       Allowing Inheritable NFSv4 Access Control Entries to Override the Umask

Abstract

   In many environments, inheritable NFSv4 Access Control Entries (ACEs)
   can be rendered ineffective by the application of the per-process
   file mode creation mask (umask). This can be addressed by
   transmitting the umask and create mode as separate pieces of data,
   allowing the server to make more intelligent decisions about the
   permissions to set on new files. This document proposes a protocol
   extension to accomplish that.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8275.

Table of Contents

1. Problem Statement

   On Unix-like systems, each process is associated with a file mode
   creation mask (umask). The umask specifies which permissions must be
   turned off when creating new file system objects.

   When applying the mode, Section 6.4.1.1 of [RFC7530] recommends that
   servers SHOULD restrict permissions granted to any user or group
   named in the Access Control List (ACL) to be no more than the
   permissions granted by the MODE4_RGRP, MODE4_WGRP, and MODE4_XGRP
   bits. Servers aiming to provide clients with Unix-like chmod
   behavior may also be motivated by the same requirements in [SUSv4].
   (See the discussion of additional and alternate access control
   mechanisms in "File Permissions", Section 4.4 of [SUSv4].)

   On many existing installations, all ordinary users use the same
   effective group ID by default. To prevent granting all users full
   access to each other's files, such installations usually default to a
   umask with very restrictive permissions. As a result, inherited ACL
   entries (inheritable ACEs) describing the permissions to be granted
   to named users and groups are often ignored. This makes inheritable
   ACEs useless in some common cases.

   Linux solves this problem on local file systems by ignoring the umask
   whenever a newly created file inherits ACEs from its parent; see
   [LinuxACL].

   The same solution should work for NFS. However, the NFSv4 protocol
   does not currently give the client a way to transmit the umask of the
   process opening a file. And clients have no way of atomically
   checking for inheritable permissions and applying the umask only when
   necessary. As a result, the server receives an OPEN with a mode
   attribute that already has the umask applied.

This document solves the problem by defining a new attribute that
allows the client to transmit umask and the mode specified at file
creation separately, allowing the client to ignore the umask in the
presence of inheritable ACEs. At least in the Linux case, this
allows NFSv4 to provide the same semantics available using local
access.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

3. Protocol Extension Considerations

This document presents an extension to minor version 2 of the NFSv4
protocol as described in [RFC8178]. It describes a new OPTIONAL
feature. NFSv4.2 servers and clients implemented without knowledge
of this extension will continue to interoperate with clients and
servers that are aware of the extension (whether or not they support
it).

Note that [RFC7862] does not define NFSv4.2 as non-extensible, so
[RFC8178] treats it as an extensible minor version. This Standards
Track RFC extends NFSv4.2 but does not update [RFC7862] or [RFC7863].

4. XDR Extraction

The additional lines of External Data Representation (XDR) [RFC4506]
description embedded in this document can be extracted by feeding
this document into the following shell script:

<CODE BEGINS>

```
#!/bin/sh
grep '^ *///' $* | sed 's?^ */// ??' | sed 's?^ *///$??'
```

<CODE ENDS>

That is, if the above script is stored in a file called "extract.sh",
and this document is in a file called "umask.txt", then the reader
can do:

```
sh extract.sh < umask.txt > umask.x
```

The effect of the script is to remove leading white space from each
line, plus a sentinel sequence of "///".

Once that extraction is done, these added lines need to be inserted
into an appropriate base XDR of the generated XDR from [RFC7863]
together with XDR from any additional extensions to be recognized by
the implementation. This will result in a ready-to-compile XDR file.

5. The mode_umask Attribute

   <CODE BEGINS>

```
///    struct mode_umask4 {
///       mode4  mu_mode;
///       mode4  mu_umask;
///    };
///
///    %/*
///    % * New For UMASK
///    % */
///    const FATTR4_MODE_UMASK         = 81;
```

   <CODE ENDS>

```
+------------+----+-------------+-----+------------+
| Name       | Id | Data Type   | Acc | Defined in |
+------------+----+-------------+-----+------------+
| mode_umask | 81 | mode_umask4 | W   | Section 5  |
+------------+----+-------------+-----+------------+
```

                              Table 1

The NFSv4.2 mode_umask attribute is based on the umask and on the
mode bits specified at open time, which together determine the mode
of a newly created UNIX file. Only the nine low-order mode4 bits of
mu_umask are defined. A server MUST return NFS4ERR_INVAL if bits
other than those nine are set.

The mode_umask attribute is only meaningful for operations that
create objects (CREATE and OPEN); in other operations that take
fattr4 arguments, the server MUST reject it with NFS4ERR_INVAL.

The server MUST return NFS4ERR_INVAL if the client attempts to set
both mode and mode_umask in the same operation.

When the server supports the mode_umask attribute, a client creating
a file should use mode_umask in place of mode, with mu_mode set to
the unmodified mode provided by the user and mu_umask set to the
umask of the requesting process.

The server then uses mode_umask as follows:

o On a server that supports ACL attributes, if an object inherits
   any ACEs from its parent directory, mu_mode SHOULD be used and
   mu_umask ignored.

o Otherwise, mu_umask MUST be used to limit the mode: all bits in
   the mode that are set in the unmask MUST be turned off; the mode
   assigned to the new object becomes (mu_mode & ~mu_umask) instead.

6. Security Considerations

The mode_umask attribute shifts to the server the decision about when
to apply the umask. Because the server MUST apply the umask if there
are no inheritable permissions, the traditional semantics are
preserved in the absence of a permission inheritance mechanism. The
only relaxation of permissions comes in the case in which servers
follow the recommendation that they ignore the umask in the presence
of inheritable permissions.

The practice of ignoring the umask when there are inheritable
permissions in the form of a "POSIX" default ACL is of long standing
and has not given rise to security issues. The "POSIX" default ACL
mechanism and the mechanism for permission inheritance in NFSv4 are
equivalent from a security perspective.

7. IANA Considerations

This document does not require any IANA actions.

8. References

8.1. 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>.

   [RFC4506] Eisler, M., Ed., "XDR: External Data Representation
              Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May
              2006, <https://www.rfc-editor.org/info/rfc4506>.

   [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>.

   [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>.

   [RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor
              Version 2 External Data Representation Standard (XDR)
              Description", RFC 7863, DOI 10.17487/RFC7863, November
              2016, <https://www.rfc-editor.org/info/rfc7863>.

   [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>.

   [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>.

8.2. Informative References

   [LinuxACL] Gruenbacher, A., "ACL(5) - Access Control Lists", Linux
              man pages online, ACL(5), March 2002,
              <http://kernel.org/doc/man-pages/online/pages/man5/
              acl.5.html>.

   [SUSv4]   The Open Group, "Single UNIX Specification, Version 4",
              2013.

Acknowledgments

Authors' Addresses

   J. Bruce Fields
   Red Hat, Inc.

   Email: bfields@redhat.com

   Andreas Gruenbacher
   Red Hat, Inc.

   Email: agruenba@redhat.com