The following paper was originally published in the
Proceedings of the Large Installation System Administration of Windows NT Conference
Seattle, Washington, August 5–8, 1998

File System Security: Secure Network Data Sharing for NT and UNIX

Bridget Allison, Rob Hawley, Andrea Borr, Mark Muhlestein, and Dave Hitz
Network Appliance

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

File System Security: Secure Network Data Sharing for NT and UNIX

Bridget Allison - bridget@netapp.com, Robert Hawley, hawleyr@netapp.com,
Andrea Borr - aborr@netapp.com, Mark Muhlestein - mmm@netapp.com
David Hitz - hitz@netapp.com
Network Appliance Inc.

Abstract

Sharing network data between UNIX and NT systems is becoming increasingly important as NT moves into areas previously serviced entirely by UNIX. One difficulty in sharing data between UNIX and NT is that their file system security models are quite different. NT file servers use access control lists (ACLs) that allow permissions to be specified for an arbitrary number of users and groups, while UNIX NFS servers use traditional UNIX permissions that provide control only for owner, group, and other. This paper describes a merged model in which a single file system can contain both files with NT-style ACLs and files with UNIX-style permissions. For native file service requests (NFS requests to UNIX-style files and NT requests to NT-style files) the security model exactly matches a UNIX or NT fileserver. For non-native requests, heuristics allow a reasonable level of access without compromising the security guarantees of the native model.

Introduction

For the past 18 months Network Appliance has been shipping its Multiprotocol filer - a dedicated file server appliance which can be licensed and configured to speak some combination of NFS, CIFS and HTTP. One of the primary challenges facing any solution which offers a single location for data to clients that speak disparate protocols is the maintenance of data integrity. If the file server cannot ensure the integrity of the data then it has failed in its purpose. When a client trusts a file server with its data that leap of faith must be honored. If it is not, then the file server will not be used for data storage. Clearly a dedicated file server appliance must address data integrity head-on if it is to fulfill its mission in life. If we set aside the hardware issues related to data integrity and focus only on the software issues then data integrity is comprised of two parts: secure file/byte range locking and file and directory level security.

This paper addresses Network Appliance's approach to the second part of the data integrity story: file level security. Specifically, it describes how Network Appliance has merged UNIX and Windows NT (NTFS) file level security to provide a flexible yet secure merged security model which allows Windows and NFS users to use the most appropriate security model for the data being stored. This model allows all data to be accessible by all protocols, UNIX perms and NT ACLs accommodated within the same file system while giving the administrator control over which file level security model will be used in each area of the file server.

This paper also describes how the merged security model is being used by a Network Appliance customer: Glaxo Wellcome R&D.

Finally, this paper details areas to be considered when migrating to a merged system and mentions other tools that may be useful in a mixed UNIX and Windows environment. It also talks briefly about how other multiprotocol solutions address merged security issues; specifically AT & T's Advanced Server for Unix and Samba.

A word about qtrees

Before going into the design of the integrated security model it is worth taking a moment to explain how the NetApp filer

uses qtrees since they are the basis for security style configuration. On a NetApp filer qtrees allow a flexible segmentation of a filesystem for quota management, opportunistic lock (oplock) configuration, backup management and also security styles. Since a qtree can be dynamically grown, deleted or created this provides a lot of flexibility for the administrator. Qtree security styles can be changed on the fly, conversion of a UNIX style qtree into an NTFS style qtree for example, can be done quickly and easily as your storage needs change.

For a simpler configuration, an entire volume can itself be a qtree. So, for example, setting the security style on the root of a single volume filer with the command

qtree security / ntfs

causes the volume to report a filesystem type of 'NTFS' to Windows clients and allows NT ACLs to be created across the entire filesystem. This is the equivalent of formatting the C: drive of your Windows NT file server as NTFS from a file security perspective.

1. Merged Security Model Design Goals

The Network Appliance merged security model was designed to meet several goals:

(1) Make Windows Users Happy

Support an NT-centric security model based on NT Access Control Lists (ACLs). To a Win95/98 or NT client using CIFS (the standard Windows remote file service protocol) this security model should behave exactly like a Windows NT file server.

(2) Make UNIX Users Happy.

Support a UNIX-centric security model based on UNIX permissions. To an NFS client, this security model should behave exactly like a UNIX NFS server.

(3) Let Them Work Together.

Provide reasonable heuristics so that Windows users can access files with UNIX-centric security, and UNIX users can access NT-centric files.

To meet these goals, NetApp supports both NTFS-trees and UNIX-trees. Administrators can set the security model on the root of a WAFL file system, but they can also set the security for individual qtrees, allowing Windows and UNIX users both to be happy in the same file system.

Native requests -- NFS to a UNIX-tree or CIFS to an NTFS-tree -- work exactly as expected. UNIX-trees are modeled after Solaris, and NTFS-trees are modeled after Windows NT. Non-native requests use heuristics designed to operate as intuitively as possible while still maintaining security.

CIFS requests to UNIX-trees are handled by mapping each CIFS user to an equivalent UNIX user, and then validating against the standard UNIX permissions.

In the simplest case, a user named John might have the account "john" on both NT and UNIX. When John starts a CIFS session, the filer looks up "john" in /etc/passwd (or over NIS), and uses the specified UID and GID for all access validation.

A user name mapping file handles the case where John's NT account is "john", but his UNIX account is "jsmith". NT users with no UNIX account may be mapped to a specified UNIX account, or they may simply be denied access.

NFS requests to NTFS-trees in phase 1 of the implementation are validated using special UNIX permissions that are set whenever an ACL is updated. The UNIX permissions are guaranteed to be more restrictive than the ACL, which means that users can never circumvent ACL-based security by coming in through NFS. On the other hand, since UNIX permissions are less rich than NT ACLs, a multiprotocol user may be unable to access some files over NFS even though they are accessible over CIFS. In practice, NFS access to NTFS-trees works well for the owner, and for access granted to the NT "everyone" account, but not for other cases.

In a future addition of the integrated security model we will implement UID to SID mapping which will allow us to map each NFS user to an equivalent CIFS user, and then validate against the file's ACL. At that point it may not be useful to maintain a set of UNIX perms on a file which has an ACL.

In addition to UNIX-trees and NTFS-trees, the filer supports a Mixed-tree in which the security model is determined on a file-by-file basis. Files created by UNIX users use UNIX permissions, and files created by NT users use NT ACLs. A file's security model may be flipped from one style to another by NFS "setattribute" or CIFS "set ACL" requests. Only a file's owner can flip the style.

Several features allow special control over privileged users. The filer supports the NT "Administrators" local group, which lists the NT accounts that have administrator privileges. The NT User Manager interface can be used to manage the "Administrators" local group over the network. The user name mapping file may be used to map the NT "Administrator" account into UNIX "root", or to a non-privileged UNIX account such as "nobody". Privileges for UNIX "root" are controlled using the /etc/exports "root=" flag. Requests from "root" are mapped to "nobody" unless the "root=" flag on an export explicitly allows root privileges.

2. Understanding File System Security Models

2.1 Native File System Security Models for NT and UNIX

This section provides a quick overview of UNIX and NT file system security, since many people are familiar with one or the other, but not both.

2.1.1 UNIX Security Model

UNIX uses user IDs (UIDs) to identify users, and group IDs (GIDs) to identify groups. The UNIX permissions stored with each file consist of:

- UID of the owner

- GID of the owner

- User perm bits (defining read, write, execute for the owner)

- Group perm bits (defining rwx for the group)

- Other perm bits (defining rwx for anyone else)

When performing validation, UNIX first determines whether the request is from the file's owner, someone in the file's group, or anyone else, and then uses the user perms, group perms, or other perms, respectively.

2.1.2 NT Security Model

NT uses security IDs (SIDs) to identify both users and groups. The NT permissions for each file consist of:

- SID of the owner

- SID of the owner’s primary group

- ACL (Access Control List) for the file

The ACL contains one or more access control entries (ACEs). Each ACE contains a SID, indicating the user or group to which the ACE applies, and a set of permission bits. NT permission bits include the three UNIX bits -- read, write, and execute -- as well as "change permissions" (P), "take ownership" (O), "delete" (D), and others. An ACE can either allow the specified permissions, or deny them. When performing validation, NT walks the list of permissions granted to the user by the ACEs until a deny (or the end of the list) is reached. At that point processing stops.

UNIX/NFS and NT/CIFS also differ in how they authenticate users. NFS is a connectionless protocol, and each NFS request includes the UID and GIDs of the user making the request. The UNIX client determines the UIDs and GIDs when the user first logs in, by looking at the files /etc/passwd and /etc/groups. CIFS is session based, so the identity of the user can be determined just once, when the session is first set up. At session connect time, the client sends the user’s login name and password to the CIFS server, and the server determines the sessions user SID and group SIDs. Servers in an NT domain environment commonly forward the name, the challenge and the client’s response to an NT domain controller (DC) and let the DC determine the SIDs.

2.2 Important Issues for Non-Native Filesystem Security

The previous section describes the native security models for both UNIX and NT. This section examines the issues that are important for non-native security -- the security for an NFS request to a file with NT ACLs, or a CIFS request to a file with UNIX permissions.

The primary function of any filesystem security model is to validate requests -- to accept them or deny them based on the authenticated user and the permissions of the file. Thus, validating non-native requests is obviously an important topic.

In addition, there are several file system actions that require special attention. In particular, to define how non-native requests are processed, we must answer the following questions:

- How are requests to display permissions handled?

- How are requests to set permissions handled?

- How are permissions set for newly created files?

Handling non-native permissions is tricky, because the NT and UNIX permission models are so different. To define a merged security model, one must specify how an NT ACL will be converted to UNIX permissions, and visa-versa.

3. Handling Non-Native CIFS Requests

This section describes how we handle CIFS requests to UNIX-style files. Remember that files with UNIX permissions occur both in UNIX-trees, where all files are UNIX-style, and in Mixed-trees, which have both UNIX-style and NTFS-style files.

Section 3.1 discusses how non-native CIFS requests are validated, and section 3.2 discusses the non-native handling for displaying permissions, setting permissions, and creating new files.

3.1 Validating Non-Native CIFS Requests

NetApp filers validate CIFS requests to UNIX-style files by generating a mapped UID (and GIDs) for each CIFS session, and then using the UID (and GIDs) to check against the UNIX permissions.

Suppose that the NT user "john" connects to a filer. Here are the steps that the filer will take to determine the mapped UID and GIDs for "john".

(1) The filer sends a request to the NT domain controller (DC), to authenticate "john", and to find out the NT SID for "john".

(2) The filer looks in the user mapping file to determine whether the NT account "john" maps into a different account name under UNIX. In this case, let's assume that "john" maps into the UNIX account "jsmith".

(3) The filer looks up "jsmith" in /etc/passwd (possibly via NIS) to determine the UNIX UID for John.

(4) The filer uses /etc/groups (possibly via NIS) to determine the UNIX GIDs for John.

These steps provide each CIFS session with a full set of UNIX authentication information, which allows the filer to easily validate most requests against the UNIX permissions.

Some CIFS operations don't map well to UNIX operations, so they must be handled specially:

- Set ACL

The CIFS "set ACL" operation is always denied in UNIX-trees. In Mixed-trees, a "set ACL" operation is only allowed by the owner (i.e. the mapped UID for the CIFS session must match the file's UID.) In this case, the file is converted from UNIX-style permissions to NT-style permissions.

Only the owner can set an ACL, because in UNIX only the owner is allowed to set attributes.

- Take Ownership

The CIFS "take ownership" operation is always denied in UNIX-trees. In Mixed-trees, only the file's owner can "take ownership". Like the "set ACL" request, this converts the file to NT-style permissions.

3.2 Request Processing for Non-Native CIFS Requests

This section considers non-native CIFS requests in light of the three questions listed above, in Section 2.2, "Important Issues for Non-Native Filesystem Security":

- How are requests to display permissions handled?

- How are requests to set permissions handled?

- How are requests to create a file handled?

3.2.1 How are requests to display permissions handled?

For non-native CIFS requests to display permissions, WAFL dynamically builds an ACL designed to represent the UNIX permission as best as possible.

On might hope to build an NT ACL that perfectly represents the UNIX permission like this:

Owner - map the file's UID into an NT SID.

Group - map the file's GID into an NT SID.

ACL

ACE for owner SID, based on UNIX user perms.

ACE for group SID, based on UNIX group perms.

ACE for well known NT "Everyone" SID, based on UNIX other perms.

In practice, this doesn't work if we have no way to convert UIDs or GIDs into SIDs. In constructing the ACL, we can only use well-known NT SIDs, and the SID for the CIFS session itself. These are sufficient to let us construct an ACL that, while not perfect, does provide useful information.

Each ACL contains two access control entries (ACEs):

- An entry for NT "Everyone" SID, based on the UNIX "other" permissions.

- An entry for the SID of the CIFS session, based on whichever UNIX permission is appropriate. If the mapped UID is the file's owner, the ACE is based on the UNIX owner perms. If the group matches, then it's based on the group perms. Otherwise it's based on the other perms.

If the CIFS session owns the file, then in the faked up ACL the session's SID is shown as the owner. If not, then the well known NT SID "CREATOR_OWNER" is shown as the owner.

3.2.2 How are requests to set permissions handled?

In UNIX-trees, CIFS requests to set permissions are always rejected. Outside of UNIX-trees, non-native requests to set permissions are allowed only by the file's owner. If allowed, the ACL takes effect just as it would have if the file had been an NT-style file. After the set ACL request is processed, the file becomes an NT-style file.

3.2.3 How are permissions set for newly created files?

Unlike UNIX, which passes the permissions for the new file as part of the create request, NT expects permissions to be inherited from the parent directory.

To handle CIFS create requests in a UNIX-style tree, the filer simply inherits the parent directory's UNIX permissions:

- The file's owner is set to the mapped UID of the CIFS session.

- The file's group is set to the mapped GID for the CIFS session.

- The permission bits are inherited directly from the parent directory, except that SUID and SGID bits are cleared for non-directory creates.

4. Handling Non-Native NFS Requests

This section describes how we handle NFS requests to NTFS-style files -- i.e. files with NT ACLs.

Section 4.1 discusses how non-native CIFS requests are validated, and section 4.2 discusses the non-native handling for displaying permissions, setting permissions, and creating new files.

4.1 Validating Non-Native NFS Requests

Whenever an NT ACL is set or changed, WAFL calculates a corresponding set of UNIX permissions. As a result, very little special processing is required to validate NFS requests to files with NT ACLs. Simply doing the normal checks against the UNIX permissions does (almost) the right thing. The rest of this section describes how the UNIX permissions are constructed from the ACL, and explains the "(almost)" in the previous sentence.

Converting an NT ACL into UNIX permissions is surprisingly tricky. This section gives a brief overview, but an observant user may encounter slight differences in the actual implementation.

- The file’s UID is set to the mapped UID for the CIFS session.

- The file’s GID is set to the mapped GID for the CIFS session.

- The UNIX user perm is set based on the access rights that the ACL grants to the CIFS session creating the file. (These may or may not be explicitly specified by an ACE for the owner.)

- The UNIX other perm is set based on the access rights granted to the NT "Everyone" account. (If the ACL contains any denies, then the denied permissions are subtracted from the other perms.)

- The UNIX group perm is set equal to the other perm.

This design avoids security holes by ensuring that the UNIX permission is always at least as restrictive as the NT ACL is. Unfortunately, UNIX permissions cannot represent the full richness of the NT security model. As a result, a file that a user can reach via CIFS may not be accessible via NFS.

Because NT supports some specific permissions that UNIX lacks, it is not possible to rely entirely on the UNIX permissions to validate NFS requests:

REMOVE/RMDIR

Only the owner of a file is allowed to delete it. This is necessary to avoid violating the NT "delete child" permission.

CREATE

Only the owner of a directory can create anything in it. This is required in order for NT ACL inheritance to work properly.

4.2 Request Processing for Non-Native NFS Requests

This section considers non-native NFS requests in light of the three questions listed above, in Section 2.2, "Important Issues for Non-Native Filesystem Security":

- How are requests to display permissions handled?

- How are requests to set permissions handled?

- How are permissions set for newly created files?

4.2.1 How are requests to display permissions handled?

As described above, in section 4.1, every file with an NT ACL also has a set of UNIX permissions stored with it. To handle an NFS "get attributes" request, the filer simply returns those stored permissions.

4.2.2 How are requests to set permissions handled?

In NTFS-trees, NFS requests to set permissions are always rejected.

In Mixed-qtrees, only a file's owner is allowed to set permissions. When UNIX permissions are set on a file, the NT ACL is deleted; the file changes from NT-style to UNIX-style.

Note that "set attribute" requests that update non-security information such as access time or modify time are allowed even in NTFS-trees, and they do not delete the NT ACL.

4.2.3 How are permissions set for newly created files?

NFS creates in NTFS-trees are only allowed by the directory's owner. This is because an NT SID is required to handle NT ACL inheritance.

An NFS request has no SID, but for a create request from a directory's owner, WAFL can use the owning SID from the directory's ACL and handle ACL inheritance according to normal NT rules. (If the parent directory has no ACL, WAFL follows normal UNIX rules.)

In Mixed-trees, NFS create requests are handled according to normal UNIX rules.

5. Other Issues

5.1 FAT versus NTFS

NT servers support both FAT file systems and NTFS file systems. FAT is the traditional DOS file system -- it has no file-level security at all. The NTFS file system was designed for NT, and it supports a security model based on ACLs.

Since NTFS-trees and Mixed-trees both support ACLs, they must be advertised as "NTFS" file systems. (If they were advertised as "FAT", clients would assume that they had no ACLs, and would disable the interfaces for controlling ACLs.)

It is less obvious how to advertise UNIX-trees. One can make a case either way:

- UNIX-trees don't support ACLs, so advertising them as FAT sends a clear message to clients not to try to use ACLs. Advertising as NTFS would be confusing, since no ACLs are really present and any request to set ACLs will fail.

- UNIX-trees support file level security, and advertising them as NTFS allows the filer to display the UNIX permissions using faked-up ACLs. Advertising as FAT would be confusing, because it would seem to imply that no file-level security is present.

In the end we decided to advertise UNIX-trees as FAT, because this seems least likely to confuse Windows programs that

absolutely must have ACLs.

Still, there are several situations in which it is useful to construct a fake ACL for an NT-style file, as described in section 3.2.1:

(1) In Mixed-trees

Mixed-trees contain files both UNIX-style and NTFS-style files. To support ACLs they must be advertised as "NTFS", yet not all files in them contain ACLs.

(2) In NTFS-trees that originated as UNIX or Mixed-trees

A qtree's security style can be changed at any time, so a tree that began as UNIX-style may later be converted to NTFS. In this case, it will clearly be advertised as "NTFS", but it may contain files without ACLs.

(3) In UNIX-trees accessed via an NTFS or Mixed Share.

The root of a filesystem may have NTFS or Mixed security, but it may contain a UNIX quota tree. In this case, the C$ (or root) share will be advertised as "NTFS", but there is nothing to stop a user from going down into the UNIX qtree and trying to display an ACL.

5.2 Migration from previous versions of Data ONTAP

For sites already using filers with CIFS, migration to the NTFS security model is an important issue.

System administrators can update the security model for any qtree (including the root of the filesystem), using the qtree command. The syntax is:

qtree security <quota tree> [unix|ntfs|mixed]

When a UNIX-tree is converted to an NTFS-tree, shares are advertised as "NTFS" instead of "FAT", and ACLs will be dynamically created for the files as described above in section 3.

When an NTFS-tree is converted to a UNIX-tree, shares are advertised as "FAT" instead of "NTFS", and any ACLs in the tree are simply ignored. ACLs are not actually deleted however, so if the tree is converted back to NTFS, the ACLs will still be present. The best way to delete the ACLs is to write a script that runs as root and chowns each file to its existing owner. (Remember that doing a chown or chmod deletes the ACL on a file.)

5.3 Share Level ACLs

Share level ACLs are also based on NT SIDs, and they can be edited over the network using the NT Server Manager.

For backward compatibility, share level ACLs based on UNIX user names continue to function, although they cannot be controlled via Server Manager.

6. User input into the design

This section describes some of the feedback we received from users during a customer conference on the design. It also describes how our design was changed to incorporate the feedback.

In February 98 Network Appliance invited a group of customers to a day of discussion around the integrated security design.

We particularly wanted to focus on the integration issues that arose and the tradeoffs we'd made. We'd all convinced ourselves that our design was sound but 20 pairs of eyes all focused on just how this new technology would play out in their environment back home would be the reality check we needed.

We had identified one major gap in the phase 1 implementation - the lack of UID to SID mapping - and we were also interested in testing our logic regarding group mapping. In theory, the concept of mapping non-native access into a native context at the user level and then applying the security appropriate to the native user completely removes the need for group mapping. After some research with customers we had concluded that the vast majority of mixed UNIX/NT environments did not have identical group names in NT domains and NIS (e.g. NIS group 'eng' vs. NT global group 'Engineering Users'). So, if the names were not identical, then the best of all possible worlds would remove the need to maintain a group mapping file while also providing seamless mapping. Our answer was to map at the user level and having, for example, mapped a UNIX user into an NT user, apply the group membership of that NT user to the access request.

The UID to SID mapping, already identified as a possible next step, was a hot topic for discussion. Most of the customer representatives felt that for many areas within their organizations the security model would function just fine without UID to SID mapping. Typically, this phase 1 implementation was judged to be sufficient for areas where security was necessary without being too complex. Without UID to SID mapping a file with an ACL could be accessed via NFS by the file owner flawlessly but NFS access requests that came from someone other than the file owner would be evaluated based on the existence of the NT group 'Everyone' in the ACL. If there was no access granted to the group 'Everyone' then the NFS access would be denied. The lack of UID to SID mapping also prevents the ability to update the ACL from NFS: something that our customers requested we consider for a follow-up release. The discussion provided Network Appliance with a clear picture of what the phase 2 implementation needed - UID to SID mapping was clearly necessary based on the feedback we got.

The decision to avoid group mapping was hotly debated and is being considered for the phase 2 integrated security implementation. Clearly, UID to SID mapping fixes a lot of problems and Network Appliance intends to invite customers back to review our revised design and preliminary implementation of this which will be based on feedback from phase 1 usage.

7. How the design works in practice

This section focuses on the real world experience of using the merged security model and gives examples of how one customer has successfully used the different security styles to share data in a mixed UNIX/NT environment.

Glaxo Wellcome, a UK-based Network Appliance customer, has been using filers for some time in a predominantly NFS environment, waiting for key Windows features before moving to consolidate mixed data on the filers. They spent some time testing the Network Appliance Data ONTAP 5.1 release, evaluating its usefulness in their specific environment. They are migrating from a VMS/Pathworks environment to using Windows NT and UNIX servers and are looking to offer users the same access to shared data with the same granularity of file level permissions which the VMS/Pathworks combination gave them.

One such group of users have moved from the VMS/Pathworks environment. Data is exported from a UNIX based Oracle database and is then analyzed by a NT based application., giving users a graphical interface to Pathology data, helping to more readily identify trends in the data.

The filers are in the Glaxo Wellcome Research division, an environment with UNIX, Windows NT and Windows 95 clients. Both UNIX and PC clients generate data which then has to be available to others. The PC clients run Microsoft Office applications to generate files that are then available to UNIX applications which incorporate this data into reports and web pages. The UNIX clients export data from Oracle databases into flat files which are then picked up by PCs and incorporated into technical reports, or analyzed by Windows based applications.

The Glaxo Wellcome requirement for a common data location that allows all users to access data independent of client platform is becoming increasingly common.

The Glaxo Wellcome R&D environment is rapidly migrating to a mixed UNIX/NT one, where Unix provides the database engine, and NT the reporting and GUI tools. The filer gives interoperability between the two platforms, without the database engine having to perform unnecessary file sharing operations via Samba, or similar. In addition, by making UNIX data directly available to NT users, those users are able to analyze the data using the Microsoft programming tools, with which they are familiar.

Since the Glaxo Wellcome R&D users were used to using VMS ACLs the ability to provide NT ACLs was an important piece of the solution. Providing Windows users with familiar ACLs also cuts down on calls to the HelpDesk since system administrators no longer have to explain UNIX file perms to Windows users. The system administrators see the filers as a simple solution to the complex problem of disk usage. Glaxo Wellcome uses Solaris, HP-UX and SGI IRIX plus Windows NT running on Compaq and HP servers. As usage of these servers changes the proprietary disks attached to the servers are left unused. Being able to host data on one platform gives much more flexibility during migration and makes better use of the investment in disks.

8. Real-world migration issues

This section covers some of the migration issues involved in moving from a UNIX security model to a merged NT/UNIX security model and describe some of the tools users need to ease the transition process. It also discusses some of the utilities that are useful but are beyond the scope of a pure fileserver appliance: account synchronization tools and authentication modules such as Nigel Williams' NIS GINA for NT.

8.1 Conversion from UNIX permissions to NT ACLs

For those who decide to move from a single file level security model to a merged environment the first question usually asked is 'Will I lose all my UNIX perms on those areas where I now need NT ACLs?' No one wants to ask their users to go in and re-set file permissions to take advantage of the new functionality. We realized early on in the design process that the need to fake up an ACL from a set of UNIX perms would be an important one, particularly during migration. Since we had many Windows customers using UNIX file security we toyed with a one-time conversion tool which would read the UNIX perms on every file and set an ACL, using a mapping file which the administrator would set up in advance. This idea was popular during the customer discussions but no one wanted to set up the mapping file if they could help it. The solution we opted for was to fake up an ACL when a Windows user in a freshly converted NTFS tree viewed the security on a file. The faked up ACL would then be replaced by a real one when the user clicked OK, even if no changes where made to the displayed ACL information. Thus, over time, an NTFS qtree would migrate to NT ACLs. For phase 1, the UNIX permissions would still be stored for the file (and recomputed if the ACL changed) for NFS access. A phase 2 implementation would not require the UNIX perms since UID to SID mapping would allow NFS access to be evaluated directly against the ACL itself.

8.2 User Authentication - NIS or NT Domain?

Several UNIX/Windows file service solutions can be configured to authenticate Windows users via local /etc/passwd (or NIS) or a Windows NT domain controller. As examples, NetApp filers and Samba can be configured in this way. Many administrators would like to authenticate all users using just one authority; NIS or NT domain controllers but the password encryption issues make this more difficult than it should be. Windows NT 5 with support for Kerberos 5 will allow UNIX users to authenticate via NT KDCs. Today Samba can be used to authenticate Windows users via the *smbpasswd* file. Using *pwdump*, encrypted passwords can be downloaded from an NT domain database, stored in *smbpasswd* and used to authenticate Windows users without forwarding to an NT domain controller for verification.

One problem with using something other than an NT domain controller for Windows user authentication today is the lack of user information available when an alternative method is used. An NT domain controller will, if the requesting party supports NetLogon, return a great deal of useful information about the user being authenticated. One of the most useful pieces of information from a file level security perspective is group membership. An NT domain controller will validate the user request and return the user's domain SID, the SIDs of the domain global groups to which the user belongs, account restrictions and other useful information. Therefore today a NetApp filer has to be a member of an NT domain in order to offer the NTFS security style. The Samba team are working to provide equivalent functionality from a Samba server.

8.3 Copying security information between servers

Support for utilities that set and copy security information is a must when transferring files during migration. *Cacls.exe*, a Windows NT utility that ships with the Windows NT 4.0 Resource Kit which allows ACL editing from within a Windows NT command shell (useful for scripting large scale security changes). *Scopy.exe* is another very useful commandline tool which allows you to copy ACL information when you copy data. Without using *scopy.exe* (i.e. if you used the *copy* command instead) the copied file would inherit the ACL of the parent directory into which it was copied. For the large scale copying of data that is usually a part of migration process *scopy* can be extremely useful.

Another migration strategy is to backup data from one server and restore onto the new server using backup tools that retain all file metadata. The NetApp integrated security design includes support for Windows NT backup utilities, allowing you to backup data from an NT server and restore it directly onto a filer, preserving all the ACLs. This is possible due to the way the filer stores NT Security Descriptors in the WAFL filesystem combined with the ability to package up UNIX file metadata into Extended Attributes which the NT backup software understands.

8.4 Mapping user names

Maintenance of the username mapping file can become problematic when you are administering a group of machines rather than just one or two. One suggestion made during our customer conference was that the username mapping file should be NIS-distributable. A mechanism for centrally maintaining the username mapping file is under development and NIS distribution is one of the possibilities. We are also looking at an LDAP solution to this problem.

9. Futures

The most significant addition to the phase 1 implementation is clearly the UID to SID mapping. We are planning to use the NFS username to map to an NT SID. Unlike CIFS which is a stateful protocol, NFS' statelessness requires a mapping of username to SID for every request. We are designing a cache into the UID to SID mapping to address this problem. Group mapping is still an open question for reasons discussed elsewhere in this paper. Other additions to the integrated security features will be based on customer feedback of the phase 1 implementation.

10. Related Work

This section describes what we learnt from other approaches to solving the problem. We will cover Samba and AT & T Advanced Server for UNIX (AS/U) specifically, comparing our approach to two different ways of mapping NT style security onto a UNIX filesystem, one providing mapping between Windows and NFS users and the other keeping the security information separate.

Both Samba and AS/U have a somewhat different design philosophy from that of a filer in that for both these solutions the UNIX file system is the arbiter of access and metadata storage. PC file attributes may be generated on the fly (e.g. 8.3 file names in Samba) or stored in a separate file which is then linked to the object it pertains to (e.g. AS/U's /var/asu/.db).

AT & T's AS/U stores ACL information and other PC style metadata in a database (typically stored in /var/asu). Depending on the licensee's implementation of the AS/U product there may be no attempt to map file level security information between NFS and CIFS users. This means that access to a file with an ACL on it must first be checked against the ACL and then again by the UNIX permissions also on the file.

The current version of Samba does not include support for Windows NT style ACLs but more of the groundwork appears in each interim release and the ACL design is already in an advanced state. One key difference between the Samba design and that of Network Appliance is that the Samba priority has been to design the mapping between NT SIDs to UNIX UIDs and GIDs before implementing the NT ACL support. The Samba goal of replacing NT as a PDC in a domain (already working to a limited extent in current Samba pre-alpha code) means that this work has been given a higher priority that the ACL design. Current Samba Team public discussions on their ACL design seem to be focused on providing a simple but user configurable, on-the-fly mapping between UNIX permissions and NT ACLs.

11. References

1. Rob Reichel , *Inside Windows NT Security* , *Windows/DOS Developer's Journal.* April & May 1993.

2. Stephen Sutton, *Windows NT Security Guide*, ISBN 0201419696

3. Andy Watson, *Multiprotocol Data Access: NFS, CIFS, and HTTP (TR-3014)*, Network Appliance, Mountain View, California, December 1996.

4. Andrea Borr, Keith Brown, *SecureShare™: Guaranteed Multiprotocol File Locking (TR-3024).* Network Appliance, Santa Clara, California,
1997

5. Samba man pages at a local mirror. Pick from the list at http://samba.anu.edu.au/samba/