Network Working Group
Internet-Draft
Document: draft-ietf-nfsv4-secinfo-02.txt

M. Eisler
Editor
Network Appliance, Inc.
July 2004

NFSv4.1: SECINFO Changes

Status of this Memo

ABSTRACT

This document proposes some changes to security negotiation in NFS version 4 [RFC3530]. It is hoped, but not promised, that these changes will be part of a new minor version of NFS: NFSv4.1.

TABLE OF CONTENTS

1. Introduction

This document assumes understanding of the NFSv4.0 specification.

The NFSv4.0 specification contains three oversights and ambiguities with respect to the SECINFO operation.

First, it is impossible for the client to use the SECINFO operation to determine the correct security triple for accessing a parent directory. This is because SECINFO takes as arguments the current file handle and a component name. However, NFSv4.0 uses the LOOKUPP operation to get the parent directory of the current file handle. If the client uses the wrong security when issuing the LOOKUPP, and gets back an NFS4ERR_WRONGSEC error, SECINFO is useless to the client. The client is left with guessing which security the server will accept. This defeats the purpose of SECINFO, which was to provide an efficient method of negotiating security.

Second, there is ambiguity as to what the server should do when it is passed a LOOKUP operation such that the server restricts access to the current file handle with one security triple, and access to the component with a different triple, and remote procedure call uses one of the two security triples. Should the server allow the LOOKUP?

Third, there is a problem as to what the client must do (or can do), whenever the server returns NFS4ERR_WRONGSEC in response to a PUTFH operation. The NFSv4.0 specification says that client should issue a SECINFO using the parent filehandle and the component name of the filehandle that PUTFH was issued with. This may not be convenient for the client.

This document resolves the above three issues in the context of NFSv4.1.

2. Modified Operation 16: LOOKUPP - Lookup Parent Directory

If the NFSv4 minor version is 1, then following replaces section 14.2.14 of the NFSv4.0 specification. The LOOKUPP operation's "over

the wire" format is not altered, but the semantics are slightly modified to account for the addition of SECINFO_NO_NAME.

SYNOPSIS

```
(cfh) -> (cfh)
```

ARGUMENT

```
/* CURRENT_FH: object */
void;
```

RESULT

```
struct LOOKUPP4res {
            /* CURRENT_FH: directory */
            nfsstat4        status;
};
```

DESCRIPTION

The current filehandle is assumed to refer to a regular directory or a named attribute directory. LOOKUPP assigns the filehandle for its parent directory to be the current filehandle. If there is no parent directory an NFS4ERR_NOENT error must be returned. Therefore, NFS4ERR_NOENT will be returned by the server when the current filehandle is at the root or top of the server's file tree.

As for LOOKUP, LOOKUPP will also cross mountpoints.

If the current filehandle is not a directory or named attribute directory, the error NFS4ERR_NOTDIR is returned.

If the requester's security flavor does not match that configured for the parent directory, then the server SHOULD return NFS4ERR_WRONGSEC (a future minor revision of NFSv4 may upgrade this to MUST) in the LOOKUPP response. However, if the server does so, it MUST support the new SECINFO_NO_NAME operation, so that the client can gracefully determine the correct security flavor. See the discussion of the SECINFO_NO_NAME operation for a description.

ERRORS

NFS4ERR_ACCESS
NFS4ERR_BADHANDLE
NFS4ERR_FHEXPIRED
NFS4ERR_IO
NFS4ERR_MOVED
NFS4ERR_NOENT
NFS4ERR_NOFILEHANDLE
NFS4ERR_NOTDIR
NFS4ERR_RESOURCE

```
     NFS4ERR_SERVERFAULT
     NFS4ERR_STALE
     NFS4ERR_WRONGSEC
```

3. Modified Operation 33: SECINFO - Obtain Available Security

If the NFSv4 minor version is 1, then following replaces section 14.2.31 of the NFSv4.0 specification. The SECINFO operation's "over the wire" format is not altered, but the semantics are slightly modified to account for the addition of SECINFO_NO_NAME.

SYNOPSIS

```
     (cfh), name -> { secinfo }
```

ARGUMENT

```
     struct SECINFO4args {
          /* CURRENT_FH: directory */
          component4     name;
     };
```

RESULT

```
     enum rpc_gss_svc_t {/* From RFC 2203 */
          RPC_GSS_SVC_NONE        = 1,
          RPC_GSS_SVC_INTEGRITY   = 2,
          RPC_GSS_SVC_PRIVACY     = 3
     };

     struct rpcsec_gss_info {
          sec_oid4        oid;
          qop4            qop;
          rpc_gss_svc_t   service;
     };

     union secinfo4 switch (uint32_t flavor) {
     case RPCSEC_GSS:
           rpcsec_gss_info        flavor_info;
     default:
           void;
     };

     typedef secinfo4 SECINFO4resok<>;

     union SECINFO4res switch (nfsstat4 status) {
     case NFS4_OK:
           SECINFO4resok resok4;
     default:
           void;
     };
```

DESCRIPTION

The SECINFO operation is used by the client to obtain a list of valid RPC authentication flavors for a specific directory filehandle, file name pair. SECINFO should apply the same access methodology used for LOOKUP when evaluating the name. Therefore, if the requester does not have the appropriate access to LOOKUP the name then SECINFO must behave the same way and return NFS4ERR_ACCESS.

The result will contain an array which represents the security mechanisms available, with an order corresponding to the server's preferences, the most preferred being first in the array. The client is free to pick whatever security mechanism it both desires and supports, or to pick in the server's preference order the first one it supports. The array entries are represented by the secinfo4 structure. The field 'flavor' will contain a value of AUTH_NONE, AUTH_SYS (as defined in [RFC1831]), or RPCSEC_GSS (as defined in [RFC2203]). The field flavor can also any other security flavor registered with IANA.

For the flavors AUTH_NONE and AUTH_SYS, no additional security information is returned. The same is true of many (if not most) other security flavors, including AUTH_DH. For a return value of RPCSEC_GSS, a security triple is returned that contains the mechanism object id (as defined in [RFC2743]), the quality of protection (as defined in [RFC2743]) and the service type (as defined in [RFC2203]). It is possible for SECINFO to return multiple entries with flavor equal to RPCSEC_GSS with different security triple values.

On success, the current filehandle retains its value.

If the name has a length of 0 (zero), or if name does not obey the UTF-8 definition, the error NFS4ERR_INVAL will be returned.

IMPLEMENTATION

The SECINFO operation is expected to be used by the NFS client when the error value of NFS4ERR_WRONGSEC is returned from another NFS operation. This signifies to the client that the server's security policy is different from what the client is currently using. At this point, the client is expected to obtain a list of possible security flavors and choose what best suits its policies.

As mentioned, the server's security policies will determine when a client request receives NFS4ERR_WRONGSEC. The operations which may receive this error are: LINK, LOOKUP, LOOKUPP, OPEN, PUTFH, PUTPUBFH, PUTROOTFH, RESTOREFH, RENAME, and indirectly

READDIR. LINK and RENAME will only receive this error if the

security used for the operation is inappropriate for saved filehandle. With the exception of READDIR, these operations represent the point at which the client can instantiate a filehandle into the "current filehandle" at the server. The filehandle is either provided by the client (PUTFH, PUTPUBFH, PUTROOTFH) or generated as a result of a name to filehandle translation (LOOKUP and OPEN). RESTOREFH is different because the filehandle is a result of a previous SAVEFH. Even though the filehandle, for RESTOREFH, might have previously passed the server's inspection for a security match, the server will check it again on RESTOREFH to ensure that the security policy has not changed.

If the client wants to resolve an error return of NFS4ERR_WRONGSEC, the following will occur:

o For LOOKUP and OPEN, the client will use SECINFO with the same current filehandle and name as provided in the original LOOKUP or OPEN to enumerate the available security triples.

o For LINK, PUTFH, PUTROOTFH, PUTPUBFH, RENAME, and RESTOREFH, the client will use SECINFO_NO_NAME { style = current_fh }. The client will prefix the SECINFO_NO_NAME operation with the appropriate PUTFH, PUTPUBFH, or PUTROOTFH operation that provides the file handled originally provided by the PUTFH, PUTPUBFH, PUTROOTFH, or RESTOREFH, or for the failed LINK or RENAME, the SAVEFH.

  ==========
  NOTE: In NFSv4.0, the client was required to use SECINFO, and had to reconstruct the parent of the original file handle, and the component name of the original filehandle.
  ==========

o For LOOKUPP, the client will use SECINFO_NO_NAME { style = parent } and provide the filehandle with equals the filehandle originally provided to LOOKUPP.

The READDIR operation will not directly return the NFS4ERR_WRONGSEC error. However, if the READDIR request included a request for attributes, it is possible that the READDIR request's security triple did not match that of a directory entry. If this is the case and the client has requested the rdattr_error attribute, the server will return the NFS4ERR_WRONGSEC error in rdattr_error for the entry.

See the section "Security Considerations" for a discussion on

the recommendations for security flavor used by SECINFO and

SECINFO_NO_NAME.

ERRORS
NFS4ERR_ACCESS
NFS4ERR_BADCHAR
NFS4ERR_BADHANDLE
NFS4ERR_BADNAME
NFS4ERR_BADXDR
NFS4ERR_FHEXPIRED
NFS4ERR_INVAL
NFS4ERR_MOVED
NFS4ERR_NAMETOOLONG
NFS4ERR_NOENT
NFS4ERR_NOFILEHANDLE
NFS4ERR_NOTDIR
NFS4ERR_RESOURCE
NFS4ERR_SERVERFAULT
NFS4ERR_STALE

4. New Operation 40: SECINFO_NO_NAME - Get Security on Unnamed Object

SYNOPSIS
(cfh), secinfo_style -> { secinfo }

ARGUMENT
```
enum secinfo_style_4 {
    current_fh = 0,
    parent = 1
};

typedef secinfo_style_4 SECINFO_NO_NAME4args;
```

RESULT
```
typedef SECINFO4res SECINFO_NO_NAME4res;
```

DESCRIPTION
Like the SECINFO operation, SECINFO_NO_NAME is used by the client to obtain a list of valid RPC authentication flavors for a specific file object. Unlike SECINFO, SECINFO_NO_NAME only works with objects are accessed by file handle.

There are two styles of SECINFO_NO_NAME, as determined by the value of the secinfo_style_4 enumeration. If "current_fh" is passed, then SECINFO_NO_NAME is querying for the required security for the current filehandle. If "parent" is passed, then SECINFO_NO_NAME is querying for the required security of the current filehandles's parent. If the style selected is "parent", then SECINFO should apply the same access methodology used for

LOOKUPP when evaluating the traversal to the parent directory. Therefore, if the requester does not have the appropriate access to LOOKUPP the parent then SECINFO_NO_NAME must behave the same way and return NFS4ERR_ACCESS.

Note that if PUTFH, PUTPUBFH, or PUTROOTFH return NFS4ERR_WRONGSEC, this is tantamount to the server asserting that the client will have to guess what the required security is, because there is no way to query. Therefore, the client must iterate through the security triples available at the client and reattempt the PUTFH, PUTROOTFH or PUTPUBFH operation. In the unfortunate event none of the MANDATORY security triples are supported by the client and server, the client SHOULD try using others that support integrity. Failing that, the client can try using other forms (e.g. AUTH_SYS and AUTH_NONE), but because such forms lack integrity checks, this puts the client at risk.

The server implementor should pay particular attention to the section "Clarification of Security Negotiation in NFSv4.1" for implementation suggestions for avoiding NFS4ERR_WRONGSEC error returns from PUTFH, PUTROOTFH or PUTPUBFH.

Everything else about SECINFO_NO_NAME is the same as SECINFO. See the previous discussion on SECINFO.

IMPLEMENTATION

See the previous discussion on SECINFO.

ERRORS

NFS4ERR_ACCESS
NFS4ERR_BADCHAR
NFS4ERR_BADHANDLE
NFS4ERR_BADNAME
NFS4ERR_BADXDR
NFS4ERR_FHEXPIRED
NFS4ERR_INVAL
NFS4ERR_MOVED
NFS4ERR_NAMETOOLONG
NFS4ERR_NOENT
NFS4ERR_NOFILEHANDLE
NFS4ERR_NOTDIR
NFS4ERR_RESOURCE
NFS4ERR_SERVERFAULT
NFS4ERR_STALE

5. Clarification of Security Negotiation in NFSv4.1

This section attempts to clarify NFSv4.1 security negotiation issues.

Unless noted otherwise, for any mention of PUTFH in this section, the reader should interpret it as applying to PUTROOTFH and PUTPUBFH in addition to PUTFH.

5.1. PUTFH + LOOKUP

The server implementation may decide whether to impose any restrictions on export security administration. There are at least three approaches (Sc is the flavor set of the child export, Sp that of the parent),

a. Sc <= Sp (<= for subset)

b. Sc ^ Sp != {} (^ for intersection, {} for the empty set)

c. free form

To support b (when client chooses a flavor that is not a member of Sp) and c, PUTFH must NOT return NFS4ERR_WRONGSEC in case of security mismatch. Instead, it should be returned from the LOOKUP that follows.

Since the above guideline does not contradict a, it should be followed in general.

5.2. PUTFH + LOOKUPP

Since SECINFO only works its way down, there is no way LOOKUPP can return NFS4ERR_WRONGSEC without the server implementing SECINFO_NO_NAME. SECINFO_NO_NAME solves this issue because via style "parent", it works in the opposite direction as SECINFO (component name is implicit in this case).

5.3. PUTFH + SECINFO

This case should be treated specially.

A security sensitive client should be allowed to choose a strong flavor when querying a server to determine a file object's permitted security flavors. The security flavor chosen by the client does not have to be included in the flavor list of the export. Of course the server has to be configured for whatever flavor the client selects, otherwise the request will fail at RPC authentication.

In theory, there is no connection between the security flavor used by SECINFO and those supported by the export. But in practice, the client may start looking for strong flavors from those supported by the export, followed by those in the mandatory set.

5.4. PUTFH + Anything Else

PUTFH must return NFS4ERR_WRONGSEC in case of security mismatch. This is the most straightforward approach without having to add NFS4ERR_WRONGSEC to every other operations.

PUTFH + SECINFO_NO_NAME (style "current_fh") is needed for the client to recover from NFS4ERR_WRONGSEC.

6. RPC Definition File Changes

```
/*
 * Copyright (C) The Internet Society (2004)
 * All Rights Reserved.
 */

/*
 * nfs41_prot.x
 */

%/* $Id: nfs41_prot.x,v 1.2 2004/06/18 23:19:28 mre Exp $ */

/* new operation, SECINFO_NO_NAME */

enum secinfo_style_4 {
    current_fh = 0,
    parent = 1
};

typedef secinfo_style_4 SECINFO_NO_NAME4args;
typedef SECINFO4res SECINFO_NO_NAME4res;


/*
 * Operation arrays
 */

enum nfs_opnum4 {
   OP_ACCESS               = 3,
   OP_CLOSE                = 4,
   OP_COMMIT               = 5,
   OP_CREATE               = 6,
   OP_DELEGPURGE           = 7,
   OP_DELEGRETURN          = 8,
   OP_GETATTR              = 9,
   OP_GETFH                = 10,
   OP_LINK                 = 11,
   OP_LOCK                 = 12,
   OP_LOCKT                = 13,
```

```
      OP_LOCKU                = 14,
      OP_LOOKUP               = 15,
      OP_LOOKUPP              = 16,
      OP_NVERIFY              = 17,
      OP_OPEN                 = 18,
      OP_OPENATTR             = 19,
      OP_OPEN_CONFIRM         = 20,
      OP_OPEN_DOWNGRADE       = 21,
      OP_PUTFH                = 22,
      OP_PUTPUBFH             = 23,
      OP_PUTROOTFH            = 24,
      OP_READ                 = 25,
      OP_READDIR              = 26,
      OP_READLINK             = 27,
      OP_REMOVE               = 28,
      OP_RENAME               = 29,
      OP_RENEW                = 30,
      OP_RESTOREFH            = 31,
      OP_SAVEFH               = 32,
      OP_SECINFO              = 33,
      OP_SETATTR              = 34,
      OP_SETCLIENTID          = 35,
      OP_SETCLIENTID_CONFIRM  = 36,
      OP_VERIFY               = 37,
      OP_WRITE                = 38,
      OP_RELEASE_LOCKOWNER    = 39,
      OP_SECINFO_NO_NAME      = 40,
      OP_ILLEGAL              = 10044
   };

   union nfs_argop4 switch (nfs_opnum4 argop) {
   case OP_ACCESS:        ACCESS4args opaccess;
   case OP_CLOSE:         CLOSE4args opclose;
   case OP_COMMIT:        COMMIT4args opcommit;
   case OP_CREATE:        CREATE4args opcreate;
   case OP_DELEGPURGE:    DELEGPURGE4args opdelegpurge;
   case OP_DELEGRETURN:   DELEGRETURN4args opdelegreturn;
   case OP_GETATTR:       GETATTR4args opgetattr;
   case OP_GETFH:         void;
   case OP_LINK:          LINK4args oplink;
   case OP_LOCK:          LOCK4args oplock;
   case OP_LOCKT:         LOCKT4args oplockt;
   case OP_LOCKU:         LOCKU4args oplocku;
   case OP_LOOKUP:        LOOKUP4args oplookup;
   case OP_LOOKUPP:       void;
   case OP_NVERIFY:       NVERIFY4args opnverify;
   case OP_OPEN:          OPEN4args opopen;
   case OP_OPENATTR:      OPENATTR4args opopenattr;
```

```
   case OP_OPEN_CONFIRM:  OPEN_CONFIRM4args opopen_confirm;
```

```
   case OP_OPEN_DOWNGRADE:        OPEN_DOWNGRADE4args opopen_downgrade;
   case OP_PUTFH:         PUTFH4args opputfh;
   case OP_PUTPUBFH:      void;
   case OP_PUTROOTFH:     void;
   case OP_READ:          READ4args opread;
   case OP_READDIR:       READDIR4args opreaddir;
   case OP_READLINK:      void;
   case OP_REMOVE:        REMOVE4args opremove;
   case OP_RENAME:        RENAME4args oprename;
   case OP_RENEW:         RENEW4args oprenew;
   case OP_RESTOREFH:     void;
   case OP_SAVEFH:        void;
   case OP_SECINFO:       SECINFO4args opsecinfo;
   case OP_SETATTR:       SETATTR4args opsetattr;
   case OP_SETCLIENTID:   SETCLIENTID4args opsetclientid;
   case OP_SETCLIENTID_CONFIRM:   SETCLIENTID_CONFIRM4args
                                      opsetclientid_confirm;
   case OP_VERIFY:        VERIFY4args opverify;
   case OP_WRITE:         WRITE4args opwrite;
   case OP_RELEASE_LOCKOWNER:     RELEASE_LOCKOWNER4args
                                  oprelease_lockowner;
   case OP_SECINFO_NO_NAME:       SECINFO_NO_NAME4args
                                  opsecinfo_no_name;
   case OP_ILLEGAL:       void;
   };

   union nfs_resop4 switch (nfs_opnum4 resop){
   case OP_ACCESS:        ACCESS4res opaccess;
   case OP_CLOSE:         CLOSE4res opclose;
   case OP_COMMIT:        COMMIT4res opcommit;
   case OP_CREATE:        CREATE4res opcreate;
   case OP_DELEGPURGE:    DELEGPURGE4res opdelegpurge;
   case OP_DELEGRETURN:   DELEGRETURN4res opdelegreturn;
   case OP_GETATTR:       GETATTR4res opgetattr;
   case OP_GETFH:         GETFH4res opgetfh;
   case OP_LINK:          LINK4res oplink;
   case OP_LOCK:          LOCK4res oplock;
   case OP_LOCKT:         LOCKT4res oplockt;
   case OP_LOCKU:         LOCKU4res oplocku;
   case OP_LOOKUP:        LOOKUP4res oplookup;
   case OP_LOOKUPP:       LOOKUPP4res oplookupp;
   case OP_NVERIFY:       NVERIFY4res opnverify;
   case OP_OPEN:          OPEN4res opopen;
   case OP_OPENATTR:      OPENATTR4res opopenattr;
   case OP_OPEN_CONFIRM:  OPEN_CONFIRM4res opopen_confirm;
   case OP_OPEN_DOWNGRADE:        OPEN_DOWNGRADE4res opopen_downgrade;
   case OP_PUTFH:         PUTFH4res opputfh;
   case OP_PUTPUBFH:      PUTPUBFH4res opputpubfh;
```

```
   case OP_PUTROOTFH:     PUTROOTFH4res opputrootfh;
```

```
   case OP_READ:          READ4res opread;
   case OP_READDIR:       READDIR4res opreaddir;
   case OP_READLINK:      READLINK4res opreadlink;
   case OP_REMOVE:        REMOVE4res opremove;
   case OP_RENAME:        RENAME4res oprename;
   case OP_RENEW:         RENEW4res oprenew;
   case OP_RESTOREFH:     RESTOREFH4res oprestorefh;
   case OP_SAVEFH:        SAVEFH4res opsavefh;
   case OP_SECINFO:       SECINFO4res opsecinfo;
   case OP_SETATTR:       SETATTR4res opsetattr;
   case OP_SETCLIENTID:   SETCLIENTID4res opsetclientid;
   case OP_SETCLIENTID_CONFIRM:   SETCLIENTID_CONFIRM4res
                                      opsetclientid_confirm;
   case OP_VERIFY:        VERIFY4res opverify;
   case OP_WRITE:         WRITE4res opwrite;
   case OP_RELEASE_LOCKOWNER:     RELEASE_LOCKOWNER4res
                                  oprelease_lockowner;
   case OP_SECINFO_NO_NAME:       SECINFO_NO_NAME4res
                                  opsecinfo_no_name;
   case OP_ILLEGAL:       ILLEGAL4res opillegal;
   };

   struct COMPOUND4args {
      utf8str_cs      tag;
      uint32_t        minorversion; /* == 1 !!! */
      nfs_argop4      argarray<>;
   };

   struct COMPOUND4res {
      nfsstat4 status;
      utf8str_cs      tag;
      nfs_resop4      resarray<>;
   };
```

7. Security Considerations

The security considerations of NFSv4.0 apply to NFSv4.1, with the proviso that security considerations of SECINFO apply to the new operation, SECINFO_NO_NAME.

8. IANA Considerations

The IANA considerations of NFSv4.0 apply to NFSv4.1.

9. Acknowledgements

The basis for the text in this document comes from the NFSv4.0 specification as edited by Spencer Shepler. Peng Dai wrote the "Clarification of Security Negotiation in NFSv4.1" section. Sergey

Klyushin contributed to the discussion that led to this document. Mike Eisler proposed the SECINFO_NO_NAME operation to address the issues Sergey and Peng brought to the nfsv4 working group's attention. Carl Burnett reviewed an earlier draft of this document which resulted in substantial improvements.

10. Normative References

[RFC3530]
S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck, Internet RFC3530, "NFS version 4 Protocol", April, 2003.

[RFC1831]
R. Srinivasan, Internet RFC1831, "RPC: Remote Procedure Call Protocol Specification Version 2", August, 1995.

[RFC2743]
J. Linn, Internet RFC2743, "Generic Security Service Application Program Interface Version 2, Update 1", January, 2000.

[RFC2203]
M. Eisler, A. Chiu, L. Ling, Internet RFC2203, "RPCSEC_GSS Protocol Specification", September, 1997.

[RFC1832]
R. Srinivasan, Internet RFC1832, "XDR: External Data Representation Standard", August, 1995.

11. Informative References

None.

12. Editor's Address

Mike Eisler
5765 Chase Point Circle
Colorado Springs, CO 80919
USA

Phone: 719-599-9026
EMail: mike@eisler.com

Comments on this document should be sent to the NFSv4 working group: nfsv4@ietf.org

13. IPR Notices

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

14. Copyright Notice