



WHITE PAPER

Understanding Windows & UNIX File Permissions on GuardianOS®

Introduction

Overland Storage® Snap Server® storage systems powered by the GuardianOS™ operating system support many file access protocols that allow access to files from Windows, UNIX, and Apple clients; as well as remote access via FTP and HTTP. Although GuardianOS runs on an optimized Linux kernel and has many characteristics of Linux, the integrated cross-platform features make it very different than a pure Linux distribution. Administrators should not expect the same behaviors or to administer the system like a pure Linux distribution. Instead, they should view systems running GuardianOS as special-purpose storage appliances.

The document is intended to introduce security concepts as they pertain to file services and to provide guidance on how to best configure Snap Servers running GuardianOS for integration into networks supporting Windows, UNIX, and Apple clients. Since different file access protocols have very different security architectures, some concessions must be made to allow for simultaneous access and predictable behavior for client permissions. How Snap Servers integrate into this type of mixed access environment will be further explored and described in this document.

Note: Snap Servers powered by GuardianOS also support block-level services via iSCSI, but this topic as well as many other features will not be covered in this document.

In any cross-platform file sharing environment supporting both Windows clients and clients using a UNIX-style security paradigm, the primary point of contention exists when a Windows client tries to access a file with UNIX permissions, or conversely when a UNIX client tries to access a file with Windows permissions. In both cases the accessing client expects to set and enforce permissions based on the rules of the native file system of its OS. In some cases these permissions can be easily translated and in others there is no direct correlation.

With the introduction of GuardianOS 5.0, native Windows permissions have been implemented that provide 100% Windows compatible permissions handling. Additionally, this new permissions infrastructure enables permissions enforcement in a cross-platform environment to be completely predictable without artificial permission translation. This is a unique benefit when compared to other Linux-based NAS solutions.

Terminology

When describing the client type, this document uses the terms Windows client and UNIX client. For purposes of this document a "Windows client" refers to a workstation, PC, or server that utilizes the CIFS (Common Internet File System) file sharing protocol developed originally by Microsoft® and based on the earlier SMB (Server Message Block) file sharing protocol. A "UNIX client" refers to a workstation or server that utilizes the POSIX permission paradigm that assigns read, write, and execute permissions to owners, owning groups, and "other", including clients connecting over the NFS (Network File System) protocol and Mac OS X clients connecting over AFP (Apple FileShare Protocol).

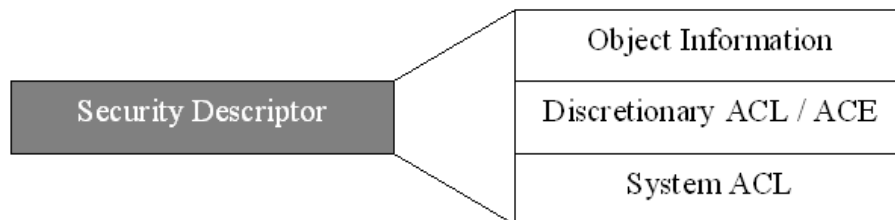
Two terms that are generally used interchangeably in the computing world are "folder" and "directory". For purposes of this document the term "folder" will refer to the container in a Windows environment that holds files and other folders. The term "directory" will refer to the container in a UNIX environment that holds files and other

directories. Note that the only real distinction is that Windows environments use “folders” and UNIX environments use “directories”. Mac OS X users will be accustomed to using the term “folder”, but since Mac OS X clients will follow the UNIX security paradigm when accessing data on GuardianOS, this document will use the “directory” term for Mac OS X. Since GuardianOS is based on Linux, all explicit references with regard to GuardianOS will use the “directory” nomenclature.

Another term that is important to understand is Access Control List (ACL). An ACL is a collection of Access Control Entries (ACE) for each user or group that has permissions defined on the object to which the ACL is attached. An ACE consists of a security identifier to specify a user or group and an access mask to define the permitted (or denied) access level for that user or group. In a native Windows NTFS file system security paradigm, an ACL is contained inside a Security Descriptor (SD), which defines additional security-related parameters for the file or folder to which the security descriptor is attached. In a Windows SD, Security IDs (SIDs) are used to uniquely identify all security-requesting entities in a Windows environment, including users, groups, and other special types of objects.

An SD contains three major components for defining access control for a file or folder as shown in Diagram 1.

Diagram 1 – Make up of Windows Security Descriptor



- Object Information – Contains the object owner's SID. For files and folders, it will also contain the location as a folder path.
- Discretionary ACL (DACL) / ACE – The DACL lists SIDs for all users and groups that have been granted or denied access to this object. The list is comprised of ACEs, each defining the access for a SID.
- System ACL (SACL) – Used to control the auditing messages that will be generated. These are used to create entries in a Windows Security audit log.

Note: Support for SACLs is currently not included in GuardianOS.

In this document the term “ACL” will be used to generically describe the permissions and access control associated with files and folders following a Windows security paradigm, and the term “ACE” will be used to describe a specific user or group entry within an ACL.

Two more terms that will be used throughout this document are the UNIX terms that describe the user and group identifiers. A User Identifier (UID) is a number that is unique to a particular user. The operating system uses the UID to uniquely identify a user within the system. For example, a UID might represent the owner of a file or process, or it may identify the user who is attempting to access a system resource. A

Group Identifier (GID) is a unique number that identifies a set of users, referred to as a group, under UNIX. GIDs associated with each file, directory, or system resource designate the ownership or access permissions for all users who belong to that group.

The following terms refer to aspects of the types of file system security that can be applied to Windows, UNIX, and GuardianOS servers:

- Security Paradigm: the method of file system security inherently associated with a given OS platform, including Windows and UNIX.
- Personality: the type of security paradigm applied to a specific file or directory on a GuardianOS server.
- Security Model: the rule applied to a GuardianOS SnapTree to define what types of personalities are permitted and whether those personalities can be modified.

UNIX Security Paradigm

In UNIX, permissions are represented as three triplets. Each triplet represents the access privileges to read, write, or execute for a specific file or directory. There is also a leading field that designates whether the object is a directory, symbolic link, or file (the default when blank).

Here is an example of a typical directory listing:

	1	2	3	4	5	6-8	9
	↔ (Permission Flag Bit Positions)						
	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓						
	drwxr-xr-x	1	jdoe	enggrp	12040	Aug 20 08:51	draft
	-rw-----	1	jdoe	enggrp	2048	Dec 23 16:11	permission-table.xls
	-rw-r--r--	1	jdoe	enggrp	2048	Dec 23 16:11	unix-perms.doc
	Lrwxrwxrwx	2	jdoe	enggrp	2048	Sep 13 10:23	unix-perms.doc
Field	1	2	3	4	5	6-8	9

Table 1 – Definition of above fields

Field 1:	Set of ten permission flags.
Field 2:	Link count
Field 3:	Owner of the file
Field 4:	Associated group for the file
Field 5:	Size in bytes
Field 6-8:	Date of last modification (format varies, but always 3 fields)
Field 9:	Name of file (possibly with path, depending on options used for 'ls')

Table 2 – Description of each bit position within permission flags (left to right)

1	Designates whether or not it's a directory. 'd' for a directory, '-' for a normal file. Something else occasionally may appear here for special devices, For example, 'l' for a symbolic link.
2, 3, 4	Read, Write, Execute permission for User (Owner)
5, 6, 7	Read, Write, Execute permission for Group
8, 9, 10	Read, Write, Execute permission for Other

Table 3 – Definition of permissions

-	The flag is not set.
r	The File or Directory is readable by owner, group or other
w	The File or Directory is writable by owner, group or other
x	The File or Directory is executable by owner, group or other
s	In the place of 'x' is the set-UID or set-groupID flag
t	In the place of 'x' is the "sticky bit"

An executable program with set-UID or set-groupID runs with the effective permissions of its owner or group.

Another permission that is used is called the "sticky bit". Back when machines did not have very much memory, the sticky bit was used to force a program or file to remain in memory. The sticky bit originally had no relevance for directories, but was later overloaded to control the ability to rename or delete files and directories inside a parent directory. When enabled on a parent directory, only the 'superuser', or the owner of the parent directory, or child file or directory can rename or delete files, regardless of other permissions. The `/tmp` directory normally has a sticky bit set. Executing a directory listing (`ls -l /tmp`) on a directory where the sticky bit is set might look like this: `drwxrwxrwt 5 sys sys 543 May 29 09:41 tmp`

When performing validation, UNIX first determines whether the request is from the file's owner, someone in the file's group, or anyone else. Permissions are not processed by the union of all permissions. The user is granted only the first access level that applies, evaluated in order: 'user', 'group', and 'other'. Access is denied if there is no positive result.

As mentioned above, Mac OS X clients also use standard UNIX permissions that apply to and behave largely the same as other UNIX clients. These can be configured from a Mac OS command line shell using `chmod` in the same manner as with any other UNIX client; however, the Mac OS GUI abstracts these permissions to "Read & Write", "Read Only", "Write Only", and "No Access", which are translated by the client to the corresponding UNIX permission when configured (`rwX`, `r-X`, `-wX`, and `---`, respectively).

Windows Security Paradigm

Windows clients have distinct security paradigms that apply to the different file systems that are available. For FAT and FAT32 file systems there are no file level permissions. Anyone who is able to access the files either locally or remotely has full access to all files and folders on the file system.

Since the introduction of Windows NT, Windows platforms have been able to utilize a more advanced file system called NTFS (New Technology File System). NTFS includes features that improve reliability, such as transaction logs to help recover from disk failures. To control access, permissions can be set for folders and/or individual files.

Windows also allows administrators to apply permissions to shares, which are enforced independent of the security established on the files and folders that are accessed via those shares.

Windows Permissions

NTFS contains highly granular access control permissions that are stored as ACLs within each file and folder. These ACLs allow administrators the capability to tightly and flexibly manage access to data, and are the heart of all access control within NTFS.

The “Standard Permissions” shown on the main Security properties page for a file or folder are:

- Full Control
- Modify
- Read & Execute
- List Folder Contents
- Read
- Write
- Special Permissions (this means there are permissions outside of above schemes)

Standard Permissions are summary groups of multiple “Special Permissions”. Table 4 illustrates how Standard Permissions translate to the underlying Special Permissions. Standard Permissions can be accessed from a Windows system and are found on the main Security tab of a file or folder’s Properties window, which can be accessed by right-clicking on the file or folder and selecting Properties. Special Permissions can be accessed by clicking on the Advanced button in the Security tab.

Note: For clarification, Standard Permissions consist of inheritance and propagation properties, as well as the composite of the “real” permission bits. If any of these are changed in the “Advanced” view, the permission that displays on the main Security tab is displayed as a “Special Permission” instead of one of the Standard Permissions schemes that it may have shown as “checked” prior to changing any of the inheritance or propagation properties, even if the individual permission bits have not changed.

Table 4 – Standard Permissions are translated to multiple Special Permissions

Resulting Windows Special Permissions that are Set														
Specific Permissions	Full Control	Traverse Folder/Execute File	List Folder/Read Data	Read Attributes	Read Extended Attributes	Create Files/Write Data	Create Folders/Append Data	Write Attributes	Write Extended Attributes	Delete Subfolders and Files	Delete	Read Permissions	Change Permission	Take Ownership
Full Control	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Modify		x	x	x	x	x	x	x	x		x	x		
Read & Execute		x	x	x	x							x		
List Folder Contents		x	x	x	x							x		
Read			x	x	x							x		
Write						x	x	x	x					

Windows and NTFS offer an additional dimension for setting permissions that are not explicitly available within UNIX. NTFS enables Administrators to set “Allow” access for the above list of permissions, as well as the option to explicitly “Deny” permission to specific users and groups. In a Windows ACE, the “Deny” permissions are generally

processed first before “Allow” permissions. Due to this processing order, any “Deny” permission usually overrides conflicting “Allow” permissions (subject to inheritance precedence). Permissions processing details and how Deny permissions are enforced will be discussed in detail later.

Windows Inheritance

Inheritance is another very important concept for configuring permissions in Windows. Inheritance allows for the higher level folder permissions to flow down and control permissions for all subordinate files and folders.

Inheritance within Windows refers to the process by which child files and folders obtain permissions from parent folders. More simply put, child files and folders can be set to “inherit” their permissions from the parent folder, and individual ACEs on a parent folder can be set to individually propagate to all or only certain sub-files and subfolders.

In an NTFS ACL marked to inherit permissions from its parent, each individual ACE has a bit that determines whether it is inherited from a parent folder (to distinguish it from other ACEs not inherited on the ACL). This is sometimes referred to as implicit inheritance (though technically the ACE is explicitly set on the ACL). With NTFS implicit inheritance, if there are many levels of folders that all inherit permissions from a single parent folder, a change to the permissions of the top-level parent folder will affect the permissions of the entire folder structure. Windows then automatically propagates this change to sub-files and subfolders according to the established inheritance rules set on the ACLs on all files and folders in the folder tree.

NTFS has seven different types of inheritance rules that can be set on individual ACEs (See Table 5). These rules provide for granular and flexible configuration of automatic propagation of a given ACE to sub-files and subfolders configured to inherit permissions from their parent. For example, an administrator could add three ACEs for user “jdoe” to a folder ACL to give “jdoe” Full Control on Files, Read Only on Subfolders, and Read/Write on the folder itself. Again, this can be done for each user and group that has assigned permissions to that folder.

Table 5 – the seven ACE inheritance rules

This folder only	ACE applies access control only to the folder being modified
This folder, subfolders, & files	ACE applies access control to the folder being modified and all sub-files and subfolders
This folder & subfolders	ACE applies access control to the folder being modified and all subfolders
This folder & files	ACE applies access control to the folder being modified and all sub-files
Subfolders & files only	ACE applies access control to sub-files and subfolders only, and does not control access to the folder being modified
Subfolders only	ACE applies access control to subfolders only, and does not control access to the folder being modified
Files only	ACE applies access control to sub-files only, and does not control access to the folder being modified

Note: in each of the cases above, access control applied to sub-files and subfolders only applies to those which are configured to inherit permissions from their parent.

ACEs can also be set to apply only to a single level ("Apply these permissions to objects and/or containers within this container only"). This prevents ACEs from propagating further than a single folder level, regardless of the inheritance rule set on the ACE.

The lack of inheritable ACEs on a folder (when all ACEs are set for "This folder only") can lead to unexpected results for newly created files or folders. If a user creates a file or folder within a folder with no inheritable ACE's, instead of the file or folder being created with an empty ACL, the creator user and the special SYSTEM object are given Full Control. GuardianOS has other scenarios where this can occur and will be detailed later in this document.

Windows Permission Processing and Precedence

Most security permissions in Windows are set from the standpoint of determining which users and groups are "allowed" access to files or folders. In these circumstances, all permissions that are applicable to a given user by virtue of multiple ACEs assigned either to that user or a group to which that user belongs (including special groups like "Everyone"), are summed and apply equally to that user. There are, however, circumstances in which explicit "deny" permissions are important to an environment's security scheme and impact the precedence of permissions applied to a user.

The order in which permissions for Windows files are processed can be seen when looking at the Advanced Permissions tab in Windows while sorted by the "Type" column. This shows the order of precedence in which the system checks for rights when a user requests access.

The first ACEs to be processed are the ACEs assigned to the file or folder (i.e. ACEs not inherited from a parent directory) that explicitly Deny access to the file or folder. All such "deny" ACEs that are applicable to a given user are summed and applied to that user. This means that an explicit Deny ACE set on a file or folder for a given right will override all other ACEs that grant that right to that user or group, either explicitly or inherited for the file or folder.

The next ACEs processed are the ACEs that explicitly Allow access to a file or folder (again, this refers to ACEs that are not inherited from a parent directory). All such "allow" ACEs that are applicable to a given user are summed and applied to that user. This means that an explicit Allow ACE set on a file or folder for a given right will override all other Deny ACEs inherited from a parent directory that deny that right to that user or group.

The next ACEs to be processed are the inherited Deny ACEs from the parent directory, followed by inherited Allow ACEs from the parent. Similarly, this is followed by Deny and Allow ACEs inherited from the grandparent, great-grandparent, and all other directories above the object in question, from which it inherits ACEs. In all inheritance levels, all Deny permissions at a given level are summed and take precedence over Allow permissions at that level, which are summed themselves and take precedence over Deny permissions on the next level up, until all inherited permissions are accounted for.

Note: The ability for the Properties window to properly display permission precedence order and to identify the parent directory from which a given ACE is inherited is dependent on the current user's ability to read permissions on all applicable parent directories. If a given ACE is inherited from a parent directory to which the current user does not have Read Permissions access, the ACE will be listed as inherited from "Parent Object" and will not necessarily be displayed in the correct precedence order; however, the ACL is still enforced following the precedence rules.

Group Policy Objects

In a domain environment, Group policies provide a way to reduce administrative overhead and the TCO of network administration. Administrators configure and deploy Group policies by building Group Policy Objects (GPOs). GPOs can both specify configuration settings and contain a list of users and groups that are granted specific privileges that override all other security configuration (including NTFS permissions). A GPO can specify numerous privileges, such as the ability to arbitrarily set or take ownership of any file or folder, and can define various configuration options including a set of applications to be installed on all user desktops, disk quota policy, and Password and Account Lockout policies.

Both Local and Domain GPOs exist. Local GPOs only affect the local machine, while Domain GPOs are applied to all computers within a Domain.

Some common policies that apply to file access include:

- Password policies
- Account lockout policies
- Quota policies
- File System Security settings
- Restricted Group Security
- Auditing policies
- User Rights Assignment

Guidance on deployment of these policies and how they can be utilized is outside the scope of this document, but can be researched using Microsoft Windows specific resources.

Share-Level Permissions

Share-level access control to storage resources is the most basic level of security for file sharing in a Windows environment. Share permissions are checked only when a user attempts to access a shared resource across a network. Since a share is always defined at the folder level, share permissions are also defined and enforced at this level. When share permissions are established, the permissions are equally enforced for all files and folders accessed through that share over the network; but share-level permissions cannot be used to independently protect individual files or folders within a share, nor do they apply to users accessing those files and folders – neither locally, nor through a different share.

Share permissions are modified through the folder's Sharing properties window, which is accessed by right-clicking on a Windows folder. Clicking on the Permissions button will allow you to view and modify the share permissions. If the same folder has been shared with multiple share names, permissions for each share must be managed separately.

Share permissions are specified using an ACL. However, it should be noted that this ACL is completely independent from an ACL set on a file or folder, and unlike file and folder ACLs, only has basic control available to it. There are three permissions that can be set for a user or group on a share ACL:

- Full Control – Allows user/group to read, modify, and change share-level permissions
- Change – Allows user/group to modify contents, add files, and delete files from a share
- Read – Allows user/group to view the contents of a share and read and execute files

Note: Share permissions are automatically applied to all files and folders within a shared folder. There is no way to control the inheritance of these permissions as you can with NTFS permissions, which will be discussed in detail later.

By default the “Everyone” group is given Full Control access to a shared folder. Most administrators either remove this group completely, or limit the permissions for the “Everyone” group.

Permissions enforcement can be handled completely at the file and folder level, but a combination of share-level and file/folder-level permissions can provide administrators with a very powerful security control model.

DOS Attributes

All Windows file systems use a collection of attributes and permissions to collectively form a system’s overall security policy. Attributes originated from early versions of Microsoft’s MS-DOS, and are often referred to as “DOS Attributes”. There are specific attributes that relate to each folder and file, and are stored in the file’s header. Table 6 and Table 7 list DOS attributes. Note that Table 7 only applies to NTFS file systems.

Table 6 – Attributes that apply to all DOS and Windows file systems

Attribute	What it means
Read Only (R)	Identifies write protection
Hidden (H)	Determines whether the file is hidden from user queries
System (S)	Identifies if the file is reserved for the system
Archive (A)	Identifies if the file is an archive file – used for backup
Create Date	Date the file was created
Modify Date	Date the file was last modified
Access Date	Date the file was last accessed

It is important to note that the traditional RHPA attributes are not security attributes. Enforcement of their characteristics is advisory, and is therefore completely dependent on the client application.

Also note that the last three attributes (Create Date, Modify Date and Access Date) are commonly referred to as “time stamps” and are therefore not always considered DOS Attributes.

There are other NTFS “Windows attributes” that Microsoft created as part of NTFS. They do not apply as DOS attributes as described above, but can have an effect on file behavior. In some cases, they are part of security access control mechanisms and are described in more detail in other areas, as applicable, within this document.

Table 7 – Additional Windows attributes that only apply to NTFS

Attribute	What it means
Ownership	Identifies the file owner (Stored in the Security Descriptor)
Encrypted	Reflects whether or not the file is encrypted
Inheritance	Reflects if the file/folder is set for automatic inheritance (Stored in the Security Descriptor)
Alias	Indicates that the file is a shortcut to another file/folder
Compression	Reflects whether or not the file is compressed

GuardianOS Security Paradigm

GuardianOS employs a hybrid security paradigm to best match the security used and expected by the various types of client operating systems accessing the Snap Server over the network, and to properly enforce security between clients of differing security paradigms. In a mixed UNIX and Windows client environment, GuardianOS attempts to mask and match permissions between the different network access methods. The permissions that are applied upon access of a file or directory are dependent upon the security personality of that file or directory. File system permissions also work in conjunction with permissions established on the share or export by which the file system is accessed.

Security Personalities on GuardianOS

GuardianOS 5.0 introduces a new concept of security “personalities” to better handle cross-platform permissions enforcement.

In GuardianOS 5.0, Windows permissions are implemented natively. This gives Windows users all of the file system access control to which they are accustomed on an NTFS file system in a Windows environment. Therefore, files and folders following the Windows security paradigm have the “Windows personality” and behave as though they had been created on a Windows server.

Similarly, files and directories created and controlled by UNIX users (including traditional UNIX and Linux clients, as well as Mac OS X clients) follow the UNIX security paradigm and therefore have the “UNIX personality” and behave as though they had been created on a UNIX server.

Note: There are technically several UNIX security personalities that match the various client operating systems and protocols, but since they all follow the same paradigm, they are collectively referred to as “UNIX personalities”.

Files and directories using both Windows and UNIX security personalities can coexist on a Snap Server and can be accessed by clients of either security paradigm. When a client attempts to access a file or directory of a given personality, the enforcement rules of the file or directory’s personality and the associated access permissions are applied, regardless of the native security paradigm of the client.

When viewing security settings on a file or directory, permissions are presented in a format that matches the native security paradigm of the client OS that is used to view permissions. When necessary, the true security configuration of the file or directory is roughly translated for client presentation; but as a consequence, security configuration of one personality cannot be accurately viewed from a client using a different security paradigm. See “Windows Personality” and “UNIX Personality” sections below for additional detail.

The personality of a file or directory can be changed if necessary (as long as the security model of the parent SnapTree permits personality changes). This is normally performed transparently by changing permissions from a client matching the desired security personality (subject to the privileges of the user attempting to perform the permission change). For example, if a file has a Windows personality and an NFS client with sufficient privileges (as established by the Windows ACL on the file) uses **chmod** to apply UNIX permissions to the file, the file will change to a UNIX personality.

Subsequently, the Windows ACL will no longer exist; only the new UNIX permissions will be enforced. This same principle applies whenever permissions or ownership are set

on a file or directory by a client: if the client operating system matches a different personality than the one currently set on the file or directory, the personality of the file or directory is changed to match the client and the security established by the client is subsequently enforced.

The personality of a file or directory can also be changed by modifying the security model of a SnapTree in which the file or directory exists. This will be described in more detail when SnapTrees are discussed. For now, just recognize that changing the security model of a SnapTree can recursively modify the personality of each file and directory within that hierarchy to match that of the SnapTree type.

Windows Personality

Since Windows permissions are 100% native and compatible with Windows in GuardianOS 5.0, manipulating and controlling complex permissions schemes for Windows clients is much easier and more intuitive than in previous GuardianOS releases. As detailed earlier, NTFS supports 13 distinct Special Permissions. GuardianOS supports all of these permissions, as well as the six Permission Templates most frequently used. Controlling access for a homogeneous Windows environment can be performed just as it would in a pure Windows environment. All applicable permissions for a given user are equally considered, and precedence from Deny, Allow, and inherited ACEs is respected as if it were on a true Windows NTFS file system.

Since GuardianOS supports mixed access from both Windows and UNIX clients, it is important to understand how permissions are handled for UNIX clients when the file has a Windows personality. The implementation on GuardianOS simplifies cross-platform access control by adhering to one simple rule: permissions are processed based solely on the personality of the file or folder. This means that specific user or group permissions that would normally not be able to be granularly controlled for UNIX clients via NFS can be controlled within the ACL of a file or folder with a Windows personality. Since the Windows NTFS security paradigm provides much more granular control over file access permissions, Windows personality files and directories contain additional permissions enforced for UNIX clients that may not be expected by users accustomed to a UNIX security paradigm.

Additionally, the presentation of Windows ACLs on GuardianOS files and directories to clients is determined by the client OS. Windows clients from Windows 2000 and above will see Windows ACLs on GuardianOS files exactly as they exist on the GuardianOS file system. Windows NT clients will see a simplified version of the ACL that is consistent with Windows NT's simplified security paradigm; it will be reported as if the file existed on a Windows 2003 server with exactly the same ACL. Older ("Windows 9x") versions of Windows cannot view or configure ACLs.

UNIX clients viewing permissions on a Windows personality file (e.g. using `ls` or `stat`) will always see "777" or "`rwxxrwxrwx`", regardless of the actual permissions on the file. This is done because the depth of permission configuration possible in a Windows ACL cannot be properly represented to the limited UNIX security paradigm, as well as to ensure that only GuardianOS can control the level of access that is allowed for a given user on a given file. In some cases, if UNIX permissions are not reported as "777", clients and client applications may artificially restrict privileges on themselves, even if the desired access would have been permitted by the Windows ACL established on the file. It is important to note that in this circumstance the "777" UNIX permissions are for display purposes only. The Windows ACL assigned to the file stored on a GuardianOS-powered Snap Server will be fully enforced.

Owning User and Group reported to UNIX clients are dependent on the Windows owner. See the "Owning Users and Groups in GuardianOS" section for more detail.

UNIX Personality

For environments that need to have mixed access to files, but have strict requirements that permissions be enforced with a traditional UNIX style of permissions processing, files can be set to have a UNIX personality in GuardianOS and will therefore follow the UNIX permissions paradigm.

Files or folders with a UNIX personality are still accessible from all file access protocols supported on GuardianOS, just like files or folders with a Windows personality. The same principle that applies to UNIX clients accessing Windows personality files applies to Windows clients accessing UNIX personality files: permissions are processed based solely on the personality of the file or folder. UNIX permissions are enforced on Windows clients following the UNIX permissions and security paradigm, as if they were on UNIX clients.

As with accessing Windows personality files from non-Windows clients, the presentation of UNIX permissions to non-UNIX clients is dependent on the client operating system. A Windows client viewing permissions on a UNIX personality file will see an approximate representation of permissions set for the owning user, owning group, and other (listed as "Everyone"). UNIX permissions are translated to Windows Special Permissions, as detailed in Table 8.

The owner of the file displayed in a Windows Properties window will be that of the UNIX file owner.

Table 8 – UNIX permissions translated for Windows Clients

	Resulting Windows Effective Permissions that are Set												
UNIX Permissions	Traverse Folder/Execute File	List Folder/Read Data	Read Attributes	Read Extended Attributes	Create Files/Write Data	Create Folders/Append Data	Write Attributes	Write Extended Attributes	Delete Subfolders and Files	Delete	Read Permissions	Change Permission	Take Ownership
Read (r)		x	x	x							x		
Write (w)					x	x	x	x			x		
Execute (x)	x		x								x		

Owning Users and Groups on GuardianOS

Ownership of files and directories has varying relevance, depending on the security personality of the object and the security paradigm of the client. Additionally, ownership has special meaning with respect to quotas.

Windows personality files and directories are owned by either a user or a group. Identification of the owner is stored in the Windows Security Descriptor, and is visible from a Windows client in the Properties window for the object. The owner or a member of the owning group always has implicit permission to change permissions on the object, regardless of the other permissions on the ACL. Moreover, unlike the UNIX security paradigm, permissions in an ACL assigned specifically to the owner of the file

have no more relevance to the effective permissions of that user or group than any other permissions that apply to that user or group (since all permissions applicable to a given user are summed).

Unlike the UNIX security paradigm, there is no concept of both an owning user and an owning group on a Windows personality file – there is only one or the other. However, for purposes of reporting UNIX permissions to UNIX clients, both an owning user and an owning group are reported, based on whether the “Windows owner” is a user or a group:

- If the Windows owner is a user, the owner is reported to UNIX clients as the owning user and “admingrp” (GID 0) is reported as the owning group
- If the Windows owner is a group, the owner is reported to UNIX clients as the owning group and “admin” (UID 1) is reported as the owning user

Ownership of a Windows personality file also has special relevance to quotas. Quotas only observe the “UNIX user owner” of files, regardless of personality. Therefore, although space consumption for a Windows personality file owned by a specific user is charged by the quotas subsystem against that specific user, space consumption of a Windows personality file owned by a group is charged by the quotas subsystem against the “UNIX user owner” which, as described above, is “admin” (UID 1).

UNIX personality files are owned by an owning user and an owning group. The owning user of a file always has implicit permission to change permissions on the file (members of the owning group do not). Unlike the Windows security paradigm, permissions assigned to the owning user and group have special significance, due to the POSIX definition of the UNIX security paradigm:

- If the accessing user is the owning user of a file, only the permission assigned to the owning user is applied as the user’s permission
- If the accessing user is not the owner, but is a member of the owning group, only the permission assigned to the owning group is applied as the user’s permission
- If the accessing user is neither the owner nor a member of the owning group, the permission assigned to “other” is applied as the user’s permission

Since UNIX personality files always have an owning user, space consumption of UNIX personality files is always charged to that user.

The owning user of UNIX personality files is reported to Windows users as the owner.

Permission Inheritance on GuardianOS

As described earlier in the “Windows Inheritance” section, inheritance refers to the process by which child files and folders obtain permissions from parent folders. Inheritance for Windows files has been integrated since GuardianOS 2.5. However, prior to GuardianOS 5.0 there remained some missing pieces. With GuardianOS 5.0, all Windows inheritance rules are natively implemented and enforced for Windows personality files and folders created by Windows clients. Additionally, the ACE setting that prevents propagation of that ACE beyond the current folder level (“Apply these permissions to objects and/or containers within this container only”) is also respected.

Noted earlier in the discussion of inheritance, a file or folder created by a Windows client in a folder that has no inheritable ACEs will give the creator user Full Control on that file or folder. This occurs naturally even on a Windows system with NTFS when the folder in which the file is created has inheritable permissions set for “This folder only”. In GuardianOS, this can occur under one of two circumstances: when the directory in

which the file is created has a UNIX personality, or if the directory has a legacy POSIX ACE left over from a GuardianOS system that was upgraded to GuardianOS 5.0 from a prior version.

UNIX personality files and directories have no notion of inheritance. New files or directories created by UNIX clients will acquire a UNIX permission based on the umask of the client, as set by the client or client application.

Context-Sensitive Permissions

Releases prior to GuardianOS 5.0 had special issues with Windows files related to inheritance and context sensitive permissions; specifically, those applied to the owner and owning group. These were the result of the former GuardianOS POSIX ACL implementation used to emulate Windows ACLs. Since Windows permissions in GuardianOS 5.0 are natively supported in the Windows personality, these issues no longer exist. Permissions assigned explicitly to the owner or owning group of a Windows personality file or folder have no more significance to a user accessing the file or folder than other permissions that apply to the same user.

Permissions on UNIX personality files, however, are completely context-sensitive, following the POSIX security paradigm. (See the “Owning Users and Groups on GuardianOS” section above for details.)

GuardianOS Permissions Processing

With the introduction of GuardianOS 5.0, the processing of all permissions for files and folders with a Windows personality follows all of the rules of a Windows system using NTFS. Prior to v5.0, GuardianOS made concessions for compatibility; translating many of the permissions as well as processing permissions differently than Windows. The explanation of how Windows processes permissions above, including the processing for Deny permissions, is now accurate for GuardianOS, as well.

Permissions on UNIX personality files are enforced through the traditional POSIX security paradigm.

Special Groups for Windows Personality Files

Certain “Built-in” Windows groups are granted special privileges above and beyond those of normal users. Understanding these groups and their privileges is important to properly managing permissions in a Windows environment. Some of the notable groups include:

- **Administrators:** The special builtin Windows group “Administrators” has the special privilege to ALWAYS change ownership of a file or folder, including the ability to take ownership and arbitrarily assign ownership to any user or group. In the event that permissions make files or folders inaccessible, the members of the “Administrators” group can at least set the ownership of a file or folder, so the permissions can be set properly. Members of the Administrators group include local users who are members of admingrp and Windows Domain users who are members of the Domain Admins group of the domain to which the Snap Server belongs.
- **Owner:** As mentioned earlier, the owner (or a member of the owning group, if owned by a group) can ALWAYS change permissions of a Windows file or folder. This allows the owner (or owning Group) to set permissions properly, in the event that files or folders become inaccessible.
- **Users:** Though the Builtin group “Users” does not have any special abilities per se, it does have an important distinction in that it is equivalent to the frequently used group, “Everyone”, with one exception; it does not include the anonymous “Guest”

user. This makes it a better option when setting up permissions than the “Everyone” group, which most administrators have become accustomed to using. This way, only authenticated users within a particular Windows environment can gain access and anonymous “Guest” users can be kept out.

- CREATOR OWNER/CREATOR GROUP: These special Built-ins represent permissions to be translated to explicit ACEs to owners on a Windows file or folder. If an access ACE for either of these is assigned to a file or folder, the Windows client automatically translates it to an explicit ACE for the owner or owning group. If an inheritable ACE is assigned to a folder for either of these, new files and folders created inside this folder by Windows clients will inherit the specified permission as an explicit ACE for the owner or group.

Share-Level Access on GuardianOS

GuardianOS has features to accommodate the different methods used by the CIFS and NFS protocols for sharing data. Recall that Windows offers three levels of share-level permissions: Full Control, Change, and Read. In GuardianOS, administrators can assign “Full Access” or “Read Only” share access to individual Windows (and local) users and groups. In GuardianOS, “Full Access” is equivalent to “Change” for a share created on a Windows system. Since share permissions can only be controlled via the Web Browser Administration interface of GuardianOS, there is really no concept of “Full Control” that allows for the modification of share-level permissions via a Windows client’s native UI the way there is in Windows. Any administrative user with access to the GuardianOS Web Browser Administration interface can modify share-level permissions regardless of that administrative user’s share-level permission.

By default, the local group AllUsers has Read/Write access to the top-level share. This group is equivalent to the “Everyone” group in Windows and “Other” in UNIX. It is important to recognize that this is only the first step in securing a GuardianOS-powered Snap Server, as it only specifies access at the individual share level. All other access control must be handled through either a UNIX or Windows client, depending on which access method is trying to be controlled.

GuardianOS Administrators also have the ability to hide a share from view of all protocols by selecting the Hidden checkbox in the Share Access screen of the Web Administration Tool. When a share is “Hidden”, the share can not be “ browsed” from a Windows client and must be explicitly called out using the UNC path to gain access. This has a similar effect as adding a dollar sign (\$) to the end of a share name on a Windows system. The dollar sign can be used in the same way on GuardianOS shares and it will have the same effect for Windows clients accessing the share. Only the “Hidden” option will hide shares for all file access protocols.

Note: It is important to note that any new user or group that is given access to a share is given “Full Access” to the share by default.

There are occasions where share access must be controlled for UNIX clients. In these instances, Administrators can control how UNIX clients access the shares via the Web Browser Administration Tool. Modifications are performed on a per-share basis and are accessed by clicking the NFS Access button in the Share Access screen. The changes made in this screen affect the NFS “exports” file within GuardianOS. The “exports” file cannot be manually modified, since GuardianOS controls the precise state of all system files and will not retain any modifications made to this file. However, since the Web Browser Administration interface allows for free text entry for NFS exports, any standard Linux export option can be used. Specific information and details on the available export options are readily available with online resources.

Another important fact regarding share-level permissions on GuardianOS is that all permissions are cumulative. For example, if the user “jdoe” has Read-Only share access and belongs to the group “sales”, which happens to have Read/Write share access, the end result is that the user “jdoe” will have Read/Write access.

Best Practice: For purposes of simplicity of administration and security, it is highly recommended that shares NOT be created at the root of a volume on GuardianOS. For example, if the volume name is VOL0, do not create a share that points to VOL0. This may seem confusing since all GuardianOS systems are shipped by default with a share named SHARE1 that points to the root of VOL0. A share to the root of the volume should only be used by administrators as a “backdoor” into the rest of the directory structure in the event that permissions of a child directory are inadvertently altered by a user, thus disallowing administrative access to that directory structure. If it is necessary to have the root of the Volume accessible, using the Hidden option and restricting share access to administrative users is the best way to ensure that only those who absolutely need access to that share can easily get to it. Creating a root directory beneath the volume and creating a share to that root directory is the recommended hierarchy.

Diagram 2 illustrates an example of this recommendation:

Volume and Directory Hierarchy	Shares
VOL0	(H) SHARE1 (Hidden)
- home_dir	(None defined)
- jdoe	jdoe (Virtual share)
- bsmith	bsmith (Virtual share)
- Departments	Dept
- Applications	Apps

DOS Attributes on GuardianOS

Snap Server has had the ability to support DOS Attributes, just as any Windows operating system would, since GuardianOS 3.0 . GuardianOS stores the following DOS Attributes in file system Metadata called Extended Attributes (EA's):

- Read Only
- Hidden
- System
- Archive

These are not permissions; they are indicators that the file is in a certain state. For example, “Read Only” does not really prevent write access to a file. However, because of this attribute, some applications open the file as Read Only, even though the user account accessing the file may actually possess the appropriate permissions to modify the file.

Another example has to do with backup applications. Many backup applications use the Archive bit to determine whether or not a file needs to be backed up. Whenever a file is changed, the Archive bit is set so this file can be recognized as changed by the backup application.

You may recall there are three other attributes specific to date and time that are considered DOS attributes. The time stamp attributes of Create Date, Modify Date and Access Date are seen on all files and folders maintained within GuardianOS with one important caveat. The Create Date is always mapped to the Modify Date;

therefore there is no way to accurately know when a file or folder was originally created within GuardianOS.

Since GuardianOS does not use the Windows NTFS file system, it does not support the more advanced Windows NTFS Attributes of Compression or Encryption.

Group Policy Objects on GuardianOS

As of this writing there is no support for GPOs in GuardianOS. Windows Domain policies will not affect a GuardianOS-powered Snap Server and there is currently no concept of local GPOs in GuardianOS. However, the default state of some behaviors governed by GPOs are emulated by GuardianOS, including:

- Members of "Administrators" have the implicit Take/Grant Ownership privilege on any Windows personality file or folder. This is normally governed in Windows by the "Restore Files and Folders" GPO, which is granted to Administrators by default.
- All users have the "Traverse Folder" permission. This is normally governed in Windows by the "Bypass Traverse Checking" GPO, which is granted to Everyone by default (note that Microsoft strongly discourages changing this behavior, and notes that many subsystems and applications will not function properly if changed).

SnapTree Security Model

Directories created on the root of a GuardianOS volume are assigned one of two security models: Windows/Mixed or UNIX. The security model determines the rules regarding which security personality will be present on files and directories created by the various protocols and clients, and whether the personality of files and directories can be changed by changing permissions. These directories are referred to as SnapTrees.

Note: The security model of a SnapTree directory may differ from the personality of the directory. For example, files or directories in a Windows/mixed SnapTree may have a UNIX personality and vice-versa.

The advantage of SnapTrees is that an administrator can now make a specific directory structure behave most like its native environment without having to have one specific storage hardware platform for UNIX and one specific hardware platform for Windows.

Windows/Mixed SnapTrees are more flexible than UNIX SnapTrees. Files and directories in a Windows/Mixed SnapTree can be set to either a Windows or a UNIX personality. In a UNIX SnapTree, all files and directories have a UNIX personality.

Creating a directory structure that only adheres to UNIX permissions can be particularly useful in environments where UNIX clients control the creation and modification of data, and Windows clients have access to the data. This type of directory structure avoids the potential problems of Windows clients resetting permissions while accessing the data, since Windows clients are not able to modify permissions on UNIX SnapTrees. The security model of a SnapTree (or of an entire volume) can be changed at any time in the *Security > SnapTrees* screen of the GuardianOS Web Browser Administration Tool. When doing so, the personality of the SnapTree directory is changed to match the new security model (Windows for "Windows/Mixed", and UNIX for "UNIX"), and a default permission is set. Also, the new security model can be propagated to all sub-files and subdirectories underneath the SnapTree, in which case their personalities will be set to match the security model and a default permission is set (*note that this has the effect of resetting all permissions on an entire SnapTree to defaults*).

SnapTree Functionality

The default SnapTree directory ownership differs according to the method used to create the SnapTree directory. Table 9 describes the interaction between SnapTrees and security models.

Table 9 – Behavior of SnapTrees and Security Models

Function	Description
SnapTree Directory Ownership	<p>Default ownership differs according to the method used to create the SnapTree directory:</p> <ul style="list-style-type: none"> • <u>From the client</u> — For UNIX personality directories, the owner and owning group will be according to the logged-in user. For Windows personality directories, the owner will be the logged-in user, or "Administrators" for directories created by Domain Admins or members of the local admingrp. • <u>From the Administration Tool</u> — For UNIX personality directories, the user and group owner will be admin and admingrp. For Windows personality directories, the owner will be the local admingrp ("Administrators").
Security Personality of Files and Directories	<p>Files and directories created by clients inside SnapTrees will acquire security personality and permissions according to the rules of the SnapTree security model.</p> <p><u>Windows/Mixed SnapTree:</u></p> <ul style="list-style-type: none"> • Files and directories created by SMB clients will have the Windows security personality. Permissions will either be inherited according to the ACL of the parent directory (if Windows) or will receive a default ACL that grants the user full access (only if the parent is UNIX or has no inheritable permissions). • Files and directories created by non-SMB clients will have the UNIX personality. UNIX permissions will be as set by the client (per the user's local umask on the client). • The security personality of a file or directory can be changed by any user with sufficient rights to change permissions or ownership. If a client of one security personality changes permissions or ownership of a file or directory of a different personality, the personality will change to match the personality of the client protocol (e.g.: if an NFS client changes UNIX permissions on a Windows file, the file will change to the UNIX personality). <p><u>UNIX SnapTree:</u></p> <ul style="list-style-type: none"> • Files and directories created by non-SMB clients will have the UNIX personality. UNIX permissions will be as set by the client (per the user's local umask on the client). • Files and directories created by SMB clients will have the UNIX personality. UNIX permissions will be set to a default. • The personality of files and directories cannot be changed on a UNIX SnapTree. All files and directories always have the UNIX personality.
SnapTree File System Permissions	<p>Security model and permissions differ according to the method used to create the SnapTree directory:</p> <ul style="list-style-type: none"> • From the client: If SMB, permissions will either be according to ACL inheritance (if the parent volume root directory has the Windows security model) or Full Access to the owning user only. Permissions for directories created by all other protocols will be set by the client (per the client's umask). • From the Administration Tool: If created in a UNIX volume, permissions will be 777 (rwxrwxrwx). If created in a Windows/Mixed volume, permissions will allow all users to create, delete, and change permissions on files created inside the SnapTree, and will grant full control to Administrators.
Toggle Security Models	<p>Changes to a SnapTree's security model can optionally be propagated to the corresponding personality with a default permission to all files and directories underneath the SnapTree.</p> <p>When changing the security model on a SnapTree:</p> <ul style="list-style-type: none"> • If changing from Windows to UNIX, all files and directories will be changed to be owned by admin and admingrp, with UNIX permissions of 777 (rwxrwxrwx). • If changing from UNIX to Windows, files and directories will be changed to default permissions that allow all users the ability to create and manage their own files and directories and to access other users' files and directories.
Mixing SnapTrees	You can create SnapTrees of different security models on the same volume.

It is important to note that, as suggested above, directories created in the root of a volume (which by definition are SnapTrees) inherit the security model of the volume root directory by default. In other words, a volume is assigned a security model, and thus new directories created at the root of the volume default to that security model, even if the personality of the SnapTree directory is different than the security model. For example, a SnapTree created by a UNIX client inside the root of a Windows/Mixed volume will have the Windows/Mixed security model, but will have the UNIX personality. Administrators can also create a mixed SnapTree model beneath any volume. As mentioned before, this can all be done in the *Security > SnapTrees* screens. The result is that administrators have the ability to have both Windows and UNIX SnapTrees on the same volume.

Table 10 – Example of what a mixed SnapTree structure might look like

Name	SnapTree Security Model
VOL0	Windows
- home_dir	Windows
- Finance	Windows
- Engineering	UNIX

It is important to note that SnapTrees can only be configured on the root of a volume or the immediate subdirectories off of the root. The SnapTree type for all other subordinate subdirectories cannot be modified and will inherit the SnapTree type from the parent directory.

UNIX Features in Windows

Symbolic links are not a feature of Windows. Some file systems do have the capability to translate UNIX symbolic links for Windows access; however, GuardianOS does not support the traversal of UNIX symbolic links via a Windows client. However, hard links created on GuardianOS via UNIX are maintained and accessible from a Windows client.

ID Mapping in GuardianOS

It is sometimes important, if not essential, in a mixed access environment to be able to provide an individual user both Windows and UNIX client access to his or her data. Within GuardianOS this can be done by ensuring that the UID and GID used from the UNIX client are synchronized with the corresponding Windows Domain user and group accounts.

It is typical in most mixed environments that there is not only a Windows Domain or Active Directory for handling user accounts on the Windows side, but also a NIS Server to handle the accounts used in UNIX. Reconciling these two unique authentication architectures to allow for common access to data can be difficult.

Within GuardianOS, administrators can map local or NIS user and group IDs to Windows Domain users and groups using the ID Mapping feature. ID Mapping allows users and groups that exist on Windows Domains to share user IDs with local or NIS users and groups. This results in the same permissions and quota consumption applying to both the Windows Domain user and the local or NIS user. This mapping functionality allows for a cohesive permissions access model across protocols.

Default UID and GID Assignments in GuardianOS

The GuardianOS-powered Snap Server uses the POSIX standard to assign a UID and GID, in which each user and group must have an ID. This requirement applies to all users and groups on the Snap Server, including local, Windows, and NIS users and

groups. If you join the Snap Server to a Windows or NIS Domain, IDs are automatically assigned. UIDs and GIDs are now assigned on a "first come, first served" basis. This means that whichever authentication service claims the UID/GID first gets to use the ID for the user or group. Consider the following table when creating users and groups.

Table 11 – GuardianOS UID and GID Ranges by User Type

User Type	Range
Reserved	0 – 100
NIS and Local	101 – 17999
nobody	65534 – The NIS user 'nobody' is reserved. It cannot be mapped to another ID, nor can another ID be mapped to 'nobody'.
Local Only	Starts at 18000
Windows	Starts at 30000

If joining GuardianOS to a Windows Domain, UIDs are automatically assigned. If joining GuardianOS to a NIS Domain, consider the following guidelines: (1) GuardianOS does not recognize users or groups whose identification numbers are outside the range indicated in the table; and (2) each UID or GID must be unique.

The combined limit for total number of UIDs and GIDs on the Snap Server is 65,435 UIDs and 65,436 GIDs. The breakdown of the number of users by type is in the table below.

Table 12 – GuardianOS User Maximums by User Type

User Type	Maximum Users
Local	2,000
Windows Domain	46,535
NIS	16,900
Total	65,435

Getting a list of All Users and Groups on GuardianOS

Within the Command Line Interface (CLI), which is available utilizing SSH, an administrator has the ability to obtain a list of the users and groups. All commands and syntax are available once logged into to the Snap Server via SSH by typing the word 'help' – either by itself, or after any command.

Obtaining a list of users or groups requires the command 'user' or 'group' along with the 'list' option. Examples of commands to obtain a list of Local, Windows and NIS users or groups are in the table below.

Table 13 – CLI command examples for listing users and groups on a Snap Server

User Type	CLI Command Example to List
Local Users	<code>user list</code>
Windows Domain Users	<code>user list domain-name=MyWinDom domain-type=windows</code>
NIS Users	<code>user list domain-name=MyNISDom domain-type-nis</code>
Local Groups	<code>group list</code>
Windows Domain Groups	<code>group list domain-name=MyWinDom domain-type=windows</code>
NIS Groups	<code>group list domain-name=MyNISDom domain-type-nis</code>

Data Migration

One of the most important aspects of security has to do with migrating files and directories from one file sharing platform to a Snap Server. Once a security scheme is planned and established, it will be time to migrate the data, unless the storage will be for new data.

In a mixed environment that has existing UNIX and Windows file sharing, there is a specific order for migrating data to ensure the best preservation of permissions. This document will not attempt to describe all details of these steps, but instead is a guide to help better plan for and test the migration process.

Since GuardianOS contains an embedded Data Migration tool, we will focus on utilizing this utility over other options. This paper will only provide an overview of how to use the Data Migration tool. Other documentation is available that dives deeper into the details of the Data Migration tool. Alternative methods will be listed where appropriate.

Step 1: Ensure proper Windows Domain and NIS integration

If the data will be accessed by both Windows and UNIX clients, make sure to map IDs utilizing the ID Mapping feature described above.

In order to ensure proper data migration of permissions, it is essential that the Snap Server join the Windows Domain and/or NIS Domain prior to proceeding any further. This is required because GuardianOS needs to know the Windows SIDs and any UIDs/GIDs in order to preserve permissions properly during migration.

Step 2: Migrate UNIX client data

Once all UNIX UIDs and GIDs are added to GuardianOS as Local users and groups, or integrated to a NIS Domain on the new Snap Server, the UNIX data is ready to be migrated. The Data Migration tool available on the [Maintenance > Data Migration](#) screen in the Web Browser Administration Tool can be used to copy or move the data via the NFS protocol.

When performing an NFS data migration job, be sure to specify a user that can access all source data via NFS. This is usually root. In some cases, however, the root user will not have root privileges due to presence of the "root_squash" (or lack of "no_root_squash") option. This should be changed to temporarily allow root full access to the files for the duration of the migration process.

Step 3: Migrate Windows client data

Once the Snap Server is integrated properly with the appropriate Windows Domain, the Windows client data can be migrated. Similar to migrating the UNIX data, the Data Migration tool located on the [Maintenance > Data Migration](#) screen in the Web Browser Administration Tool can be used to copy or move the data via the CIFS protocol. If the files already exist because they have been migrated for UNIX client access, then the only updates will be the Windows ACLs.

Note: permissions for users and groups that cannot be recognized by the Snap Server—including local users and unknown domain users on the source machine—will be retained verbatim with the appropriate SID, but will not be enforced.

When performing Windows data migration, be sure to specify an administrative username that can properly access all the source data over the network for data migration.

Alternative Migration Methods

If it is desirable to migrate data using methods other than the included Data Migration tool, existing tools for both UNIX and Windows can be used to migrate client data.

For UNIX client data, it is valid to utilize the UNIX “tar” command, while taking care to maintain all permissions while migrating data to the Snap Server. The “cp” command with the “—preserve” option can also be used.

For Windows client data, the built-in Windows “xcopy” command with the “/o” option run from a command line will also allow all permissions to be copied over. Please refer to the “xcopy” help for more details on the use of this command. Note that permissions for users and groups that cannot be recognized by the Snap Server—including local users and unknown domain users on the source machine—will be retained verbatim with the appropriate SID, but will not be enforced.

Remember to log in with a username that has access to all the source data via the network. This is often a member of Domain Admins, but can differ for each environment (*note: in the Windows security paradigm, there is no user who always has implicit access to file data*).

Note: some very complicated permissions hierarchies may give errors when the user logged in and attempting to migrate the data has not been given proper access to copy the data. It may require that Read, Write and Execute permissions be granted to the administrative user logged in to successfully copy the Windows permissions. If these permissions are not granted, the user performing the “xcopy” will be presented with “Access Denied” to portions of the folder structure, due to the inability to Read and Write to a folder.

There are other Windows-based copy utilities available, all based on the “xcopy” command, but for the best results the “xcopy” command should be used.

As with any data migration careful planning and testing is essential.

Upgrading From a GuardianOS Version Prior to 5.0

Since the new Windows security personality was introduced in GuardianOS 5.0, many existing customers are anxious and excited to upgrade to GuardianOS 5.0, to take advantage of the new security paradigm. GuardianOS 5.0 makes administration in a Windows environment 100% compatible with a typical Windows NT and Active Directory environment.

In previous versions of GuardianOS, a POSIX ACL scheme was the underlying infrastructure for all access control on a Snap Server. That led to scenarios where a Windows client would read and enforce the permissions from files on a Snap Server; then if rewritten, the Snap Server would translate the permissions that the Windows client attempted to write. This rarely matched the administrator’s exact expectations. With GuardianOS 5.0, these inconsistencies no longer exist.

However, it is important to point out that even though upgrading to GuardianOS 5.0 will provide administrators with all the new features, pre-existing, “legacy” files will not necessarily behave as expected in all situations. Files and folders are not automatically updated from the previous POSIX ACL with the new native Windows

personality upon upgrade, and will continue to be enforced identically to the pre-5.0 implementation. However, there are a couple of ways that files will receive new ACL security information.

One way is due to the fact that some applications running on Windows have the notion of deleting and rewriting a file when an end user saves a new version of a file. In this case, the new security model will be utilized any time a file is saved with that application. If, however, the application simply appends to the file, the old POSIX ACL will remain.

The second way to accomplish this is for the administrator to use the native Windows tools for setting permissions on all of the files and folders. By opening the “Properties” dialogue and selecting the “Security” tab to verify that permissions and inheritance are setup properly, then saving the permissions, all ACLs will be written with the new native ACL permissions scheme. This method is useful when an administrator neither wants to wait until every file and folder is processed by an end user, nor wants to count on the uncertain event that saving an existing file will actually update the ACL information.

Of course, it is not absolutely essential to force all files and folders to employ the new security paradigm. If you were happy with how your Snap Server was handling permissions in the past, just move forward with the assurance that over time your files will be updated to use the Native Windows ACLs; and that access will not be compromised for files that are not updated.

Conclusion

With GuardianOS 5.0, all permissions, regardless of the security paradigm being used, are handled in the expected way for that protocol. For all supported file access protocols, permissions are retained natively and accurately, without translation to approximate representations as in typical Linux/NAS/Samba implementations. Permission handling for clients follows their native security paradigm – which means that all permissions attributes are preserved in the GuardianOS file system and returned in their native state, unaltered, and as requested.

In particular, Windows environments can benefit tremendously from the new native Windows file system security implementation, which makes permissions on GuardianOS-powered Snap Servers 100% compatible with Microsoft Windows, thereby working the way administrators of Windows permissions and security schemes would expect.

The other benefit gained with the new GuardianOS 5.0 security paradigm is that administrators will be able to take advantage of the fact that all permissions generated on files with a Windows personality will be enforced – even on clients that would not normally recognize those permissions (e.g.: UNIX and Mac OS X)—thus enabling standard Microsoft file access enforcement for all file access protocols and creating a “universal ACL” that guarantees consistent and predictable cross-platform file access.

About Overland Storage

Overland Storage provides affordable end-to-end data protection solutions that are engineered to store smarter, protect faster and extend anywhere — across networked storage, media types, and multi-site environments.

Overland Storage products include award-winning NEO SERIES® and ARCVault™ tape libraries, REO SERIES® disk-based backup and recovery appliances with VTL capabilities, Snap Server® NAS appliances, and ULTAMUS™ RAID high-performance, high-density storage. For more information, visit www.overlandstorage.com.

WORLDWIDE HEADQUARTERS

4820 Overland Avenue
San Diego, CA 92123 USA
TEL 1-800-729-8725
1-858-571-5555
FAX 1-858-571-3664

UNITED KINGDOM (EMEA OFFICE)

Overland House, Ashville Way
Wokingham, Berkshire
RG41 2PL England
TEL +44 (0) 118-9898000
FAX +44 (0) 118-9891897

