US 20020019936A1

(19) United States
(12) Patent Application Publication (10) Pub. No.: US 2002/0019936 A1
Hitz et al. (43) Pub. Date: Feb. 14, 2002

(54) FILE ACCESS CONTROL IN A MULTI-PROTOCOL FILE SERVER

(76) Inventors: **David Hitz**, Los Altos, CA (US); **Andrea Borr**, La Jolla, CA (US); **Robert James Hawley**, San Jose, CA (US); **Mark Muhlestein**, Tucson, AZ (US); **Joan Pearson**, Los Altos, CA (US)

Correspondence Address:
SWERNOFSKY LAW GROUP
P O BOX 390013
MOUNTAIN VIEW, CA 94039-0013 (US)

(21) Appl. No.: **09/927,409**

(22) Filed: **Aug. 10, 2001**

Related U.S. Application Data

(63) Continuation of application No. 09/035,234, filed on Mar. 3, 1998.

(30) **Foreign Application Priority Data**

Mar. 2, 1999 (US) PCT/US99/04550

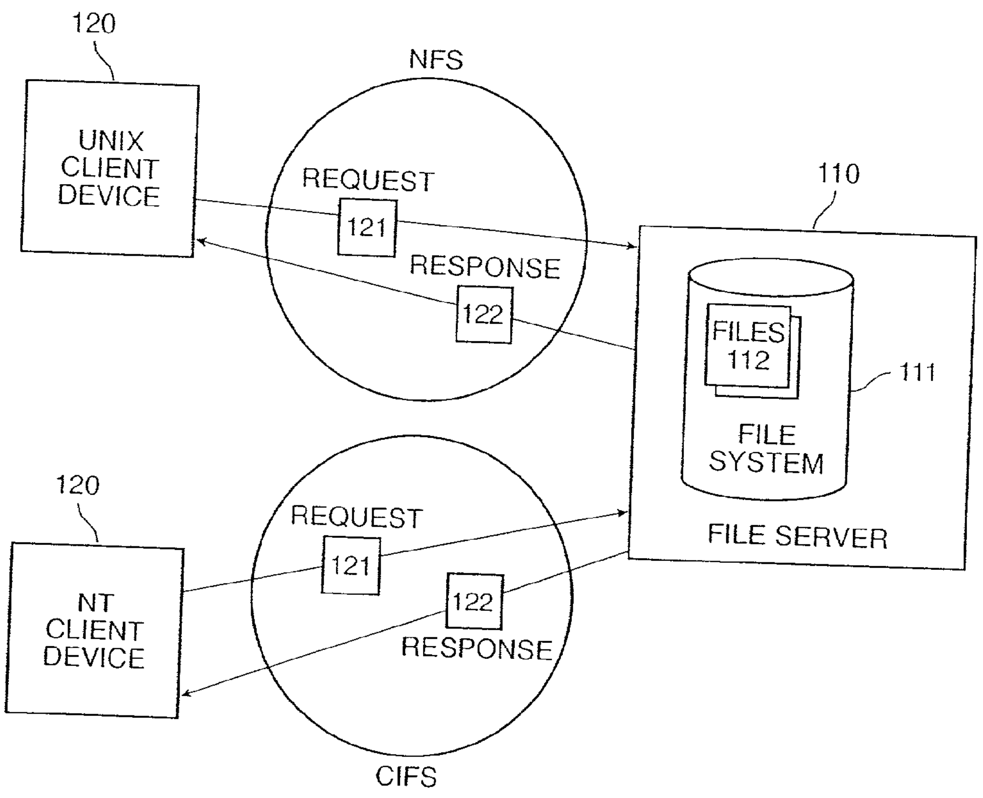
Publication Classification

(51) **Int. Cl.**^{7} **G06F 12/14**

(52) **U.S. Cl.** **713/165**; 713/201

(57) **ABSTRACT**

The invention provides a method and system for enforcing file access control among client devices using multiple diverse access control models and multiple diverse file server protocols. A multi-protocol file server identifies each file with one particular access control model out of a plurality of possible models, and enforces that one particular model for all accesses to that file. When the file server receives a file server request for that file using a different access control model, the file server translates the access control limits for that file into no-less-restrictive limits in the different model. The file server restricts access by the client device using the translated access control limits. Each file is assigned the access control model of the user who created the file or who last set access control limits for the file. When a user having a different access control model sets access control limits, the access control model for the file is changed to the new model. Files are organized in a tree hierarchy, in which each tree is limited to one or more access control models (which can limit the ability of users to set access control limits for files in that tree). Each tree can be limited to NT-model-only format, Unix-model-only format, or mixed NT-or-Unix-models format.



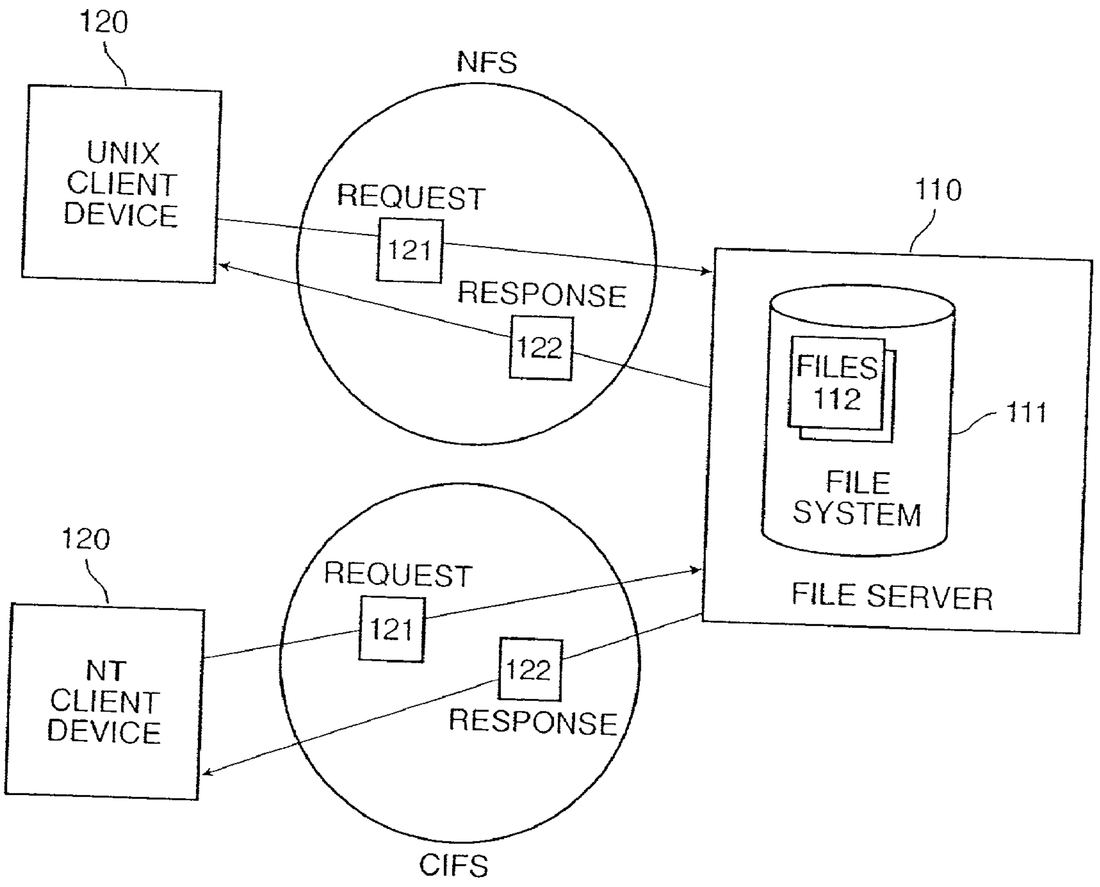

120
UNIX CLIENT DEVICE
NFS
REQUEST
121
RESPONSE
122
110
FILES 112
111
FILE SYSTEM
FILE SERVER
120
NT CLIENT DEVICE
REQUEST
121
122
RESPONSE
CIFS

FIG. 1

FILE ACCESS CONTROL IN A MULTI-PROTOCOL FILE SERVER

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The invention relates to file access control in a multi-protocol file server

[0003] 2. Related Art

[0004] In an integrated computer network, it is desirable for multiple client devices to share access to the same files. One known method is to provide a network file server for storing files, capable of receiving and responding to file server requests from those client devices. These file server requests are made using a file server protocol, which is recognized and adhered to by both the file server and the client device. Because the files are stored at the file server, multiple client devices have the opportunity to share access to the same files.

[0005] In a file system intended for use by more than one user, it is desirable to restrict access by programs to files in the file system. Restricting access includes at least the aspects of (1) user authentication—determining that requesting users are truly who they say they are, and (2) access control validation—determining that an authenticated user is allowed to access a particular file in a particular way. When the file system is maintained on a file server remote from the user making the request, there is an additional aspect of the access control protocol—what requests can be made by the user to access files or to set access control for files.

[0006] One problem in the known art is that there are multiple diverse models for access control validation, each typically associated with a particular file system, and there are multiple diverse access control protocols, each typically corresponding to a model for access control validation. Despite the differences between these models and protocols, the file server should respond to file server requests from each user, and should exhibit access control validation behavior, consistent with each user's model and without security violations or surprises for users.

[0007] For example, a first access control model in common use is associated with the Unix operating system (or a variant thereof). This first access control model associates permissions with each file for a file owner, an owner's group, and all other users. These permissions allow access (for the owner, group, or all other users) to read, write, or execute the indicated file. This first access control model is typically implemented by the NFS ("Network File System") file server protocol, possibly augmented with an adjunct file-locking protocol, NLM ("Network Lock Manager"). A second access control model in common use is associated with the Windows NT operating system. This second access control model associates an ACL (access control list) with each file, each entry in the ACL specifying an individual user, a group of users, or all users. Each entry can allow access (for the specified users) to read, write, or execute the indicated file, or can specifically deny access. This second access control model is typically implemented by the CIFS ("Common Internet File System") protocol. However, NT devices can also use the NFS protocol by means of the "PC NFS" implementation, and Unix devices can also manipulate POSIX ACLs. These two access control models in common use differ in significant ways, including (1) what permissions can be assigned to a file, (2) with what granularity of specificity permissions can be assigned, and (3) how users are identified so as to match them with permissions.

[0008] One method known in the art is to provide a multi-protocol file server that maps all security semantics to that of a single native operating system for the file server, and uses that single native operating system to validate file access control. The "Samba" system and similar emulation packages are believed to use this known method. This known method has the drawback that it can result in security errors or surprises for those client devices using security semantics other than the file server's native operating system.

[0009] Another method known in the art is to provide a multi-protocol file server that supports differing types of security semantics for differing files, but attempts to validate date file access control for each user using the user's access control model. Some "Netware" products available from Novell Corporation are believed to use this known method. This known method has the drawback that the user's access control model can differ significantly from the access control model set for the file, resulting in security errors or surprises for those client devices using security semantics other than associated with the target file.

[0010] Accordingly, it would be desirable to provide a method and system for enforcing file security semantics among client devices using multiple diverse access control models and multiple diverse file server protocols. This advantage is achieved in an embodiment of the invention in which a multi-protocol file server identifies each file with one particular access control model out of a plurality of possible access control models, and enforces that particular access control model for all accesses to that file. When the file server receives a file server request for that file using a file server protocol with a different access control model, the file server translates the access control limits imposed by the file's access control model into no-less-restrictive access control limits in the different access control model. The file server restricts access to the file using the translated access control limits.

SUMMARY OF THE INVENTION

[0011] The invention provides a method and system for enforcing file access control among client devices using multiple diverse access control models and multiple diverse file server protocols. A multi-protocol file server identifies each file with one particular access control model out of a plurality of possible models, and enforces that one particular model for all accesses to that file. When the file server receives a file server request for that file using a different access control model, the file server translates the access control limits for that file into no-less-restrictive limits in the different model. The file server restricts access by the client device using the translated access control limits.

[0012] In a preferred embodiment, each file is assigned the access control model of the user who created the file or who last set access control limits for the file. When a user having a different access control model sets access control limits, the access control model for the file is changed to the new model. Files are organized in a tree hierarchy, in which each tree is limited to one or more access control models (which can limit the ability of users to set access control limits for files in that tree). Each tree can be limited to NT-model-only format, Unix-model-only format, or mixed NT-or-Unix-models format.

BRIEF DESCRIPTION OF THE DRAWINGS

[**0013**] The FIGURE shows a block diagram of a system for enforcing diverse access control models among client devices.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[**0014**] In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. However, those skilled in the art would recognize, after perusal of this application, that embodiments of the invention may be implemented using one or more general purpose processors (or special purpose processors adapted to the particular process steps and data structures) operating under program control, and that implementation of the preferred process steps and data structures described herein using such equipment would not require undue experimentation or further invention.

[**0015**] Inventions described herein can be used in conjunction with inventions described in the following applications:

[**0016**] Application Ser. No. 08/985,398, filed Dec. 5, 1997, in the name of Andrea Borr, titled "Multi-Protocol Unified File-locking", attorney docket number NET-023.

[**0017**] This application is hereby incorporated by reference as if fully set forth herein.

[**0018**] The enclosed technical appendix is part of the disclosure of the invention, and is hereby incorporated by reference as if fully set forth herein.

[**0019**] System Elements

[**0020**] The FIGURE shows a block diagram of a system for enforcing diverse access control models among client devices.

[**0021**] A system **100** includes a file server **110**, and a set of client devices **120**.

[**0022**] The file server **110** maintains a file system **111**, including a set of files **112**.

[**0023**] In a preferred embodiment, the file server **110** maintains the file system **111** using inventions described in the following patent applications:

[**0024**] Application Ser. No. 08/471,218, filed Jun. 5, 1995, in the name of inventors David Hitz et al., titled "A Method for Providing Parity in a Raid Sub-System Using Non-Volatile Memory", attorney docket number NET-004;

[**0025**] Application Ser. No. 08/454,921, filed May 31, 1995, in the name of inventors David Hitz et al., titled "Write Anywhere File-System Layout", attorney docket number NET-005;

[**0026**] Application Ser. No. 08/464,591, filed May 31, 1995, in the name of inventors David Hitz et al., titled "Method for Allocating Files in a File System Integrated with a Raid Disk Sub-System", attorney docket number NET-006.

[**0027**] Each of these applications is hereby incorporated by reference as if fully set forth herein.

[**0028**] The file server **110** is disposed for receiving file server requests **121** from the client devices **120**. The file server **110** parses each request **121**, determines whether the operation requested in the request **121** is allowed (for the client device **120** that sent the request **121** and for the one or more target files **112** specified by the request **121**). If allowed, the file server **110** performs that operation on the one or more target files **112**.

[**0029**] The file server **110** is also disposed for transmitting file server responses **122** to the client devices **120**. The file server **110** determines the response to each request **121** (possibly including a response indicating that the requested operation was not allowed), generates that response **122**, and transmits that response **122** to the client device **120** that sent the request **121**.

[**0030**] Each client device **120** is disposed for transmitting file server requests **121** to the file server **110**, and for receiving file server responses **122** from the file server **110**.

[**0031**] Access Control Models

[**0032**] In a preferred embodiment, each client device **120** can be either a Unix client device **120** or a Windows NT client device **120**. Each client device **120** can either use the NFS file server protocol to make requests **121**, or use the CIFS file server protocol to make requests **121**. (Although typically Unix client devices **120** use the NFS file server protocol and NT client devices **120** use the CIFS file server protocol, it is possible for NT client devices **120** to use the NFS file server protocol by using the PC-NFS implementation of that file server protocol.) The file server **110** receives each request **121** and (if allowed) performs the requested operation on the target files **112** specified by the request **121**.

[**0033**] The file server **10** supports more than one access control model, including a "Unix Perms" access control model (herein "Unix security style") and an "NT ACL" access control model (herein "NT security style").

[**0034**] Unix security style uses user IDs (UIDs) to identify users, and group IDs (GIDs) to identify groups to which those users belong. Unix security style associates the following access control limits with each file:

[**0035**] a UID for the owner;

[**0036**] a GID for the owner;

[**0037**] a set of "user" permissions—allowing permission to read, write, or execute the file by the owning user;

[**0038**] a set of "group" permissions—allowing permission to read, write, or execute the file by the owning user's group; and

[**0039**] a set of "other" permissions—allowing permission to read, write, or execute the file by all other users.

[**0040**] Unix security style is supported by the NFS ("Network File System") file server protocol, possibly augmented with the NLM ("Network Lock Manager") adjunct file-locking protocol.

[**0041**] NFS is a stateless protocol, so each NFS file server request **121** includes the UIDs and GIDs of the user making the request. The Unix client device **120** determines the UIDs and GIDs for the user at a login time for the user, by reference to the system password file (/etc/passwd) and group file (/etc/groups).

[**0042**] To enforce file access control using Unix security style, the file server **110** determines if the request **121** is from the owning user, from a user in the owning user's group, or from some other user. Responsive to this determination, the file server **110** uses one of the user permissions, the group permissions, or the other permissions, to determine whether to allow the request **121**.

[**0043**] NT security style uses security IDs (SIDs) to identify both users and groups. NT security style associates the following access control limits with each file:

[**0044**] an SID for the owner;

[**0045**] an SID for the owner's group;

[**0046**] an ACL (access control list).

[**0047**] The NT ACL includes one or more ACEs (access control entries), each of which includes an SID indicating the user or group to which it applies, and a set of permissions. NT security style provides for the three Unix permissions (read, write, or execute), as well as "CHANGE PERMISSIONS" permission, "TAKE OWNERSHIP" permission, "DELETE" permission, "DELETE CHILD" permission, and other permissions.

[**0048**] NT security style is supported by the CIFS ("Common Internet File System") protocol. NT security style is further described in the following articles: R. Reichel, "Inside Windows NT Security", Windows/DOS Developers' Journal (April & May 1993), and in Stephen Sutton, "Windows NT Security Guide" (ISBN 0201419696).

[**0049**] CIFS is a session-based protocol, so the NT client device **120** transmits the NT user name and password to the file server **110** at a session connect time, from which the SIDs for the user and the user's groups are determined. The file server **110** can attempt to authenticate the user itself, or (preferably) forward the NT user name and password to an NT primary domain controller.

[**0050**] To enforce file access control using NT security style, the file server **110** determines the SID for the user and the user's group, for the user making the request **121**. The file server **110** accumulates permissions granted to that user from ACEs that apply, then subtracts permissions specifically denied to that user. Responsive to this accumulation and subtraction, the file server **110** determines whether to allow the request **121**.

[**0051**] Although a preferred embodiment of the invention is described with regard to Unix security style and NT security style, the invention can readily be used with other access control models, such as the "POSIX ACL" access control model supported by some Unix devices, and by some other operating systems. The concepts and features of the invention described herein can readily be used in a file server **110** supporting the "POSIX ACL" access control model in addition to or instead of the access control models described in detail herein, without further invention or undue experimentation. Accordingly, the scope and spirit of the invention includes such file servers and methods for their use.

[**0052**] The file server **110** designates each file **112** maintained in its file system **111** as having one specific access control model out of the plurality of access control models it supports. In a preferred embodiment, each file **112** is designated to use either Unix security style or NT security style. The file server **110** enforces the designated security style for each file **112** for all attempts to access that file **112**. Thus, the file server **110** enforces the designated security style for all requests **121** made for that target file **112**, whether those requests **121** come from Unix devices or NT devices, and whether those requests **121** use the NFS file server protocol or the CIFS file server protocol.

[**0053**] Access Control Enforcement

[**0054**] If the file server **110** receives a request **121** for a target file **112**, and the request **121** matches the security style target file **112**, the file server **110** validates the request **121** against access control limits for that file **112** imposed by that security style.

[**0055**] The file server **110** thus recognizes and enforces at least the following circumstances:

[**0056**] NT security style. The file **112** has NT security style and has a corresponding set of access control limits (an NT ACL), which has been set by a client device **120** using the CIFS file server protocol.

[**0057**] If a client device **120** makes a request **121** to access the file **112** using the CIFS file server protocol, the file server **110**, if it is able to, enforces the NT ACL using NT security style.

[**0058**] If the file server **110** is able to determine the NT user, either by communication with an NT domain controller, or by reference to an NT user SID (security ID) database, the file server **110** is able to enforce the NT ACL using NT security style.

[**0059**] If the file server **110** is not able to determine the NT user, it determines the equivalent Unix user, using a UID for a Unix user recorded for the CIFS file server protocol session, and enforces the NT ACL as if the request **121** came from that Unix user.

[**0060**] Unix security style. The file **112** has Unix security style and has a corresponding set of access control limits (Unix Perms), which have been set by a client device **120** using the NFS file server protocol

[**0061**] If a client device **120** makes a request **121** to access the file **112** using the NFS file server protocol, the file server **110** enforces the Unix Perms using Unix security style.

[**0062**] However, the file server **110** can also receive a request **121** that does not match the security style for the target file **112**. The file server **110** can enforce the security style for the target file **112** against a non-matching client device **120** by validating either (1) a translated user ID for the client device **120** or (2) a translated set of access control limits for that file **112**. As described herein, the file server **110** validates translated user IDs for all Unix security style files **112**, and preferably validates translated user IDs for NT security style files **112** (when possible). Moreover, when the file server **110** validates translated access control limits for the file **112**, the translated access control limits are not recomputed for each request **121**, but are cached with the file **112** for reuse.

[**0063**] The file server **110** thus also recognizes and enforces at least the following circumstances:

[**0064**] NT security style. The file **112** has NT security style and has a corresponding set of access control limits (an NT ACL), which has been set by a client device **120** using the CIFS file server protocol.

[**0065**] If a client device **120** makes a request **121** to access the file **112** using the NFS file server protocol, the file server **110** determines the Unix user, associated with the client device **120**, that is making the request **121**. The Unix user has a UID (user ID).

[**0066**] In a preferred embodiment, the file server **110** maps the Unix user into an equivalent NT user. The file server **110** translates the UID into an SID (security ID) that is an equivalent user to the Unix user. The file server **110** enforces the access control limits (the NT ACL) for the equivalent NT user (the SID).

[**0067**] The file server **110** performs the following process to map the Unix user into an equivalent NT user:

[**0068**] The client device **120** contacts the file server **110** using the NFS file server protocol. The NFS file server protocol request **121** includes a UID for the Unix user associated with the client device **120**.

[**0069**] The file server **110** looks up the UID in the Unix password file (/etc/passwd), and thus identifies the Unix user name for the Unix user. The Unix user name is an alphanumeric string identifying the Unix user.

[**0070**] The file server **110** translates the Unix user name into an NT user name using a selected mapping file. Similar to the Unix user name, the NT user name is an alphanumeric string identifying the NT user. If there is no such translation for the Unix user name, the file server **110** uses the Unix user name, without translation, as the NT user name.

[**0071**] In a preferred embodiment, the mapping file includes a set of records each identifying an NT user by NT user name, and associating an equivalent Unix user name with the NT user name.

[**0072**] The file server **110** contacts an NT domain controller to determine an SID for the NT user name. If there is no such NT user, the file server **110** uses a selected parameter for unmapped Unix users. In a preferred embodiment, this selected parameter is set to the NT user "guest" by default.

[**0073**] The file server **110** contacts the NT domain controller to obtain the SIDs of all groups for which the NT user is a member.

[**0074**] The file server **110** caches UID-to-SID mappings for a period of time, preferably about a few hours.

[**0075**] In an alternative preferred embodiment, or if the file server **110** is unable to map Unix users to NT users (for example, if domain authentication has been turned off), the file server **110** maps the NT ACL into a no less restrictive set of Unix Perms. The file server **110** determines these Unix Perms in response to the NT ACL and in response to the Unix user. The file server **110** enforces the mapped access control limits (the Unix Perms) for the actual Unix user (the UID).

[**0076**] The file server **110** might perform dynamic permission mapping, in which the file server **110** maps the NT ACL into a set of Unix Perms at the time the mapping is required. In a preferred embodiment, the file server **110** caches the translated Unix Perms with the file **112**. Accordingly, for validating access control limits, the file server **110** performs static permission mapping, in which the file server **110** maps the NT ACL into a set of Unix Perms at the time the NT ACL is set.

[**0077**] The file server **110** performs the following process to achieve static permission mapping:

[**0078**] The file server **110** determines the NT user that is the owner of the file **112**, and maps the NT user into an equivalent Unix user (the file server **110** maps the SID for the NT user into a UID for a Unix user).

[**0079**] The file server **110** examines the NT ACL for the file **112** and determines whether there are any "deny access" provisions.

[**0080**] If the NT ACL for the file **112** has no "deny access" provisions, the file server **110** generates a set of Unix Perms having an entry for "user permissions," consistent with the file access control limits provided by the NT ACL. The file server **110** sets the Unix Perms for "group permissions" equal to the Unix Perms for "other permissions." The file server **110** sets the Unix Perms for "other permissions" equal to the NT ACL entry for "everyone," if one exists.

[**0081**] If the NT ACL for the file **112** does have "deny access" provisions, the file server **110** rejects the request **121**.

[**0082**] Because static permission mapping is not responsive to the particular user making the request **121**, the file server **110** does not attempt to determine what the provisions of the NT ACL are for that particular user.

[**0083**] Unix security style. The file **112** has Unix security style and has a corresponding set of access control limits (Unix Perms), which have been set by a client device **120** using the NFS file server protocol

[**0084**] If a client device **120** makes a request **121** to access the file **112** using the CIFS file server protocol, the file server **110** determines the NT user, associated with the client device **120**, that is making the request **121**. The NT user has an SID (session ID).

[**0085**] The file server **110** maps the NT user into an equivalent Unix user. The file server **110** translates the SID into a UID that is an equivalent user to the NT user. The file server **110** enforces the access control limits (the Unix Perms) for the equivalent Unix user (the UID).

[**0086**] The file server **110** performs the following process to map the NT user into an equivalent Unix user:

[0087] The client device **120** starts a CIFS session (the client device **120** first contacts the file server **110** using the CIFS file server protocol). The client device **120** transmits its NT user name to the file server **110**.

[0088] The file server **110** translates the NT user name into a Unix user name using a mapping file. If there is no such translation for the NT user name, the file server **110** uses the NT user name, without translation, as the Unix user name.

[0089] The file server **110** looks up the Unix user name in the Unix password file (/etc/passwd), and thus identifies the Unix user, the UID for the Unix user, the Unix user's primary group, and the primary GID (group ID) for the Unix user. If there is no such Unix user name in the Unix password file, the file server **110** uses a selected parameter for unmapped NT users. In a preferred embodiment, this selected parameter is set to the Unix user "nobody" by default.

[0090] The file server **110** looks up the Unix user name in the Unix group file (/etc/groups) to determine any other groups and any other GIDs associated with the Unix user.

[0091] Reading and Modifying Access Control Limits

[0092] Each file **112** has its security style set by the file server **110** so that either (a) a request **121** to perform an operation on the file **112**, or (b) a request **121** to perform an operation that sets the access control limits for the file **112**, produce expected results.

[0093] When the file **112** is first created, the file server **110** sets the security style for the file **112** equal to a security style associated with the file server protocol used to create it. (This is limited by restrictions imposed by access control trees, described herein.) Thus, if the file **112** is created using the NFS file server protocol, the security style for the file **112** is set to Unix security style. Similarly, if the file **112** is created using the CIFS file server protocol, the security style for the file **112** is set to NT security style.

[0094] When the file **112** has its access control limits modified, the file server **110** sets the security style for the file **112** equal to a security style associated with the new access control limits. (This is limited by restrictions imposed by access control trees, described herein.) Thus, if a client device **120** sets a set of Unix Perms for the file **112**, the security style for the file **112** is set to Unix security style. Similarly, if a client device **120** sets an NT ACL for the file **112**, the security style for the file **112** is set to NT security style.

[0095] The file server **110** can receive a request **121** to read or view the access control limits for a file **112**. Also, when the file server **110** receives a request **121** to make an incremental change to the access control limits for a file **112**, it determines the current access control limits for the file **112** before making the incremental change.

[0096] The file server **110** thus recognizes and enforces at least the following circumstances:

[0097] NT security style. The file **112** has NT security style and has a corresponding set of access control limits (an NT ACL), which has been set by a client device **120** using the CIFS file server protocol.

[0098] If a client device **120** makes a request **121** to read or modify the access control limits for the file **112** using the NFS file server protocol, the file server **110** determines the Unix user, associated with the client device **120**, that is making the request **121**.

[0099] The file server **110** performs the same process to map an NT ACL into a set of Unix Perms as described above for validation of file access control, with the following exceptions:

[0100] Unlike validation of access control limits, the file server **110** treats translation of access control limits differently for requests **121** to read or modify the access control limits for the file **112**.

[0101] Preferably, the file server **110** performs dynamic permission mapping, in which the file server **110** maps the NT ACL into a set of Unix Perms at the time the mapping is required. NT security style is richer than Unix security style-for example, Unix security style has no "deny access" provisions. Thus, it is possible for the file server **110** to map the NT ACL into a set of Unix Perms that appears different for different Unix users. For example, if the NT ACL, for a file **112** whose owner is Charles, specifically provides read access to Allen but specifically denies read access to Beth, the file server **110** will provide different Unix perms to each of the users Allen and Beth. One set will allow read access by Allen's group and one set will disallow read access by Beth's group, in harmony with the access provided by the actual NT ACL.

[0102] The file server **110** performs the following process to achieve dynamic permission mapping:

[0103] The file server **110** determines the NT user that is the owner of the file **112**, and maps the NT user into an equivalent Unix user (the file server **110** maps the SID for the NT user into a UID for a Unix user).

[0104] The file server **110** examines the NT ACL for the file **112** and determines whether there are any "deny access" provisions.

[0105] If the NT ACL for the file **112** has no "deny access" provisions, the file server **110** generates a set of Unix Perms having entries for "user permissions" and "other permissions," consistent with the file access control limits provided by the NT ACL. The file server **110** sets the Unix Perms for "group permissions" equal to the Unix Perms for "other permissions."

[0106] If the NT ACL for the file **112** does have "deny access" provisions, the file server **110** attempts to determine if any apply to the particular Unix user. If the file server can tell, it generates a set of Unix Perms that reflect the access control limits its currently available for this particular file **112** and this particular Unix user. If the file server **110** cannot tell, it rejects the request **121**. (Alternatively, the file server **110** could reject the request **121** if there are any "deny access" provisions in the NT ACL.)

[0107] In alternative embodiments, the file server **110** may perform static permission checking, similar to validation of access control limits, for requests **121** to read or modify the access control limits for the file **112**.

[**0108**] If the request **121** attempts to modify attributes of the file **112** that have no effect on access control limits for the file **112** (such as access time or modify time), the file server **110** makes those modifications without change to the access control limits for the file **112**.

[**0109**] If the request **121** attempts to modify some but not all access control limits for the file **112**, the file server **110** generates a set of Unix Perms in response to the NT ACL for the file **112**, as described above. The file server **110** modifies the generated Unix Perms as specified by the request **121**. If the file server **110** cannot generate a set of Unix Perms in response to the NT ACL for the file **112**, the file server **110** rejects the request **121**.

[**0110**] One difference in setting access control limits is that, according to NT security style, files **112** can be specifically set to be "READ-ONLY." According to Unix security style, files are set to be read only by clearing the WRITE permission for the owner of the file **112**. When a client device **120** using the CIFS file server protocol attempts to set the READ-ONLY attribute of a file **112** with Unix security style, the file server **110** clears the WRITE permission for the owner of the file **112** in the Unix Perms for that file **112**.

> [**0111**] Unix security style. The file **112** has Unix security style and has a corresponding set of access control limits (Unix Perms), which have been set by a client device **120** using the NFS file server protocol

[**0112**] The file server **110** performs the following process to map a set of Unix Perms into an NT ACL for display or modification of those Unix Perms by a CIFS client device **120**:

> [**0113**] The file server **110** generates an NT ACL entry for "owner," providing the same access control limits as the Unix Perms entry for "user permissions."
>
> [**0114**] The file server **110** generates an NT ACL entry for "everyone," providing the same access control limits as the Unix Perms entry for "other permissions."
>
> [**0115**] If possible, the file server **110** generates an NT ACL entry for the actual requesting user, providing the same access control limits as the Unix Perms entry for that user. This step could require mapping the Unix user into an equivalent NT user using the UID-to-SID cache.

[**0116**] Similar to modification of an NT ACL entry by a Unix user, if the request **121** (for modification of Unix Perms by an NT user) attempts to modify attributes of the file **112** that have no effect on access control limits for the file **112**, the file server **110** makes those modifications without change to the access control limits for the file **112**.

[**0117**] If the request **121** attempts to modify some but not all access control limits for the file **112**, the file server **110** generates an NT ACL in response to the set of Unix Perms for the file **112**, as described above. The file server **110** modifies the generated NT ACL as specified by the request **121**.

[**0118**] Access Control Subtrees

[**0119**] In a preferred embodiment, the files **112** in the file system **111** are organized into a tree, having a set of branch nodes and a set of leaf nodes. One branch node of the tree is a root node, and each branch node of the tree is a root node for a subtree of the tree. In the file system **111**, each branch node is a directory, and each leaf node is a file **112**. A directory is a type of file **112** that includes information about those branch nodes and leaf nodes in a subtree for which it is the root node.

[**0120**] The file server **110** associates a limited set of access control models with each subtree. In a preferred embodiment in which the file server **110** supports Unix security style and NT security style, the file server **110** designates each subtree as being NT-only format, Unix-only format, or mixed format.

[**0121**] NT-only Format

[**0122**] When the file server **110** designates a subtree as being NT-only format, it restricts creation of files **112** within that subtree to files **112** having NT security style. The file server **110** also prohibits changing the access control model of files **112** within that subtree to other than NT security style.

[**0123**] According to NT security style, new files **112** inherit NT ACL settings from their parent nodes. If a client device **120** using the NFS file server protocol attempts to create a file **112** in a subtree having NT-only format, that file **112** can only be created by the Unix user equivalent to the NT user who is the NT-owner of the root node of the subtree. The file server **110** determines if the Unix user making the request **121** is the equivalent by (a) mapping the SID for the NT user who is the owner into an equivalent UID; (b) storing that UID in its record for the file **112**; and (c) comparing that UID with the UID in the request **121**.

[**0124**] According to NT security style, there is a particular "DELETE" permission and a particular "DELETE-CHILD" permission. If the file server **110** is unable to determine if a Unix user has these permissions, it rejects requests **121** to delete files **112** in NT-only format subtrees, unless the request **121** is from the owner of the file **112** (the equivalent Unix user of the NT user who is the owner) or the Unix user "root".

[**0125**] According to NT security style, there is a particular "CHANGE-PERMISSION"permission and a particular "TAKE-OWNER" permission. If the file server **110** is unable to determine if a Unix user has these permissions, it denies requests **121** to set any permissions for files **112** in a NT-only format subtree, unless the request **121** is from the owner of the file **112** (the equivalent Unix user of the NT user who is the owner) or the Unix user "root".

[**0126**] Unix-only Format

[**0127**] Similarly, when the file server **110** designates a subtree as being Unix-only format, it restricts creation of files **111** within that subtree to files **111** having Unix security style. The file server **110** also prohibits changing the access control model of files **111** within that subtree to other than Unix security style. Attempts to set an NT ACL would change the access control model for that file **112** to NT security style, and so are rejected in a Unix-only format subtree.

[**0128**] When a client device **120** using the CIFS file server protocol creates a file **112** in a Unix-only format subtree, the file server **110** sets the owner of the file **112** to the Unix user equivalent to the NT user making the request **121**. The file server **110** maps the SID for the NT user to a UID for an equivalent Unix user, and uses that UID to set the owner of the file **112**.

[0129] According to Unix security style, there is no "CHANGE-PERMISSION"permission or "TAKE-OWNER" permission. The file server **110** always denies requests **121** to set these permissions for files **112** in a Unix-only format subtree.

[0130] Mixed Format

[0131] When the file server **110** designates a subtree as being mixed format, it allows creation of files **111** with either Unix security style or NT security style. The file server **110** does not prohibit changing the access control model of files **111** within that subtree to either Unix security style or NT security style.

[0132] An administrator for the file server **110** can change the designation of a subtree from a first format to a second format (for example, from mixed format to either NT-only format or Unix-only format). When the second format is possibly incompatible with the first format (for example, a subtree changed to NT-only format includes nodes that are Unix security style), the file server **110** converts those files **112** with incompatible access control models as it sets permissions for those files **112**. Requests **121** for a file **112** which only check permissions are still validated using the access control model in place for the file **112**.

[0133] Although the invention is described herein with regard to only two access control models, the invention can readily be used with three or more access control models. In such alternative embodiments, there are a larger number of possible subtree formats, including subtree formats that restrict the files **112** within that subtree to one of a set of multiple access control models, less than the set of all access control models recognized by the file server **110**.

[0134] In a preferred embodiment, the root node of the file system **111** is designated as having mixed format. Client devices **120** that are owners of a subtree can modify the format of a subtree by request **121** to the file server **110**; thus, client devices **120** can modify subtrees to have NT-only format, Unix-only format, mixed format. When a new subtree is created, the file server **110** designates the new subtree as having the same format as its parent; thus, mixed format if it is created within a subtree that is already mixed format (the default), NT-only format if it is created within a subtree that is already NT-only format, and Unix-only format if it is created within a subtree that is already Unix-only format.

[0135] Alternative Embodiments

[0136] Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

1. A method of operating a file server, said method including steps for

identifying a first file on said file server with a first security style selected from among a plurality of security styles; and

enforcing said first security style for all accesses to said first file.

2. A method as in claim 1, wherein said plurality of security styles includes a Windows NT security style.

3. A method as in claim 1, wherein said plurality of security styles includes a Unix security style.

4. A method as in claim 1, including steps for

associating said first file with a subset of files in a file system; and

limiting said subset of files to a security subset of said plurality of security styles;

wherein attempts to set permissions in said file system tree are restricted to said security subset.

5. A method as in claim 4, wherein said security subset includes a Windows NT security style.

6. A method as in claim 4, wherein said security subset includes a Unix security style.

7. A method as in claim 1, including steps for identifying said first file with a second security style in response to a file server request.

8. A method as in claim 7, including steps for associating said second security style with a file server request for setting permissions for said first file when said file server request is successful.

9. A method as in claim 7, wherein said steps for identifying include steps for translating a first set of permissions associated with said first file in said first security style to a second set of permissions in said second security style, wherein said second set of permissions is no less restrictive than said first set of permissions.

10. A method as in claim 1, wherein said steps for enforcing include steps for

recognizing a first set of permissions associated with said first file in said first security style;

defining a first user type associated with said first security style;

translating a user from a second user type associated with a second security style into said first user type; and

enforcing a file server request from said second user type using said first user type and said first set of permissions.

11. A method as in claim 10, wherein said steps for translating are performed with regard to access control limits applicable to said first file at a time of said steps for enforcing.

12. A method as in claim 10, wherein said steps for translating are performed with regard to access control limits applicable to said first file at a time said access control limits are set.

13. A method as in claim 1, wherein said steps for enforcing include steps for

translating a first set of permissions associated with said first file in said first security style to a second set of permissions in a second security style, wherein said second set of permissions is no less restrictive than said first set of permissions; and

enforcing a file server request in said second security style using said second set of permissions.

14. A method as in claim 13, wherein said steps for translating are performed with regard to access control limits applicable to said first file at a time of said steps for enforcing.

15. A method as in claim 13, wherein said steps for translating are performed with regard to access control limits applicable to said first file at a time said access control limits are set.

16. A file server including

a set of files available said file server, each said file having an associated security style selected from among a plurality of security styles available on said file server;

wherein said file server enforces said associated security style for all accesses to said file.

17. A file server as in claim 16, wherein said plurality of security styles includes a Windows NT security style.

18. A file server as in claim 16, wherein said plurality of security styles includes a Unix security style.

19. A file server as in claim 16, including

a subtree of files in said file system associated with a security subset of said plurality of security styles;

wherein said file server restricts attempts to set permissions in said subtree to said security subset.

20. A file server as in claim 19, wherein said security subset includes a Windows NT security style.

21. A file server as in claim 19, wherein said security subset includes a Unix security style.

22. A file server as in claim 16, wherein said file server is capable of altering the security style associated with said file in response to a file server request.

23. A file server as in claim 22, wherein said file server is capable of altering the security style associated with said file in response to a file server request when said file server request is successful.

24. A file server as in claim 22, wherein said file server is capable of translating a first set of permissions associated with said file in a first security style to a second set of permissions in a second security style, wherein said second set of permissions is no less restrictive than said first set of permissions.

25. In a file server having a plurality of files, a data structure associating a security style with each said file, said security style being selected from among a plurality of security styles available on said file server.

26. A data structure as in claim 25, wherein said plurality of security styles includes a Windows NT security style.

27. A data structure as in claim 25, wherein said plurality of security styles includes a Unix security style.

28. In a file server having a plurality of files and a security style associated with each said file, said security style being selected from among a plurality of security styles available on said file server, a data structure associating a security subset of said plurality of security styles with a subtree of said files available on said file server.

29. A data structure as in claim 28, wherein said security subset includes a Windows NT security style.

30. A data structure as in claim 28, wherein said security subset includes a Unix security style.

* * * * *