.

*Support for content signing.

*Revision/extension of labelled NFS to provide true interoperability and server-based authorization.

*Work to provide more security for RDMA-based transports. This would include the peer authentication infrastructure now being developed as part of RPC-over-RDMA version 2. In addition, there is a need for an RDMA-based transport that provides for encryption, which might be provided in number of ways.

18. Security Considerations

18.1. Changes in Security Considerations

Beyond the needed inclusion of a threat analysis as Section 18.4 and the fact that all minor versions are dealt with together, the Security Considerations in this section differ substantially from those in [RFC7530] and [RFC8881]. These differences derive from a number of substantive changes in the approach to NFSv4 security presented in [RFC7530] and [RFC8881] and that appearing in this document.

These changes were made in order to improve the security of the NFSv4 protocols because it had been concluded that the previous treatment of these matters was in error, leading to a situation in which NFSv4's security goals were not met. As a result, this document supersedes the treatment of security in earlier documents, now viewed as incorrect. However, it will, for the benefit of those familiar

with the previous treatment of these matters, draw attention to the important changes listed here.

- There is a vastly expanded range of threats being considered as described in Section 18.1.1

- New facilities provided by RPC on a per-connection basis can be used to deal with security issues, as described in Section 18.1.2. These include the use encryption on a per-connection basis, and the use of peer mutual authentication, to mitigate the security problems that come with the use of AUTH_SYS.

- The handling of identities with superuser privileges is no longer part of NFSv4 semantics, even though many platforms on which NFSv4 servers are implemented continue to depend, for local operation, on the existence of such identities.

  NFSv4 servers **SHOULD NOT** provide for such unrestricted access since doing so would provide a means by which an escalation-of-privilege on a client could be used to compromise a server to which it was connected, affecting all clients of that server.

  In connection with the use of "**SHOULD NOT**" above, and similar uses elsewhere, it is to be understood that valid reasons to do other than recommended are limited to the difficulty of promptly changing existing server implementations and the need to accommodate clients that have become dependent upon the existing handling. Further, those maintaining or using such implementations need to be aware of the security consequences of such use as well as the fact that clients who become aware of this characteristic may not be inclined to store their data on such a system.

- The appropriate handling of ACL-based authorization and necessary interactions between ACLs and modes is now specified in this standards-track document rather it being assumed that the behavior of server implementations needs to be accepted and deferred to.

18.1.1. Wider View of Threats

Although the absence of a threat analysis in previous treatments makes comparison most difficult, the security-related features described in previous specifications and the associated discussion in their security considerations sections makes it clear that earlier specifications took a quite narrow view of threats to be protected against and placed the burden of providing for secure use on those deploying such systems with very limited guidance as to how such secure use could be provided.

One aspect of that narrow view that merits special attention is the handling of AUTH_SYS, at that time in the clear, with no client peer authentication.

With regard to specific threats, there is no mention in existing security considerations sections of:

- *Denial-of-service attacks.
- *Client-impersonation attacks.
- *Server-impersonation attacks.

The handling of data security in-flight is even more troubling.

- *Although there was considerable work in the protocol to allow use of encryption to be negotiated when using RPCSEC_GSS. The existing security considerations do not mention the potential need for encryption at all.

  It is not clear why this was omitted but it is a pattern that cannot repeated in this document.

- *The case of negotiation of integrity services is similar and uses the same negotiation infrastructure.

  In this case, use of integrity is recommended but not to prevent the corruption of user data being read or written.

  The use of integrity services is recommended in connection with issuing SECINFO (and for NFSv4.1, SECINFO_NONAME). The presence of this recommendation in the associated security considerations sections has the unfortunate effect of suggesting that the protection of user data is of relatively low importance.

18.1.2. Connection-oriented Security Facilities

Such RPC facilities as RPC-with-TLS provide important ways of providing better security for all the NFSv4 minor versions.

In particular:

- *The presence of encryption by default deals with security issues regarding data-in-flight, whether RPCSEC_GSS or AUTH_SYS is used for client principal identification.
- *Peer authentication provided by the server eliminates the possibility of a server-impersonation attack, even when AUTH_SYS or AUTH_NONE is used to issue requests

* When mutual authentication is part of connection establishment, there is a possibility, where an appropriate trust relationship exists, of treating the uids and gids presented in AUTH_SYS requests, as effectively authenticated, based on the authentication of the client peer.

18.1.3. Necessary Security Changes

[Consensus Needed (Items #36a, #37a)]: For a variety of reasons, there are many cases in which a change to the security approach has been adopted but for which provisions have been made in order to give implementers time to adapt to the new approach. In such cases the words "**SHOULD**", "**SHOULD NOT**", and "**RECOMMENDED**" are used to introduce the new approach while use of the previous approach is allowed on a temporary basis, by limiting the valid reasons to bypass the recommendation. Such instances fall into two classes:

* [Consensus Needed (Item #36a)]: In adapting to the availability of security services provided by RPC on a per-connection basis, allowance has been made for implementations for which these new facilities are not available and for which, based on previous standards-track guidance, AUTH_SYS was used, in the clear, without client-peer authentication.

* [Consensus Needed (Item #37a)]: In dealing with server implementations that support both ACLs and the mode attribute, previous specifications have allowed a wide range of possible server behavior in coordinating these attributes. While this document now clearly defines the recommended behavior in dealing with these issues, allowance has been made to provide time for implementations to conform to the new recommendations.

[Consensus Needed (Items #36a, #37a)]: The threat analysis within this Security Considerations section will not deal with older servers for which allowance has been made but will explore the consequences of the recommendations made in this document.

18.1.4. Compatibility and Maturity Issues

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #38a.

Given the need to drastically change the NFSv4 security approach from that specified previously, it is necessary for us to be mindful of:

* The difficulty that might be faced in adapting to the newer guidance because the delays involved in designing, developing, and testing new connection-oriented security facilities such as RPC-with-TLS.

* The difficulty in discarding or substantially modifying previous existing deployments and practices, developed on the basis of previous normative guidance.

For these reasons, we will not use the term "**MUST NOT**" in some situations in which the use of that term might have been justified earlier. In such cases, previous guidance together with the passage of time may have created a situation in which the considerations mentioned above in this section may be valid reasons to defer, for a limited time, correction of the current situation making the term "**SHOULD NOT**" appropriate, since the difficulties cited would constitute a valid reason to not allow what had been recommended against.

18.1.5. Discussion of AUTH_SYS

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #39a.

An important change concerns the treatment of AUTH_SYS which is now divided into two distinct cases given the possible availability of connection-oriented support from RPC.

When such support is not available, AUTH_SYS **SHOULD NOT** be used, since it makes the following attacks quite easy to execute:

* The absence of authentication of the server to the client allow server impersonation in which an imposter server can obtain data to be written by the user and supply corrupted data to read requests.

* The absence of authentication of the client user to the server allow client impersonation in which an imposter client can issue requests and have them executed as a user designated by imposter client, vitiating the server's authorization policy.

  With no authentication of the client peer, common approaches, such as using the source IP address can be easily defeated, allowing unauthenticated execution of requests made by the pseudo-clients

* The absence of any support to protect data-in-flight when AUTH_SYS is used result in further serious security weaknesses.

In connection with the use of the term "**SHOULD NOT**" above, it is understood that the "valid reasons" to use this form of access reflect the Compatibility and Maturity Issue discussed above in Section 18.1.4 and that it is expected that, over time, these will become less applicable.

18.2. Security Considerations Scope

18.2.1. Discussion of Potential Classification of Environments

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #40a.

This document will not consider different security policies for different sorts of environments. This is because,

- *Doing so would add considerable complexity to this document.
- *The additional complexity would undercut our main goal here, which is to discuss secure use on the internet, which remain an important NFSv4 goal.
- *The ubiquity of internet access makes it hard to treat corporate networks separately from the internet per se.
- *While small networks might be sufficiently isolated to make it reasonable use NFSv4 without serious attention to security issues, the complexity of characterizing the necessary isolation makes it impractical to deal with such cases in this document.

18.2.2. Discussion of Environments

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #40b.

Although the security goal for Nfsv4 has been and remains "secure use on the internet", much use of NFSv4 occurs on more restricted IP corporate networks with NFS access from outside the owning organization prevented by firewalls.

This security considerations section will not deal separately with such environments since the threats that need to be discussed are essentially the same, despite the assumption by many that the restricted network access would eliminate the possibility of attacks originating inside the network by attackers who have some legitimate NFSv4 access within it.

In organizations of significant size, this sort of assumption of trusted access is usually not valid and this document will not deal with them explicitly. In any case, there is little point in doing so, since, if everyone can be trusted, there can be no attackers, rendering threat analysis superfluous.

In corporate networks, as opposed to the Internet, there is good reason to be less concerned about denial-of-service attacks, since there is no tangible benefit to attackers inside the organization,

and the anonymity that makes such attacks attractive to outside attackers will not be present.

The above does not mean that NFSv4 use cannot, as a practical matter, be made secure through means outside the scope of this document including strict administrative controls on all software running within it, frequent polygraph tests, and threats of prosecution. However, this document is not prepared to discuss the details of such policies, their implementation, or legal issues associated with them and treats such matters as out-of-scope.

Nfsv4 can be used in very restrictive IP network environments where outside access is quite restricted and there is sufficient trust to allow, for example, every node to have the same root password. The case of a simple network only accessible by a single user is similar. In such networks, many thing that this document says "SHOULD NOT" be done are unexceptionable but the responsibility for making that determination is one for those creating such networks to take on. This document will not deal further with NFSv4 use on such networks.

18.2.3. Insecure Environments

As noted in Section 18.2.2, NFSv4 is often used in environments of much smaller scope than the internet, with the assumption often being made, that the prevention of NFSv4 access from outside the organization makes the attention to security recommended by this document unnecessary, the possibility of insider attacks being explicitly or implicitly disregarded.

As a result, there will be implementations that do not conform to these recommendations, many of which because the implementations were based on the RFCs [RFC3530], [RFC7530], [RFC5661], or [RFC8881]. In addition to these cases in which the disregard of the recommendations is considered valid because implementors relied on existing normative guidance, there will be other cases in which implementors choose to ignore these recommendations,

Despite the original focus of [RFC2119] on interoperability, many such implementations will interoperate, albeit without effective security, whether the reasons that the recommendations are not adhered to are considered valid or not.

When such insecure use is mentioned in this Security Considerations section it will only be in explaining the need for the recommendations, by explaining the likely consequences of not following them. The threat analysis, in Section 18.4 and included subsections, will not consider such insecure use and will concern itself with situation in which these recommendations are followed.

18.3. Major New Recommendations

18.3.1. Recommendations Regarding Security of Data in Flight

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #41b.

It is **RECOMMENDED** that client implementer always support data privaacy in some form, whether using connection-based encryption or data privacy services as provided by RPSCEC_GSS. This is despite the fact thaat previous specficatons stopped short of requiring this support on the client.

It is **RECOMMENDED** that requesters always issue requests with data security (i.e. with protection from disclosure or modification in flight) whether provided at the RPC request level or on a per-connection basis, irrespective of the responder's requirements.

It is **RECOMMENDED** that implementers provide servers the ability to configure policies in which requests without data security will be rejected as having insufficient security.

It is **RECOMMENDED** that servers use such policies over either their entire local namespace or for all file systems except those clearly designed for the general dissemination of non-sensitive data.

When these recommendations are not followed, data, including data for which disclosure is a severe [problem is exposed to unwanted disclosure or modification in flight. Depending on the server to be aware of the need for confidentiality or integrity, as expected by previous specifications, has not proved workable, making encryption by default as provided uniformly by RPC (e.g. through RPC-with-TLS) necessary.

18.3.2. Recommendations Regarding Client Peer Authentication

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #41c.

It is **RECOMMENDED** that clients provide authentication material whenever a connection is established with a server capable of using it to provide client peer authentication.

It is **RECOMMENDED** that implementers provide servers the ability to configure policies in which attempts to establish connections without client peer authentication will be rejected.

It is **RECOMMENDED** that servers adopt such policies whenever requests not using RPCSEC_GSS (i.e. AUTH_NONE Or AUTH_SYS) are allowed to be executed.

When these recommendations are not followed, it is possible for connections to be established between servers and client peers that have not been authenticated with the following consequences:

- The server will be in the position of executing requests where the identity used in the authorization of operations is not authenticated, including cases in which the identification has been fabricated by an attacker.

- When no identification of a specific user is needed or present (i.e AUTH_NO is used) there is no way of verifying that the request was issued by the appropriate client peer.

When the recommendations are followed, use of AUTH_SYS can be valid means of user authentication, so long as due attention is paid to the discussion in Section 18.4.6.1. Despite this fact, the description of AUTH_SYS as an "**OPTIONAL** means of authentication"is no longer appropriate since choosing to use it requires heightened attention to security as discussed later in this document.

18.3.3. Recommendations Regarding Superuser Semantics

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #52c.

It is **RECOMMENDED** that servers adhere to the ACL semantics defined in this document and avoid granting to any remote user, however authenticated, unrestricted access capable of authorizing access where the file/directory ACL would deny it.

Servers are free to conform to this recommendation either by implementing authorization semantics without provisions for superusers or by mapping authenticated users that would have superuser privileges to users with with more limited privileges (e.g. "nobody").

It needs to b e understood that the second of these choices is preferable when there are NFsv4-accessible files owned by a special users (e.g. root) whose compromise might be taken advantage of by attackers to enable permanent unauthorized access to a server.

18.3.4. Issues Regarding Valid Reasons to Bypass Recommendations

[Author Aside]: All unannotated paragraphs within this section are considered part of Consensus Item #41d.

Clearly, the maturity and compatibility issues mentioned in Section 18.1.4 are valid reasons to bypass the proposed recommendations requiring pervasive use of encryption, as long as these issues continue to exist.

[Author Aside]: The question the working group needs to address is whether other valid reasons exist.

[Author Aside]: In particular, some members of the group might feel that the performance cost of conection-based encryption constitutes, in itself, a valid reason to ignore the above recommendations.

[Author Aside]: I cannot agree and feel that accepting that as a valid reason would undercut Nfsv4 security improvement, and probably would not be acceptable to the security directorate. However, I do want to work out an a generally acceptable compromise. I propose something along the following lines:

In dealing with recommendations requiring pervasive use of connection-based encryption, it needs to be understood that the connection-based encryption facilities are designed to be compatible with facilities to offload the work of encryption and decryption. When such facilities are not available, at a reasonable cost, to NFSv4 servers and clients anticipating heavy use of NFSv4, then the lack of such facilities can be considered a valid reason to bypass the above recommendations, as long as that situation continues.

18.4. Threat Analysis

18.4.1. Threat Analysis Scope

Because of the changes that have been made in NFSv4 security, it needs to be made clear that the primary goal of this threat analysis is to explore the threats that would be faced by implementations that follow the recommendations in this document.

When the possibility is raised of implementations that do not conform to these recommendations, the intention is to explain why these recommendations were made rather that to expand the the scope of the threat analysis to include implementations that bypass/ignore the recommendations.

The typical audience for threat analyses is client and server implementers, to enable implementations to be developed that are resistant to possible threats. While much of the material in Section 18.4 is of that form, it also contains material that relates to threats whose success depends primarily on the ways in which the implementation is deployed, such as the threats discussed in Sections 18.4.2, 18.4.4 and 18.4.3. While it is not anticipated that those deploying implementations will be aware of the detail of this threat analysis, it is expected that implementors could use this material to properly set expectations and provide guidance helpful to making deployments secure.

18.4.2. Threats based on Credential Compromise

In the past, it had been assumed that a user-selected password could serve as a credential, the knowledge of which was adequate to authenticate users and provide a basis for authorization.

That assumption is no longer valid for a number of reasons:

- The inability or unwillingness of users to remember multiple passwords has meant that the single password they will remember controls access to large set of resources, increasing the value of this knowledge to attackers and the effort that will be expended to obtain it.

  In addition, the common use of a single password for applying to all of a user's data has resulted in a situation in which the client is aware of user passwords (since they are used for client login) that apply to data on many servers. As will be seen later, this has the effect of changing the considerations appropriate to comparing the security of AUTH_SYS and RPCSEC_GSS.

- CPU developments have made exhaustive search possible for larger classes of passwords.

- The success of "phishing" attacks taking advantage of user gullibility provides an additional path to credential compromise which need to be addressed in the near-term by those deploying NFSv4, and will eventually need work in the security infrastructure on which NFSv4 is built.

In the near term, there are a number of steps, listed below that those deploying NFSv4 servers can take to mitigate these weaknesses. These steps are outside the scope of the NFSv4 protocols and implementors only role with regard to them is to make it clear that these weaknesses exist and generally require mitigation.

- Limitations on password choice to eliminate weak passwords.

- Requirements to change passwords periodically.

- User education about "phishing" attacks including ways to report them and effective ways of replacing a compromised password.

From a longer-term perspective, it appears that password-based credentials need to be either replaced or supplemented by some form of multi-factor authentication. Since NFSv4's approach to security relies on RPC, that work would most probably be done within the RPC layer, limiting the work that implementations and the NFSv4 protocols would have to do to adapt to these changes once they are available.

While the precise form of these changes is not predictable, the following points need to be kept in mind.

* [Verification Needed (Item #53a)]: For those using RPCSEC_GSS authentication of principals, it appears that RPCSEC_GSS interface is flexible enough that the addition of a second credential element, in the form of a one-time code could be added.

  [Elaboration/Verification Needed (Item #53a)]: Enhancement of Kerberos is one possibility to provide multi-factor authentication. However, work on this is not far enough along to enable deployment to be discussed now.

  If this approach were taken, rogue servers would still have access to user passwords but their value would be reduced since the second credential element would have a very limited lifetime.

* For those using AUTH_SYS to identify principals, the client operating system's authentication of user at login would need to be enhanced to use multi-factor authentication.

  If this were done, the client would retain responsibility for credential verification with the server needing to trust the client, as discussed in Section 18.4.6.1.

  Although there is need for protocol standardization to enable this approach to be commonly used, it is not likely to be widely used until some operating system adopts it for user login.

* One important variant of AUTH_SYS use concerns clients used by a single user, when, as recommended, client-peer authentication is in effect For such clients, it is possible for the authentication of that specific client peer to effectively become the second factor, in a multi-factor authentication scheme.

  Despite the fact that the the RPC-with-TLS specification [RFC9289]) does not allow TLS to used for user authentication, this arrangement in which the user identity is inferred from the peer authentication, could be used to negate the effects of credential compromise since an attacker would need both the user password, and the physical client to gain access.

18.4.3. Threats Based on Rogue Clients

When client peers are not authenticated, it is possible to a node on the network to pretend to be a client. In the past, in which servers only checked the from-IP address for correctness, address spoofing would allow unauthenticated request to be executed, allowing confidential data to be read or modified.

Now that such use of AUTH_SYS is recommended against, this cannot happen. The recommended practice is to always authenticate client peers making this sort of imposture easily detectable by the server.

Despite this protection, it is possible that an attacker, through a client vulnerability unrelated to NFSv4, or the installation of malware, could effectively control the client peer and act as imposter client would, effectively undercutting the authentication of the client. This possibility makes it necessary, as discussed in Section 18.4.6.1 that those deploying NFSv4 clients using AUTH_SYS takes steps to limit the set of user identifications accepted by a server and to limit the ability of rogue code running on the server to present itself as a client entitled to use AUTH_SYS.

18.4.4. Threats Based on Rogue Servers

When server peers are not authenticated, it is possible for a node on the network to act as if it were an NFSv4 server, with the ability to save data sent to it and use it or pass it to other, rather than saving it in the file system, as it needs to do..

When current recommendations are adhered to, this is be prevented as follows:

- When RPCSEC_GSS is used, the mutual authentication of the server and client principal provides assurance the server is not an imposter.

- When AUTH_SYS or AUTH_NONE is used, the mutual authentication of client and server peers provides assurance the server is not an imposter.

Despite this protection, it is possible that an attacker, through a operating system vulnerability unrelated to NFSv4, or the installation of malware, could effectively control the server peer and act as an imposter server would, effectively undercutting the authentication of the server.

The above possibility makes it necessary, that those deploying NFSv4 servers take the following steps, particularly in cases in cases in which the server has access to user credentials, including, but not limited to, cases in which AUTH_SYS is supported

When an NFSv4 is implemented as part of a general-purpose operating system, as it often is, steps need to be taken to limit the ability of attackers to take advantage of operating system vulnerabilities that might allow the attacker to obtain privileged access and subvert the servers operation, turning it, effectively, into a rogue server.

Such steps include controls on the software installed on the machine acting as the server, and limitation of the network access to potentially dangerous sites.

18.4.5. Data Security Threats

When file data is transferred in the clear, it is exposed to unwanted exposure. As a result, this document recommends that encryption always be used to transfer NFSv4 requests and responses.

That encryption, whether done on encrypted connections, or on a per-request basis, using RPCSEC_GSS security services, provides the necessary confidentiality. In addition, it contributes to security in other ways as well:

* The ability of an attacker to plan and execute attacks is enhanced by the monitoring of client-server traffic, even if none of the data intercepted is actually confidentiality.

  An attacker can deduce which users are allowed to read or write a specific file by examining the results of OPEN and ACCESS operations allowing later attacks to impersonate users with the appropriate access.

* All the methods of encryption used with NFS4 provide a checksum, to enable the detection of unwanted modifications to data being read or written.

18.4.6. Authentication-based threats

18.4.6.1. Attacks based on the use of AUTH_SYS

Servers, when they allow access using AUTH_SYS, to a specific client machines using AUTH_SYS are responsible for ensuring that the principal identifications presented to the server can be relied upon.

The existence of client-peer authentication as recommended in Section 18.1.5 means that imposter servers can be detected and not allowed to use AUTH_SYS. However there are an additional number of issues that need to be addressed to adequately protect against use of AUTH_SYS enabling attacks:

* The server accepting requests using AUTH_SYS needs to determine that the authenticated client-peer can be trusted to properly authenticate the principals that it identifies in requests.

  The specific standards for trustworthiness are up to the server but they need to take account of the controls in place to prevent malware from pretending to be a client and thus taking advantage of the fact that the request is from the expected client machine.

This server **MUST NOT** accept AUTH_SYS requests from unknown clients or from unauthenticated clients.

*[Elaboration Needed (Item #54a)]: The client verification procedure needs to take steps to prevent code on a compromised client to presenting itself as the successor to a legitimate client, taking advantage of the fact that the machine is the same.

*Given the inherent vulnerabilities of client operating systems, it is desirable, to limit the set of users whose identification will be accepted. The elimination of particular users such as "root' is one long-standing approach to the issue but it probably isn't sufficient in most environments. More helpful would be the ability to exclude multiple sensitive users or group of users or to limit the user identifications accepted to a user group or a single user.

Another important that issue that arises when AUTH_SYS is used concerns the storage of credentials on the clients. While it is theoretically possible for these not to be of use elsewhere, the reluctance/inability of users to remember multiple passwords means that these credentials will be used by many clients and will need to be updated as users are added or deleted or when passwords are changed. The propagation of these credentials and their storage on clients could be the basis for attacks if appropriate step are not taken to secure this data.

While it is helpful to store a cryptographic hash of the password rather than the password itself, this does not dispose of the issue, since possession of the hash would greatly simplify an exhaustive search for the password, since the attacker could limit login attempts to guessed password whose hash value matched the value obtained from the files on the client.

Although it is true that making clients responsible for authentication of user identities undercuts much of the original motivation for making RPCSEC_GSS **REQUIRED** to implement, it needs to be understood that the situation today is different from that when this decision was made.

*It has been recommended that servers not allow unauthenticated clients to issue requests using AUTH_SYS.

*The identification of a request as issued by the user with uid zero, no longer provides access without file access authorization.

*Given that users are unaware of where their files are located and it is desirable that they are able to remain unaware of this, it is natural that they use the same password to authenticate

themselves for local resource use as for use of files located on NFSv4 servers.

Support for AUTH_SYS in NFSv4 was included for a number of reasons which still hold true today, despite the fact that the original mistake, to make no reference to the security consequences of doing so, is now being corrected. Such provision is necessary for the following reasons, that go beyond the need to temporarily accommodate implementations following the older specifications, for a number of reasons:

*When considered, as NFS was to intended to be, as consistent with local access as possible, AUTH_SYS was the natural way of providing authentication, just as it had been done for local files.

While use of AUTH_SYS exposes user passwords to the client operating system, the fact that user are unable or unwilling to use different passwords for different files in a multi-server namespace means this issue will be present even when AUTH_SYS is not used.

*[Elaboration Needed (Item #55a)]: In many important environments including cloud environments, important implementation constraints has made use of Kerberos impractical.

[Verification Needed (Item #55a)]: In such environments, client credentials are maintained by the cloud customer while the cloud provider manages network access.

18.4.6.2. Attacks on Name/Userid Mapping Facilities

NFSv4 provides for the identification users and groups in two ways (i.e. by means of strings of the form name@domain or strings containing numeric uid/gid values) while file systems used on NFSv4 servers typically use 32-bit uids and gids.

As a result, NFSv4 server implementations are required to have some means of translating between the name@domain form and the numeric form used internally. While the specifics of this translation are not specified as part of the NFSv4 protocols, is required for server implementations to work, and, if it not done securely and attackers have the ability to interfere with this translation, it gives them the ability to interfere with authorization as follows:

*When authentication occurs using user names, as occurs when RPCSEC_GSS, a mistranslation might allow the numeric value used in authorization to allow access to a file the authenticated user would not be allowed to access.

*When any authentication occurs on the client and the uid is presented to the server using AUTH_SYS a mistranslation to the string form could result in confusion and uncertainty about the users allowed to access the file.

18.4.7. Disruption and Denial-of-Service Attacks

18.4.7.1. Attacks Based on the Disruption of Client-Server Shared State

When data is known to both the client and server, a rogue client can interfere with the correct interaction between client and server, by modifying that shared data, including locking state and session information.

For this reason, it is recommended that client-peer authentication be in effect, because, it it were not, a different client could could easily modify data that the current client depend on, disrupting ones interaction with the server.

It is still possible, if one's client is somehow compromised, as described in Section 18.4.3, for various forms of mischief to occur:

*Locks required for effective mutual exclusion can be released, causing application failures.

*Mandatory share locks can be obtained preventing those with valid access from opening file that they are supposed to have access to.

*Session slot sequence numbers may be rendered invalid if requests are issued on existing sessions. As a result, the client that issued a request would receive unexpected sequence errors.

18.4.7.2. Attacks Based on Forcing the Misuse of Server Resources

It is is also possible for attacks to be mounted, in the absence of the ability to obtain or modify confidential data, with the sole goal of the attack being to make spurious requests, with no expectation that the request will be authorized but with the goal of causing resources that would otherwise be used to service valid requests to be unavailable due to the burden of dealing with numerous invalid requests.

The design of the NFSv4 protocols requires that clients establishing new connections make initial requests which establishes a shared context referred to by subsequent requests which might request substantive actions (e.g. client and session ids). This structure

helps mitigate the effect of such denial-of-service attacks as described below.

- The server can limit the resources devoted to connections not yet fully identified without unduly restricted connections which have identified themselves.

- The recommendation that client peers authenticate themselves, allows unknown clients to be dispensed with at an early stage negating their ability to make requests which could require file system action to obtain information needed to make authorization decisions (e.g. ACLs or other authorization-related) file attributes.

19. IANA Considerations

[Author Aside]: All unannotated paragraphs in this section are to be considered part of Consensus Item #32c.

Because of the shift from implementing security-related services only in connection with RPCSEC_GSS to one in which connection-oriented security has a prominent role, a number if new values need to be assigned.

These include new authstat values to guide selection of a connection types acceptable to both client and server, presented in Section 19.1 and new pseudo-flavors to be used in the process of security negotiation, presented in Section 19.2.

19.1. New Authstat Values

[Author Aside]: All unannotated paragraphs in this section are to be considered part of Consensus Item #32d.

The following new authstat values are necessary to enable a server to indicate that the server's policy does not allows requests to be made on the current connection because of security issues associated with connection type used. In the event they are received, the client needs to establish a new connection.

- The value XP_CRYPT indicates that the server will not support access using unencrypted connections while the current connection is not encrypted.

- The value XP_CPAUTH indicates that the server will not support access using connections for which the client peer has not authenticated itself as part of connection while the current connection has not been set up in that way.

19.2. New Authentication Pseudo-Flavors

[Author Aside]: All unannotated paragraphs in this section are to be considered part of Consensus Item #32e.

The new pseudo-flavors described in this section are to be made available to allow their return as part of the response to the SECINFO and SECINFO_NONAME operations. How these operations are to used to negotiate the use of appropriate auth flavors and associated security-relevant connection characteristics is discussed in Section 16.

The following pseodo-flavors are to be defined:

*PFL_CRYPT is returned to indicate that subsequent secinfo entries are to considered limited to use on connections for which transport-level encryption is provided.

*PFL_CRYPTMPA is returned to indicate that subsequent secinfo entries are to be considered limited to use on connections on which mutual peer authentication has been provided at connection setup.

*PFL_ANY is returned to indicate that subsequent secinfo entries are not to be considered limited to any particular type of connection.

20. References

20.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <https://www.rfc-editor.org/info/rfc7530>.

[RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", RFC 8178, DOI 10.17487/RFC8178, July 2017, <https://www.rfc-editor.org/info/rfc8178>.

[RFC8881] Noveck, D., Ed. and C. Lever, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 8881, DOI

10.17487/RFC8881, August 2020, <https://www.rfc-editor.org/info/rfc8881>.

[RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", RFC 7862, DOI 10.17487/RFC7862, November 2016, <https://www.rfc-editor.org/info/rfc7862>.

[RFC9289] Myklebust, T. and C. Lever, Ed., "Towards Remote Procedure Call Encryption by Default", RFC 9289, DOI 10.17487/RFC9289, September 2022, <https://www.rfc-editor.org/info/rfc9289>.

[nist-209] NIST, "SP 800-209 Security Guidelines for Storage Infrastructure".

20.2. Informative References

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/info/rfc8257>.

[RFC3010] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "NFS version 4 Protocol", RFC 3010, DOI 10.17487/RFC3010, December 2000, <https://www.rfc-editor.org/info/rfc3010>.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, DOI 10.17487/RFC3530, April 2003, <https://www.rfc-editor.org/info/rfc3530>.

[RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, DOI 10.17487/RFC5661, January 2010, <https://www.rfc-editor.org/info/rfc5661>.

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, DOI 10.17487/RFC1813, June 1995, <https://www.rfc-editor.org/info/rfc1813>.

[I-D.ietf-nfsv4-internationalization]
Noveck, D., "Internationalization for the NFSv4 Protocols", Work in Progress, Internet-Draft, draft-ietf-nfsv4-internationalization-07, 20 November 2023, <https://

datatracker.ietf.org/doc/html/draft-ietf-nfsv4-internationalization-07>.

[I-D.dnoveck-nfsv4-acls] Noveck, D., "ACLs within the NFSv4 Protocols", Work in Progress, Internet-Draft, draft-dnoveck-nfsv4-acls-00, 29 February 2024, <https://datatracker.ietf.org/doc/html/draft-dnoveck-nfsv4-acls-00>.

[errata] IESG, "IESG Processing of RFC Errata for the IETF Stream", July 2008, <https://www.ietf.org/about/groups/iesg/statements/processing-rfc-errata/>.

Appendix A. Changes Being Made

This appendix summarizes the significant changes between the treatment of security in previous minor version specification documents (i.e. RFCs 3630, 7530, 5661 and 8881) and the new treatment that is intended to apply to NFSv4 as a whole. This includes changes made in both this document and in the companion document relaing to ACLs [I-D.dnoveck-nfsv4-acls].

This revieww is expected to be helpful to implementers familiar with previous specifications but also has an important role in guiding the search for working group consensus as to these changes and in identifying potential compatibility issues.

A.1. Motivating Security Changes

A.1.1. Fundamental Security Changes

A number of changes reflect the basic motivation for a new treatment of NFSv4 security. These include the ability to obtain privacy and integrity services from RPC on a per-connection basis rather than as a service ancillary to a specific auth flavor.

This need to properly accomodate this new ficus motivated a major reorganization of the treatment of security together with further emphasis on the security of data in flight, described in detail in Section 15. In addition, the security negotiation framework for NFSv4 has been enhanced to support the combined negotiation of authentication-related services and connection characteristics, as described in Section 16.

Despite these major changes, there are not expected to be troublesome compatibility issues between peers supporting provision of security services on a per-connection basis and those without such support. Clients who are adapted to the new security facilities can use them when the server supports them, but will sometimes choose to use the older facilities when interacting with older servers. Servers who

support the new facilities have the option of rejecting client connections from older clients but are encouraged to adopt policies rejecting such connections. Over time, it is expected that use of the older facilities will become less common as the newer facilities become more commonly implemented and the importance of security becomes more generally recognized

Another such change was in the treatment of AUTH_SYS, previously described, inaccurately, as an "**OPTIONAL** means of authentication" with the unfortunate use of the RFC2119 keyword obscuring the negative consequences of the typical use of AUTH_SYS (in the clear, without client-peer authentication) for security by enabling the execution of unauthenticated requests.

The new treatment avoids the inappropriate use of term "authentication" for all activities triggered by the use of RPC auth flavors and clearly distinguishes those flavors providing authentication from those providing identification only or neither identification nor authentication. For details, see Section 14.

As part of the necessary shift from a narrative focused on justifying security choices already made to evaluating the adequacy of existing security facilities, there has been a need to discuss NFSv4 security gaps, which might be addressed later. See Section 17.1 for a discussion of possible security-related extensions.

There are twelve consensus items (described in Appendix B) that are involved in effecting this class of changes. Although it is generally understood that the working group needs to take advantage of RPC-with-TLS, change how use of AUTH_SYS is presented, and include a credible threat analysis, there needs to be further discussion of these items as working group members are likely to disagree about the relative importance of:

- *accommodating implementations based on previous specifications.
- *providing a description of the expected path toward secure implementations
- *maintaining and improving implementation performance

For each of the following items, there may need to be extensive discussion to arrive at a consensus text.

- *Consensus Item #32 concerns extension of the process of SECINFO negotiation to accommodate the use of facilities such as rpc-with-tls, in addition to the current approach which directs only the choice of a suitable auth flavor.

In the current document, pseudoflavors are added to allow the specification of connection characteristics, lthough it would be possible to avoid use and select connection characteristics by a process of trial and error.

*Consensus Item #52 concerns the handlinng of superuser semantics within NFSv4.

There is a long tradition, within NFS, of treating superusr semantic, involving the elimination of file authorization checks, as part of protocol semantics. The security problems that this creates have been dealt with via "root-squashing", outside the protocol specification. Given that a threat analysis would essentially make such suppression of superuser semantics unavoidable, the current document deals with such semantics outside of the NFSv4 protocol.

*Consensus Item #36 concerns how to deal with the troublesome legacy of use of AUTH_SYS without client authentication and the common transmission of requests and responses without encryption.

The specification now makes it clear that this has the potential to cause harm, while needing to deal ith the fact that it is a long time before this could be forbidden or recommmended against without allowances being made for current implementations. The ways that this issue needs to be dealt with require further discussion.

*Consensus Item #38 concerns temporary accommodations to deal with previous implementations which are unsafe.

This is a consequence of existing implemenations based on current RFCs. The wording will require extensive discussion.

*Consensus Item #39 defines the new safer approach to AUTH_SYS.

There needs to be discussion about how this safety should be characterized, given that you are trusting the client OS.

*Consensus Item #40 concerns the scope for the threat analysis in the Security Considerations section.

Will need to reach a consensus on this as there appears to be no way we can make allowances in which physical isolation makes a significant difference, since this ssentially ignores insider threats.

*Consensus Item #41 discusses the major new security recommendations regarding protection of data in flight and client

peer authentication. Also, covers the circumstances under which such recommendations can be bypassed.

Will have to address concerns about performance effects.

*Consensus Item #47 concerns dubious paragraph regarding AUTH_NONE in SECINFO response which should be deleted if there are no compatibility issues that make that impossible.

Will have to ascertain whether this is in fact supported by any server or relied on by any client.

*Consensus Item #35 discusses possible work on future security needs.

Need to discuss whether there is too much or too little forward-looking material.

*Consensus Item #53 concerns the possible use of multi-factor authentication.

*Consensus Item #54 discusses prevention of code on a compromised client from hijacking the client machine's peer authentication.

This relates to getting clarity about the safety of the client peer authentication available in rpc-with-tls.

*Consensus Item #55 discusses issues related to potential use of Kerberos in cloud environments.

We will have to be sure we have a strog case here, since ther may be a feeling within the IESG that AUTH_SYS is per se unacceprable.

A.1.2. ACL-Related Changes

In addition to the above there has been a respecfication of ACL-related features appearing in the companion document [I-D.dnoveck-nfsv4-acls] and summarized in Section 6.2.

Relevant consensus items relating to this respecfication are discussed in the companion docuent.

A.2. Need for Clarifying Changes

The need to make the major changes discussed in Appendix A.1 has meant that much text dealing with security has needed to be significantly revised or rewritten. As a result of that process, many issues involving unclear, inconsistent, or otherwise inappropriate text were uncovered and now need to be dealt with.

While the author believes that changes in this situation are necessary, the fact that we are changing a document adopted by consensus requires the working group to be clear about the acceptability of the changes. This applies to both the troublesome issues discussed in Section 3.4 and to the other changes included below.

The primary source of these not-clearly specified elements is the disparate goals of many participants in the specification process:

- For many participants, the ACL definition needed to include a large part of Windows-oriented semantics, including elements clearly distinct from the Unix semantics supported by earlier versions of NFS.

- For other participants, such semantic extensions were of little interest.

While the author believes such changes are necessary, the fact that we are changing a document adopted by consensus requires the working group to be clear about the acceptability of the changes. This applies to both the troublesome issues discussed in Section 3.4 and to the other changes included below.

Because of the need for re-organization of subjects relating to authorization, the ordering of the list follows the text of the current version which may differ considerably from that in earlier versions of the I-D.

The subjects related to authorization needed to be reorganized because of the following complexities:

- There are three kinds of authorization to deal with.

- The most important kind of authorization, file access authorization, is controlled by mode attributes and a set of ACL-related attributes, creating multiple authorization models and a need to coordinate them.

- ACLs, besides their prominent role with regard to file access authorization, provide control of alarm and audit facilities.

As a result, these matters are dealt with in the eight top-level sections within this document and the associated ACL document from 5 through 13. Of these, three sections, Sections 6, 9 and 13, have become brief overiews wjich introduce matters dealt with in more detail in the companion ACL document [I-D.dnoveck-nfsv4-acls].

- Section 5,a new top-level section, describes all the authorization-related attributes. Although this section cover all

of the attributes, details relating to the ACL-related ones appear in the companion ACL document.

*Section 6, a new top-level section, provide an introduction to the structure of ACLs with the corresponding detail appearing in the companion ACL document [I-D.dnoveck-nfsv4-acls]. Later their uses are described later sections 8 through 10 and in Section 13 of which the last two serve as introductions to matters dealt with in the companion ACL document.

*In Section 7, there is a discussion of authorization in general. This includes a discussion of user-based file authentication, NFSv4's approach to mandatory access control, and a discussion of locking state authorization, each of which is described in later top-level sections.

*File access authorization is described primarily in Section 8. Because this authorization can be controlled by either the mode attribute or various ACL-related attributes, there is an introduction to these authorization- determining attributes in Section 8.1.

Because file access authorization can be controlled by the posix-related authorization attributed decribed in Section 5.3 and by ACL-related attributes described in Section 5.4, there is an introduction to how these relate in Section 8.2 followed by discussion of the semantics of the two kinds of supported ACLs in Section 8.4.

Issues related to the existence of two separate file access are dealt with in Sections 9, dealing with the similarities between these two, and 10 dealing with their co-existence and potential conflict.

*Sections 11 through 13 deal with, labelled NFS authorization, state modification authorization, and the uses of ACLs outside authorization.

Beyond the reorganization described above, there are numerous matters to be dealt with, reflecting the previous inability to clearly specify two sets of authorization behaviors and their potential interactions.

*Also in Section 8.2, there is added clarity in the discussion of support for multiple authorization approaches by eliminating use of the subjective term "reasonable semantics".

In connection with this clarification, we have switched from describing the needed co-ordination between modes and acls as

"goals" to describing them as "requirements" to give clients some basis for expecting interoperability in handling these issues.

As a result of this shift to a prescriptive framework applying to all minor versions it becomes possible to treat all minor versions together. In existing RFCs and some earlier versions of this document, it had been assumed that NFSv4.0 was free to ignore the relevant prescriptions first put forth in RFC 5661 and only needed to address the "goals" for this co-ordination.

A.3. Addressing the Need for Clarifying Changes

Unlike the issues discussed in Appendix A.1, for which the path to make the necessary changes is laid out in that same section, the changes discussed in Appendix A.2, are discussed here, because resolving them is a more involved matter.

The difficulty derives principally from the fact that there are existing clients and server with very different approaches to these matters, making it difficult or impossible to reach a consensus that will make some existing clients non-compliant.

In such a situation, we need to allow a set of possibitlities, as has been done for internationalization of file names in [RFC7530] for NFSv4.0 and in [I-D.ietf-nfsv4-internationalization], for all minor versions.

As in the case of the internationalization of file names, we will have to accommodate a range of possible server implementations, but the task will, in this case, involve a lot more work. To see why, we will first explore the situation for internationalization of file names and later see how the situation for these issues poses greater difficulties and how they might be addressed. Some of these dificulties match other internationalization issues, such as the handling of internationalized domain names and the proper treatment of case-insensitive handling of file names that dealt with incorrectly or not at all in [RFC7530].

* The treatment of internationalization of file names in [RFC7530] and [RFC5661] was so divorced from the needs of implementations that it was necessary to be ignore it.

  On the other hand, the treatment of internationalization of file names in the obsoleted document [RFC3010] was available as a base and had been implemented by many clients and servers.

* The vast majority of clients and servers implemented the same approach to internationalization, even though it was not limited to the use of UTF-8.

Despite the general aversion to this approach within the IESG, it was accepted, most likely because we were describing NFSv4 internationalization on "as-implemented" basis.

*Overall, we wound up allowing a total of three sorts of server behaviors, two of which were UTF-8-aware.

One UTF-8-aware option that had been implemented, simply considred canonically equivalent names as deignating the same file with changing the name to a different canonically equivalent name. Although this approach diverged from IESG expectations regarding normalization, it was accepted.

Another UTF-8-aware option was allowed, despite the absence of any implementations. It allowed servers to normalize file names while forbidding them from rejecting unormalized names. This was allowed because it would cause no difficulties for clients and because forbidding it might have caused problems for IESG acceptance.

*We were able, where necessary, to give the client the ability to detrmine which form of internationalization supported provided by the server for any particular file system, using an **OPTIONAL** attribute made available in NFSv4.1.

The situation with regard to the clarifying changes needed in this document is different for the reasons listed below.

*Rather than three valid choices for acceptable server behavior, with authorization we have a vast number.

Given that there are nine ACE mask bits, each of whick might not be supported, there are at least 512 possible valid behaviors, even if none of the dubious instances of **SHOULD** creates additional valid behavioral patterns.

Added to that are multiple (at least two) ways of mapping ACLs to modes and at leat three different behaviors in mdifying ACLs to reflect changes in mode.

As a result, existing RFCs allow at least three thousand different server behaviors.

*There is little information available about the particular behaviors manifested by existing server-fs combinations.

It may be difficult to get this informarion because the implementations were done by people no longer participating in the working or who were never involved in the working group

* There are no existing attributes that might be used to communicates server/fs choices to the client.

This creates a difficult situation. To fully resolve it will require substantial work as follows:

* Determining whether each of the dubious instances of "**SHOULD**" is in fact relied on by existing servers or might be needed by future servers.

* Determining which ACE mask bits are supported by all existing servers or not supported by any existing server to further limit the set of **OPTIONAL** features that clients need to be made aware of.

  There might also be cases where some set of mask bits are are always either supported or unsupported together.

* Determining the set of mappings from acls to modes that are actually and making each such **OPTIONAL** as opposed to the current situation in which an "intentional" "**SHOULD**" is used despite the mismatch between this use and [RFC2119].

* Making a similar determination of the actions actually applied to acl when the mode is changed.

Once we have the above information, it will be clear how we could create adress these issues inn either of the ways discused below:

* Have a way of determining whether support of NFSv4 ACL semaniics or UNIX ACL semantics is provided by any particular file system, such as that discussed in consensus item #61.

  This approach wiuld address the motive for the current under-specfication but would not address possible negative coonsequences arising from that under-specification.

  A further difficulty with it is that it does not address the possibility of hybrids of the two models.

* providing an appropriate extension as described in Section 6.2 to provide clients with an adequate description of expected server behavior. This document will be limited to providing client implementation guidance about dealing with the uncertinty in cases in which new attributes is not available. The situation will be similar to that described in Section 15 of [I-D.ietf-nfsv4-internationalization].

Appendix B. Issues for which Consensus Needs to be Ascertained

This section, togther with a corresponding Appendix in the ACL document, helps to keep track of specific changes which the author has made or intends to make to deal with issues found in RFCs 7530 and 8881.

The changes listed exclude those that only apply to the ACL document but does include cases in which there might need co-ordination between the two documents. The changes listed here exclude those which are clearly editorial but include some that the author believes are editorial but for which the issues are sufficiently complicated that working group consensus on the issue is probably necessary.

These changes are presented in the table below, organized into a set of "Consensus Items" identified by the numeric code appearing in annotations in the proposed document text. For each such item, a type code is assigned with separate sets of code define for pending items and for those which are no longer pending.

This document and the companion ACL document use a shared set of nmeric ids to identify Cinsensus Item, Some of these were originally defined in this document and subsequently moved while others were created as part of the ACL document. In order to maintain a common list without requiring excessive document cooridnation the follow practices and consquences should be noted.

* Before the documents were split the numbers from one to sixty-two were assigned to Consensus Items while some ofthese were moved to the ACL document.

  Item numbers in this range might appear in either document, although each number will appear in only a single document.

* New issues will receive a number based on the document in which the issue appears:

  Item numbers sixty-two through ninety-nine will be assigned to issues in this document while those above one-hundred will be used in the ACL document.

* As part of the documents being split some items that apply to both were split, with the portion appearing in this document retaining the old number a new number (over one-hundred) being assigned to the ACL-related portion.

  In such cases, the related document items in the two document will refer to one anothe.

The following type codes are defined for pending consensus items:

*"NM" denotes a change which is new material that is not purely editorial and thus requires Working Group consensus for eventual publication.

*"BE" denotes a change which the author believes is editorial but for which the change is sufficiently complex that the judgment is best confirmed by the Working Group.

*"BC" denotes a change which is a substantive change that the author believes is correct. This does not exclude the possibility of compatibility issues becoming an issue but is used to indicate that the author believes any such issues are unlikely to prevent its eventual acceptance.

*"CI" denotes a change for which the potential for compatibility issues is a major concern with the expected result that working group discussion of change will focus on clarifying our knowledge of how existing clients and server deal with the issue and how they might be affected by the change or the change modified to accommodate them.

*"NS" denotes a change which represents the author's best effort to resolve a difficulty but for which the author is not yet confident that it will be adopted in its present form, principally because of the possibility of troublesome compatibility issues.

*"NE" denotes change based on an existing issue in the spec but for which the replacement text is incomplete and needs further elaboration.

*"WI" denotes a potential change based on an existing issue in the spec but for which replacement text is not yet available because further working group input is necessary before drafting. It is expected that replacement text will be available in a later draft once that discussion is done.

*"LD" denotes a potential change based on an existing issue in the spec but for which replacement text is not yet available due to the press of time. It is expected that replacement text will be available in a later draft.

*"EV" denote a potential change which is tentative or incomplete because further details need to be provide or because the author is unsure that he has a correct explanation of the issue. It is expected that replacement text will be available in a later draft.

The following codes are defined for consensus items which are no longer pending.

*"RT" designates a former item which has been retired, because it has been merged with another one or otherwise organized out of existence.

Such items no longer are referred to the document source although the item id is never reassigned. They are no longer counted among the set of total items.

*"CA" designates a former item for which consensus has been achieved in the judgment of the author, although not by any official process.

Items reaching this state are effected in the document source including the deletion of annotations and the elimination of obsoleted previous treatments.

Items in this state are still counted among the total of item but are no longer considered pending

*"CV" designates a former item for which consensus has been achieved and officially verified.

Even though the author is a Working Group co-chair,it is probably best if he is not involved in this process and intends to leave it to the other co-chairs, the Document Shepherd and the Area Director.

Items in this state are not counted among the item totals. They may be kept in the table but only to indicate that the item id is still reserved.

*"DR" designates a former item which has been dropped, because it appears that working group acceptance of it, even with some modification, is unlikely.

Such items no longer are referred to the document source although the item id is never reassigned. They are no longer counted among the set of total items.

When asterisk is appended to a state of "NM", "BC" or "BE" it that there has been adequate working group discussion leading one to reasonably expect it will be adopted, without major change, in a subsequent document revision.

Such general acceptance is not equivalent to a formal working group consensus and it not expected to result in major changes to the draft document,

On the other hand, once there is a working group consensus with regard to a particular issue, the document will be modified to remove associated annotations, with the previously conditional text appearing just as other document text does. The issue will remain in this table as a non-pending item. It will be mentioned in Appendices A.1 or A.2 to summarize the changes that have been made.

It is to be expected that these designations will change as discussion proceeds and new document versions are published. It is hoped that most such shifts will be upward in the above list or result in the deletion of a pending item, by reaching a consensus to accept or reject it. This would enable, once all items are dealt with, an eventual request for publication as an RFC, with this appendix having been deleted.

[Editor Aside]: The following Consensus Items have been moved, in their entirety, to the new acl spec: #3, #4, #5, #8, #9, #7, #10, #11, #12, #13, #14, #15, #16, #17, #26, #27, #28, #29, #30, #31, #50, #51, #6 1. They have retained their numeric existing numeric ids.

[Editor Aside]: The following Consensus Items had to be split between this document and the new acl spec, with a new id assiged for the portion in the acl spec: #6, #25, #56, #58. The new ids assigned are in the range between #62 and #65.

[Editor aside]: New items added to this spec after the split will be assigned id's between #66 and #99. New item s added to the ACL spec will be assigned ids in the range btween #100 and #199.

#	Type	...References...	Substance
1	CA	Assumed OK.	New approach distinguishing authetication and identification.
2	CA	Assumed OK.	Outline of new negotiation framework.
6	CI	#6a in S 5.3.1	New/revised description of the role of the "sticky bit" for directories, both with respect to ACL handling and mode handling. Needs to be considered together with Item #62 in the ACL document.
18	RT	#18a in S 8.1 #18b in S 8.2	Origially concerend with need for support of owner, owner_group. Has now been combined with item #57.
19	CI	#19a in S 8.2	Revised discussion of coordination of mode and the ACL-related attributes.
20	CI	#20 in S 5.3.1	More closely align ACL_based and mode-based semantics with regard to SVTX.
21	BC	#21a in S 4.1 #21b in S 5.3.1	Introduce the concept of reverse-slope modes and deal properly with them. The

#	Type	...References...	Substance
			decision as to the proper handling is addressed as Consensus Item #28.
22	BC	#22a in S 9.2	Revise treatment of divergences between ACL/mode authorization and server behavior, dividing the treatment between cases in which something authorized is still not allowed (OK), and those in which something not authorized is allowed (highly problematic from a security point of view).
23	BC	#23a in S 9.1 #23b in S 9.3	Revise discussion of client interpretation of ACLs and the use of ACCESS instead
24	BE	#24a in S 9.3	Delete bogus reference.
25	CI	#25a in S 3.3	Revised description of co-ordination of acl and mode attributes to apply to NFSv4 as a whole. While this includes many aspects of the shift to be more specific about the co-ordination requirements including addressing apparently unmotivated uses of the terms "**SHOULD**" and "**SHOULD NOT**", it excludes some arguably related matters dealt with as Consensus Items #26 and #27. Needs to be considered together with Item #63 in the ACL document.
32	BC	#32a in S 16 #32b in S 16.1 #32c in S 19 #32d in S 19.1 #32e in S 19.2	Expanded negotiation framework to accommodate multiple transport types and security services provided on a per-connection basis, i.e. encryption and peer authentication.
33	RT	Material formerly here moved to #32.	Reorganization of description of SECINFO op to apply to all minor versions. (Dropped)
34	RT	Superseded by simpler treatment.	Revision to NFSv4.0 SECINFO implementation section (Dropped).
35	NM	#35a in S 17.1	Now has preliminary work on future security needs.
36	CI	#36a in S 18.1.3	Threat analysis only dealing with RECOMMENDED behavior regarding use of per-connection security facilities and handling of AUTH_SYS.
37	CI	#37a in S 18.1.3	Threat analysis only dealing with RECOMMENDED behavior with regard to acl support including ACL/mode coordination.
38	CI	#38a in S 18.1.4	

#	Type	...References...	Substance
			Address the need to temporarily allow unsafe behavior mistakenly allowed by previous specifications
39	CI	#39a in S 18.1.5	Define new approach to AUTH_SYS.
40	CI	#40a in S 18.2.1 #40a in S 18.2.2	Discussion of scope for security considerations and the corresponding threat analysis.
41	CI	#41a in S 9.2 #41b in S 18.3.1 #41c in S 18.3.2 #41d in S 18.3.4	Discuss major new security recommendations regarding protection of data in flight and client peer authentication. Also, covers the circumstances under which such recommendations can be bypassed.
42	RT	#42a in S 18.4.5	Former placeholders for threat analysis subsections have now been superseded by new proposed subsections.
43	RT	#43a in S 18.4.6.1	
44	RT	#44a in S 18.4.6.2	
45	RT	#45a in S 18.4.7.1	
46	RT	#46a in S 18.4.7.2	
47	CI	gone for now.	Dubious paragraph regarding AUTH_NONE is SECINFO response which should be deleted if there are no compatibility issues that make that impossible.
48	RT	Superseded by simpler treatment.	Missing pieces of secinfo processing algorithm that didn't get done in -02.
49	RT	Superseded by simpler treatment.	Secinfo processing algorithm that was expected to be finished in -04.
52	NS	#52b in S 9.2 #52c in S 18.3.3	Eliminate superuser semantics as it had been, as valid by implication and essentially ignoring the new format for user ids. Also, deal with the security consequences of its inclusion appropriately.
53	EV	#53a in S 18.4.2	Discussion of possible adaptation of RPCSEC_GSS/Kerberos to multi-factor authentication.
54	EV	#54a in S 18.4.6.1	Discussion of prevention of code on a compromised client from hijacking the client machine's peer authentication.
55	EV	#55a in S 18.4.6.1	Discussion of issues with potential use of Kerberos in cloud environments
56	NI	#56a in S 4.1	

#	Type	...References...	Substance
			Discussion of issues related to the handling of allowed variants of the NFSv4 ACL model, including subsets based on the Unix ACL model. Needs to be considered together with Item #64 in the ACL document.
57	BC	#57a in S 4.1 #57b in S 5 #57c in S 5.1 #57d in S 5.3 #57e in S 8.1	Designation of mode, owner, and owner_group as **REQUIRED** attributes. Eliminates the undue complexity and gratuitous interoperability problems providing implementation that do not provide POSIX authorization semantics, including those with no authorization support at all.
58	NM	#58a in S 5 #58e in S 5.5.1	Designation of the acl, dacl, sacl, and sec_label attributes as Experimental, rather than **OPTIONAL**. Note that this is separate from the possibility of sufficiently clarifying the description of the acl, dacl, and sacl attributes to make the Experimental designation unnecessary, which will be covered as Item #XX. Needs to be considered together with Item #65 in the ACL document.
59	NM	#59a in S 5.1	Major clarification of the handling of id strings, including removal of denigration of the numeric form.
60	BC	#60a in S 5.1	Addressng properly issue of special users such as root snd nobody in the context of id strings.
66	NE	#66a in S 5.3.5 #66b in S 5.3.6	Discussion needed of authorization semantics for access to named attributes. Needs to be coorinated with handling of Item #100, addressed in the companion ACL document.

Table 2

The following table summarizes the issues in each particular pending state.

Type	Cnt	Issues
BC	6	21, 22, 23, 32, 57, 60
BE	1	24
CI	11	6, 19, 20, 25, 36, 37, 38, 39, 40, 41, 47

Type	Cnt	Issues
EV	3	53, 54, 55
NE	1	66
NI	1	56
NM	3	35, 58, 59
NS	1	52
All	27	Grand total for above table.

Table 3

The following table summarizes the issues in each particular non-pending state..

Type	Cnt	Issues
CA	2	1, 2
RT	9	18, 33, 34, 42, 43, 44, 45, 46, 48
All	11	Grand total for above table.

Table 4

Acknowledgments

The author wishes to thank Tom Haynes for his helpful suggestion to deal with security for all NFSv4 minor versions in the same document.

The author wishes to thank Chris Inacio for pointing out the dificulties created by the existing handlin of name mapping.

The author wishes to thank Brian Pawlowski for his helpful insights into the history of NFsv4 security handling.

The author wishes to draw people's attention to Nico Williams' remark that NFSv4 security was not so bad, except that there was no provision for authentication of the client peer. This perceptive remark, which now seems like common sense, did not seem so when made, but it has served as a beacon for those working on putting NFSv4 security on a firmer footing. We appreciate Nico's perceptive guidance.

The author wishes to thank Bruce Fields for his helpful comments regarding ACL support which had a major role in the evolution of this document.

The author wishes to acknowledge the important role of the authors of RPC-with-TLS, Chuck Lever and Trond Myklebust, in moving the NFS security agenda forward and thank them for all their efforts to improve NFS security.

The author wishes to thank Chuck Lever for his many helpful comments about nfsv4 security issues, his explanation of many unclear points,

and much important guidance he provided that is reflected in this document, including his work to streamline the security negotiation process by the definition of new pseudo-flavors.

The author wishes to thank Rick Macklem for his help in resolving NFSv4 security issues. These include clarifying possible server policies regarding RPC-with-TLS, helping to clarify "owner-override semantics" and bringing to the Working Group's attention the possibility of deriving limited principal identification from client peer authentication while still staying within the boundaries of RPC-with-TLS.

Author's Address

David Noveck (editor)
NetApp
201 Jones Road, Suite 16
Waltham, MA 02451
United States of America

Phone: +1-781-572-8038
Email: davenoveck@gmail.com