GroupLogic®

ExtremeZ-IP

Security Overview

ACLs and Unix Permissions

Version: 1.3
Date: 12/31/10
Author: Gene Furtman

Group Logic, Inc.
1100 N. Glebe Road, Suite 800
Arlington, VA 22201
(703) 528-1555
http://www.grouplogic.com
support@grouplogic.com

Table of Contents

1 ExtremeZ-IP UNIX Permissions and Security Settings 4
2 UNIX permissions in the Windows Access Control List (ACL) 4
3 Registry settings affecting security 7
3.1 General Behavior 8
3.1.1 Allow Mac clients to change permissions 8
3.1.2 Reset permissions on move 8
3.1.3 Support UNIX permissions and ACLs 10
3.1.4 Support ACLs on all volumes/Volume supports ACLs 12
3.2 Interaction between security feature settings in ExtremeZ-IP 13
3.3 Special Registry Keys for permission issues 13
3.3.1 Using UNIX permissions mode keys 14
3.3.2 Using the default umask and permissions keys 15
3.3.3 Important notes about this feature: 17
4 General Security notes 17
4.1 How to set up security 17
4.2 Information to gather before handing off case to dev 18
5 General Security Questions 19
5.1 How inherit-only ACEs work 20
6 GLOSSARY: 21

Revisions:

Name	Date	Version Number	Change Made
Gene Furtman	07/23/2009	1.0	Created
Josh Townsend	08/03/2009	1.1	Added comments and glossary
Gene Furtman	09/25/2009	1.2	Updated for 6.0.3

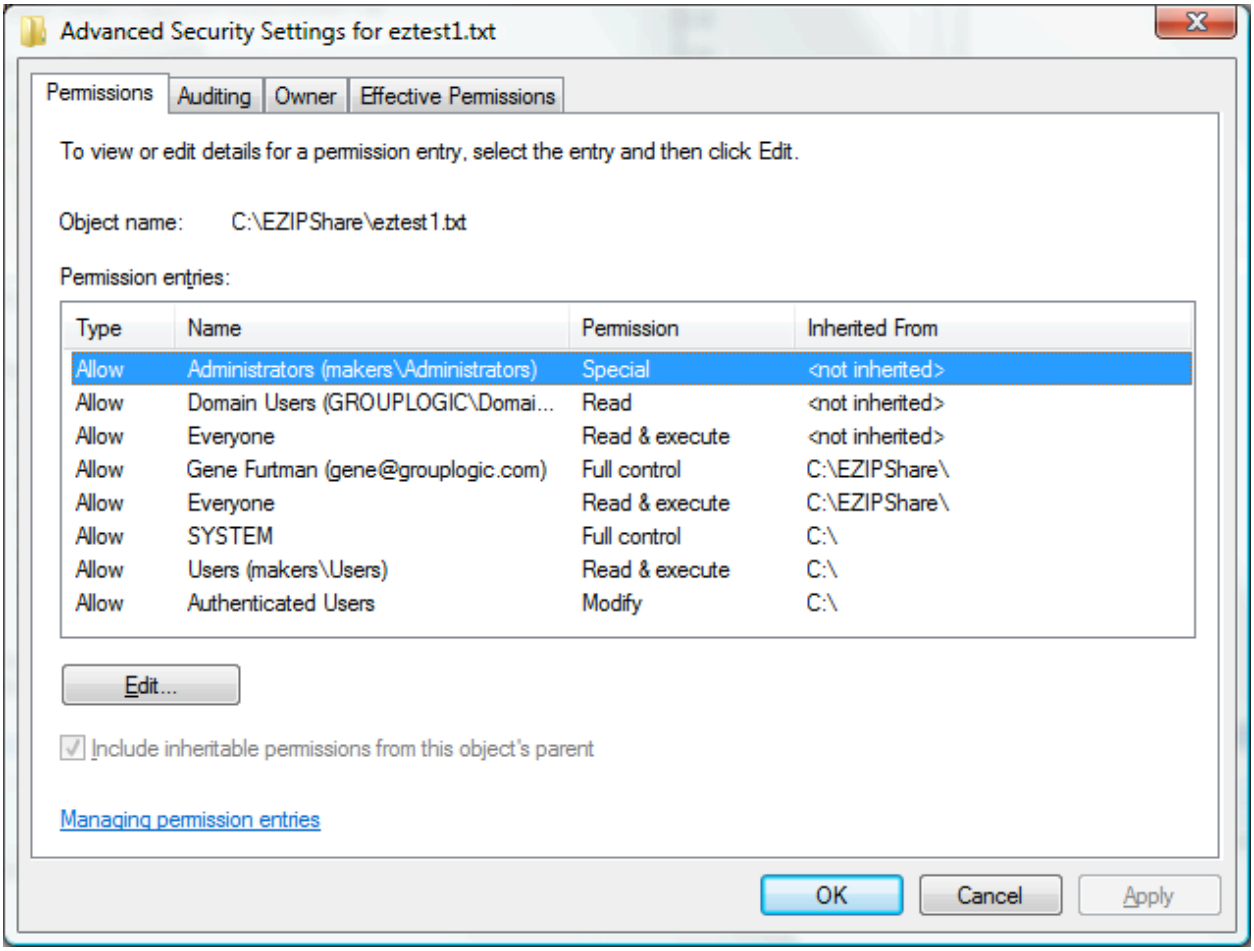
1 ExtremeZ-IP UNIX Permissions and Security Settings

It is assumed that the reader has a basic knowledge of security principles and the terminology used in Windows and Mac security models. A brief glossary is provided at the end of this document.

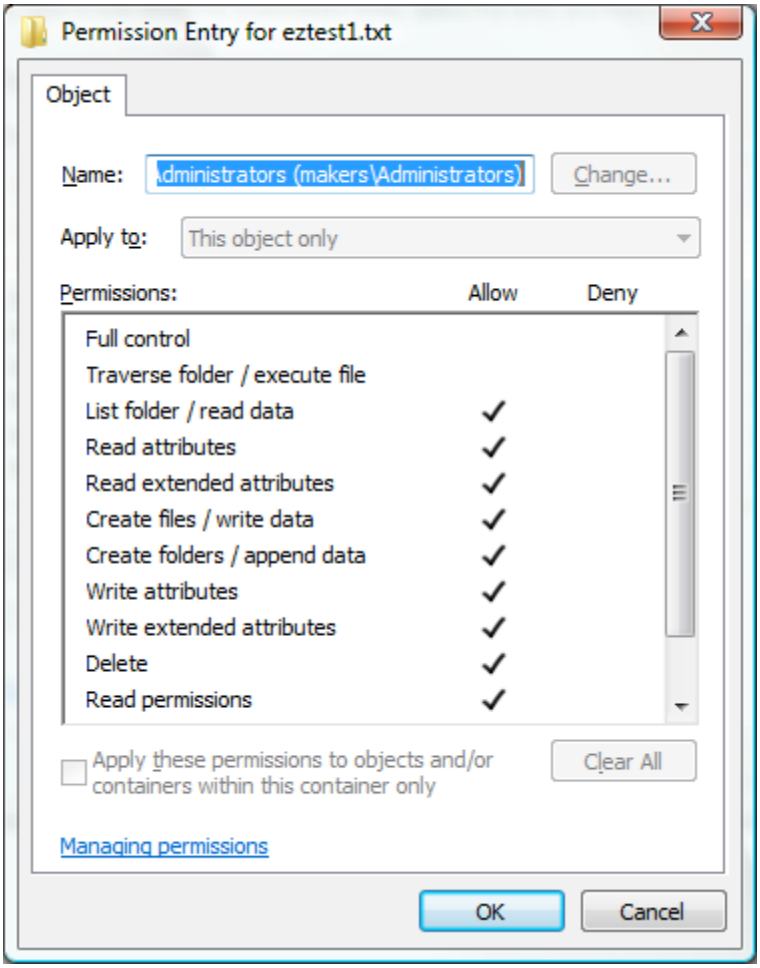
2 UNIX permissions in the Windows Access Control List (ACL)

Access Control Entries (ACEs) in the ACL come in two varieties, those inherited from their parent object(s) and those that are explicitly added to the object's security. Windows users and administrators can modify an ACL any time provided they have sufficient privileges to do so. Mac users can also modify ACLs but it is considerably more difficult.

In order to provide UNIX permissions for each permission security class (owner, group, and other or world), ExtremeZ-IP adds explicit ACEs to the ACL based on the UNIX permission settings requested by the Mac. Therefore, if UNIX permissions are supported in ExtremeZ-IP and the Mac is allowed to change permissions, an ACL will typically be a mix of inherited ACEs, explicit ACEs added by Windows users, and explicit ACEs added by Mac clients to provide UNIX permissions.



Note, in the above Advanced Security Settings Dialog display, that there are three ACEs that are '<not inherited>'. These are the explicit ACEs that ExtremeZ-IP added when the Mac set the UNIX permissions. The first explicit ACE is for 'Administrators'. Because the file was created by a member of the 'Administrators' group, the file is owned by 'Administrators'. The permissions are shown as 'Special'. If you double-click the ACE, you will see the complete list of permissions for the user. In this case, the 'Administrators' ACE grants almost complete control. It is only lacking the 'Traverse folder / execute file' permission. When the owner permissions are set, we add the other permissions that the owner is entitled to such as read permissions, change permissions, delete, etc.



The next explicit ACE is for 'Domain Users'. This is typically the owner's primary group although the group can technically be any group that the owner is a member of. The group permissions in this case are only 'Read'.

The final explicit ACE is for 'Everyone'. This corresponds to the 'other' or 'world' UNIX permissions. In this case, 'Everyone' has read and execute.

If you were to view the UNIX permissions for this file, they would be set to 645….'owner' can read and write,' group' can read,' others' can read and execute.

If any segment of the UNIX permissions is zero, the corresponding explicit ACE will be missing. For the file above, if the permissions were 605, the explicit ACE for 'Domain Users' would not be present.

Keep in mind that there is no way for ExtremeZ-IP to distinguish between the explicit ACEs it adds and any other explicit ACEs that might be added by a Mac or Windows user. For example, if a file is created and UNIX permissions are set to 640, 'Everyone' would have no permissions and you would find no explicit ACE for the 'Everyone' group in the ACL. However, if a user with

sufficient permissions to modify the ACL were to add an ACE to the file for 'Everyone', because it would be an explicit ACE, we would interpret it as part of our UNIX permissions and return whatever permissions they had set for 'Everyone'.

When Support UNIX permissions and ACLs in the Settings->Security tab of the ExtremeZ-IP Administrator is enabled, ExtremeZ-IP first attempts to determine UNIX permission using the explicit ACEs in the ACL. If an explicit ACE is found for the owner, group, or everyone, then it is assumed that UNIX permissions have been set and the permissions are determined solely from the explicit ACEs. If there are no explicit ACEs found for the owner, group, or everyone, then the UNIX permissions are determined from the effective rights of the entire ACL for the owner, group, and everyone.

Regardless of the method used to determine UNIX permissions, the actual rights granted to the user are based on Windows' evaluation of the entire ACL. That is, if UNIX permissions are determined to be 740 based on explicit ACEs for the owner and group, and there was also an inherited ACE for 'Everyone' allowing read permission, any user would still be able to read the file based on the rights granted by the 'Everyone' ACE in the ACL.

Keep in mind that no matter the Mac may think the UNIX permissions are, the ultimate security for the object is based on the entire ACL as evaluated by Windows, not by the Mac. This can lead to some confusion for customers who believe that UNIX permissions of 770 should allow only owner and group access while the ACL may have an inherited ACE for 'everyone' that would also grant everyone some form of access. Conversely, the Mac is supposed to use UNIX permissions ONLY after determining that the ACL does NOT grant or deny access to the user. But there are conditions in both Leopard and Snow Leopard where the Mac will fail to attempt to access a file based solely on the UNIX permissions.

3 Registry settings affecting security

There are four settings that affect how security behaves (or appears to behave) on ExtremeZ-IP volumes. While in certain circumstances, the security may appear to be more liberal than desired when viewed on the Mac, the actual object's security will be based solely on the file's/folder's Windows security descriptor and the privileges of the user. For example, if ExtremeZ-IP receives an error when looking up the owner or group name in Active Directory because it is a local group, permissions of 'rwx' would be returned in the UNIX permission request from the Mac. The effect would be that the Mac would consider making further requests, perhaps to read the file. When the read request was processed by ExtremeZ-IP, the security token for the user would be used by Windows to determine if he or she actually had read access to the file. In releases prior to 6.0.2, ExtremeZ-IP would return no permissions in such a circumstance and the Mac would not even attempt to make the read request, which would then cause the user to not have access to a file or folder to which he or she actually had access.

The four settings for security are:

- Allow Mac clients to change folder permissions
- Reset permissions on move
- Support UNIX permissions and ACLs
- Support ACLS on all volumes

Reset permissions on move and Support ACLs have volume-specific counterparts which allow the feature to be enabled on a volume-by-volume basis if the global feature checkbox has not been selected.

3.1 General Behavior

3.1.1 Allow Mac clients to change permissions

This setting applies to both files and folders now that Mac and ExtremeZ-IP support modifying file security.

Feature not checked (disabled):

If this setting is unchecked, no security changes will be applied as a result of requests from Mac clients. This includes changes explicitly made by users via 'Get Info' or Terminal using chmod/chown commands as well as those implicitly attempted by OS X when a file is created, moved, copied, etc. Since the security will be applied according to Windows security rules for creating, moving, and copying files, it is possible that the Mac client could be left without access to needed files. When this setting is not checked, the ExtremeZ-IP server uses the Windows-based security model. That is, behavior for creating, copying, moving, and deleting objects is the default Windows security behavior. No ACEs are added or removed from the ACL as a result of ExtremeZ-IP processing. The ACL should look exactly as it would if a Windows user had created, moved, or copied the file/folder.

Feature checked (enabled):

If this setting is checked, then the UNIX permissions and/or ACLs can be modified by 3rd party software like Photoshop, user commands in terminal, and the Finder on the Mac client.

3.1.2 Reset permissions on move

There is both a global setting and a volume-specific setting for this feature. If the global setting checkbox is checked, it overrides any volume-specific setting. This applies only to moves within

a volume as moves to another volume occur as a copy then delete pair of operations. This feature is not affected by the setting of Allow Mac clients to change permissions.

Feature not checked (disabled):

When unchecked, the security descriptor of the file or folder will not be modified when it is moved. That includes the owner, group, and the ACL. This is default behavior for both Mac OS X, and Windows. The net result is that the file has exactly the same security it had before it was moved. For example, if the file was moved from a location that did not grant access to a specific group, let's say 'Graphics Dept', to a folder where 'Graphics Dept' had full control, the file would still not be accessible to 'Graphics Dept' because the inheritable security from its new location has not been applied.

Featured checked (enabled):

When checked, the result of a file or folder move on security is essentially the same as if the 'MoveSecurityAttributes' feature of Windows was disabled. Typically, when a file is moved within a volume, the security descriptor goes along with the data without modification. That is, no inheritance is applied from its new parent and the owner remains the same. However, when the 'MoveSecurityAttributes' feature of Windows is disabled, all explicit ACEs in the ACL are removed, all inherited ACEs from the source are removed, and the inheritance from its new parent is applied. The owner is not changed.

With one exception, ExtremeZ-IP behaves the same way. ExtremeZ-IP will remove all existing ACEs in the ACL and apply the inheritance from the new parent. It will then change the object's owner to the user moving the file, which is different than the expected Windows behavior. However, no additional ACE to allow access to the object will be added. As a result, the file may no longer be usable by the user that moved it even though they become the owner. Windows does not convey any read or write permissions for the owner by default to a file or folder as UNIX does. Group Logic's position is to not modify the ACL from what it would be had Windows performed this action itself.

If there is a CREATOR_OWNER ACE in the parent folder, an ACE for the new owner will be added when the permissions are reset.

As in our example above, if this feature is checked and a file is moved from a location where 'Graphics Dept' did not have access to a folder where 'Graphics Dept' did have access, members of the 'Graphics Dept' would gain access because we would replace the original security information with the inheritable security from its new parent folder.

3.1.3 Support UNIX permissions and ACLs

This is a global setting and affects ExtremeZ-IP's ability to interact with the UNIX permissions and ACL features of OS X.

UNIX permission support and ACL support are separate but related. UNIX permissions can be enabled without enabling ACL support. This simply means that Mac requests to read or change an ACL will not be supported by ExtremeZ-IP. However, ACL support requires UNIX permissions to be enabled.

Feature not checked (disabled):

If UNIX permissions and ACLs are disabled, the Mac will not request nor will it update the UNIX permissions for any file or folder on the ExtremeZ-IP server. UNIX permissions will not be set on files and folders that are copied to the ExtremeZ-IP server from a Mac client. The Mac client will set UNIX permissions on files and folders it copies from ExtremeZ-IP to the local machine, i.e. to the desktop, based on the UNIX settings of the client machine. In that case, the UNIX permissions are determined by the owner, the owner's primary group and the umask.

ExtremeZ-IP reverts back to the use of Mac Access Rights, which only exist at the folder level. This does not affect the way file security is enforced at the server as all access to files and folders is controlled by Windows security, i.e. the ACL.

ExtremeZ-IP security behavior reverts mostly to the way it was prior to release 5.2 when UNIX permissions and ACLs were first supported with a few exceptions. For example, we now keep the correct security descriptor with the correct data when an ExchangeFile is executed. We also correct the inheritance flags in an ACL whenever a file or folder is created.

Feature checked (enabled):

Requests by the client to set UNIX permissions result in the following actions:

- an explicit ACE will be added to the ACL granting the owner the permissions requested by the Mac
- an explicit ACE will also be added to the ACL granting the owner's primary group the permissions requested by the Mac
- an explicit ACE will be added to the ACL granting the Everyone group the permissions requested by the Mac

For example, if the Mac sent a request to ExtremeZ-IP to set UNIX permissions to 764, an ACE would be added to the ACL for the owner granting them full access; an ACE would be added to the ACL for the primary group (usually Domain Users) granting read and write access; and an ACE would be added to the ACL for Everyone granting read-only access. No ACE is added if the relevant portion of the UNIX permissions is zero. Therefore, if the UNIX permissions being set were 704, no ACE for the owner's primary group would be added, and, if an explicit ACE for the group existed in the ACL, it would be removed.

Requests by the Mac to read UNIX permissions results in ExtremeZ-IP returning UNIX permission information determined in one of several ways.

Determining UNIX permissions for files and folders created by Mac clients:
Since there will be an explicit ACE in the ACL for the owner, group (perhaps), and 'Everyone' (perhaps), ExtremeZ-IP simply gets the rights from the ACE for the owner, the ACE for the group, and the ACE for Everyone, converts the rights to the standard UNIX permissions and returns the result to the client. If the client did not set 'group' or 'others' permissions (for example, permissions of 700) then no explicit ACE for group or 'Everyone' will be in the ACL.

Determining UNIX permissions for files and folders created by Windows clients:
Files and folders that are created on the server by Windows clients will not have any explicit UNIX permission ACEs in their ACL. It is therefore necessary for ExtremeZ-IP to calculate the UNIX permissions based on the content of the object's security descriptor.

All files and folders have security descriptors, which include an owner SID, group SID, and ACL. Windows assigns the owner and primary group at the time the file is created. The owner is either the user that created the file or the 'Administrators' group if the creator is a member of the 'Administrators' group. This latter behavior can be modified in Windows 2003 by changing the default for the Local Security Policy->Security Options->System Objects: Default owner for objects created by members of the 'Administrators' group to 'Owner' rather than the 'Administrators group'.

The primary group assigned to the file is the user's primary group. Typically this is Domain Users. However, if the owner is a local account on the server, the group will be <local machine name>\None. This user will typically have no group permissions when UNIX permissions are calculated. The user's primary group can be changed in AD using the 'Member of' tab.

ExtremeZ-IP calculates the UNIX permissions by getting the effective rights for the owner SID from the ACL, the effective rights for the primary group SID from the ACL, and the effective rights for the 'Everyone' group SID from the ACL. The results are then processed into the familiar 'rwx' format of UNIX permissions for owner, group, and

other permissions and returned to the client. 'Effective rights' are the rights Windows grants to the user or group based on its evaluation of all entries in the ACL for the SID of the user or group.

If the effective rights cannot be determined using this method for whatever reason, ExtremeZ-IP will return full access for each part of the UNIX permissions that cannot be accurately determined. This will not affect actual file or folder security because we are not modifying the ACL but only returning a made-up value to the Mac. But because the Mac client sometimes decides to not process a file request because the UNIX permissions wouldn't allow it even though the actual object security setting would, it is necessary to report the most lenient permissions rather than the most restrictive. If the Mac thinks it can open a file based on UNIX permissions settings and then attempts to do so but the file security does not allow the user to open it, no harm is done, the user is restricted, and the Mac client software receives the normal access denied response.

3.1.4 Support ACLs on all volumes/Volume supports ACLs

There is both a global setting and a volume-specific setting for this feature. If the global setting checkbox is checked, it overrides any volume-specific setting.

OS X requires UNIX permissions to be supported if ACLs are to be supported.

Feature not checked (disabled):

If this feature is disabled, the ACL will not be requested by the Mac nor can it be set by the Mac. It will not appear in the 'Get Info' pane and will not be shown in Terminal with the **ls –le** command.

Feature checked (enabled):

If this feature is enabled, the Mac client will be able to view and set the object's ACL if the user has permission to do so. The ACL is viewed in the 'Get Info' pane or by issuing the **ls –le** command in Terminal. While there are difficulties setting the ACL using the 'Get Info' pane due to problems with Finder, the ACL can be easily and accurately modified using the chmod command in Terminal.

The ACEs returned to the Mac whenever an FPGetACL command is issued by the client are all the ACEs in the Windows ACL EXCEPT those that we consider our explicit UNIX permission ACEs.

In Terminal, the ACEs will show all permissions allowed or denied and whether or not the ACE is inherited. In Terminal, if the ACE UUID is for a Windows local account, i.e. BUILTIN\Administrators, SYSTEM, etc., then the UUID rather than a name will appear in the ACE and will look something like 40000012-FEEF-FEEF-FEEF-FEEF40000012. These

are UUIDs that we create due to the fact that no UUID is available from AD for local accounts. Common local UUIDs are 40000012 is SYSTEM, 40000220 is BUILTIN\Administrators, and 40000221 is BUILTIN\Users. Apple uses ABCDEFAB-CDEF-ABCD-EFAB-CDEF00000012 for 'Everyone' group. Wherever possible, the Mac will attempt to convert the UUID to a name. In Finder, this is done by the client asking the server what name goes with what UUID if it cannot locate the UUID in AD. We return the name to associate with the UUID. Snow Leopard seems to be much better at asking for the name than Leopard. Frequently, Leopard will display "unknown" in the 'Get Info' pane. That indicates that Finder was unable to resolve the UUID and did not bother to ask the server for the information.

3.2 Interaction between security feature settings in ExtremeZ-IP

If Allow Mac clients to change folder permissions is unchecked and Support UNIX permissions and ACLs is checked, the Mac will be able to view UNIX permissions, view ACLs (assuming Support ACLs on all volumes/Volume Supports ACLs is checked) but will not be able to change them from the Mac client. The UNIX permissions will be calculated from the effective rights for owner, group, and everyone using the complete ACL even if it includes explicit ACEs that would otherwise be used to determine UNIX permissions (new behavior in 6.0.2). ExtremeZ-IP will not return an error to the client when an attempt is made to change permissions but will instead return success and will not perform the change operation. This behavior is required to satisfy the Mac client when files are created, for example, when it tries to set the UNIX permissions based on the umask. If an error was returned, the Mac would report that the user did not have permission to complete the process of creating the file.

If Support UNIX permissions and ACLs is checked and Reset permissions on move is also checked, all UNIX permission ACEs added by ExtremeZ-IP will be removed from the ACL when a file is moved. The new ACL for the object will contain only the ACEs inherited from its parent folder(s).

3.3 Special Registry Keys for permission issues

There are four registry keys that can be set that effect how permissions are processed. None of the keys can currently be changed through the UI.

The keys are:

- UNIXCalculatedPermissionsMode
- UNIXGroupPermissionsMode
- DefaultGlobalUmask/DefaultUmask
- DefaultGlobalPermissions/DefaultPermissions

3.3.1 Using UNIX permissions mode keys

The UNIXCalculatedPermissionsMode and UNIXGroupErmissionsMode keys ONLY function when 'Allow Mac to change permissions' is disabled (unchecked). When the Mac cannot add UNIX permissions to a file or folder, ExtremeZ-IP will calculate them based on information found in the object's security descriptor.

Values for UNIXCalculatedPermissionsMode are:

- 0 … Use Windows Only
- 1 … Use fast mode (default)
- 2 … Use all access

Setting this key to 0 (Use Windows Only) results in a call to GetEffectiveRightsForAcl for the owner, the group, and everyone, allowing Windows to return the access rights for each of these based on the content of the ACL. This is often a slow API call because Windows will try to resolve group membership using AD as well as resolve unknown SIDs. Because this can slow down the enumeration process, it is recommended (and the default is) to use a value of 1 (use fast mode).

A value of 1 (Use fast mode) causes ExtremeZ-IP to evaluate the ACL by looking for owner ACEs, group ACEs, eliminating bad SIDs, evaluating bottom level groups, etc. The process is considerably faster than the Windows Only mode.

A value of 2 (Use all access) results in ExtremeZ-IP returning 777 (all access rights) for UNIX permissions. This is similar to the response SMB provides. Remember, we are only reporting UNIX permissions and not actually granting any additional access beyond what the ACL specifies.

Values for UNIXGroupPermissionsMode are:

- 0 …. Group calculated (default)
- 1 … Group cumulative (preferred)
- 2 … Group full rights

Setting the value to 0 (Group calculated) causes group rights to be calculated based on the primary group of the object owner and using whatever method of UNIX permission calculation the setting of UNIXCalculatedPermissionsMode specifies.

Setting the value to 1 (Group cumulative) causes ExtremeZ-IP to ignore the 'primary group' SID in the security descriptor and base the group rights on the rights allowed to all group ACEs in the ACL. Where this applies most often is when the owners are members of domain users (as they typically are) but domain users is not given any access to the files and folders but another group does have access.

For example, a certain set of students are members of a particular class group, say 'ClassOf09', as well as domain users. In most cases, all students would be members of domain users. The school might set up a folder that they only want the 'ClassOf09' and maybe 'faculty' to have access. An ACL for a folder like this could be:

- CREATOR_OWNER (full control)
- Classof09 (read, write)
- Faculty (full control)

Note the absence of any 'Domain Users' ACE. Letting Windows calculate the group permissions would result in group permissions of 0 since the 'Domain Users' group has no rights. Returning group rights of 0 would possibly result in both 'Faculty' and 'ClassOf09' groups not having access if the Mac workflow bases access decisions on group rights (Photoshop, search, etc.). However, if Group Cumulative mode is used to compute group permissions, the result would be 7 (sum of rights granted to all groups in the ACL, in this case 'Faculty' provides the most access, full control). This would allow Mac workflows to continue processing since a group does have access and the user may be a member of one of the groups having access. Again, no actual access other than that granted by the ACL itself is given.

Setting the value to 2 (group full rights) results in ExtremeZ-IP returning 7 for the group part of the UNIX permissions.

3.3.2 Using the default umask and permissions keys

ExtremeZ-IP includes a feature to overcome some of the issues customers are having with UNIX permissions and the way the Mac client is setting them. *The default umask and permissions are ONLY applied when 'Allow Mac clients to change permissions' is enabled (checked) and 'Support UNIX permissions and ACLs' is enabled (checked). The changes are applied when the Mac attempts to set permissions, not when it reads permissions.*

Specifically, with Photoshop, the Mac/Photoshop will turn off the group write permissions. It will also add 'Everyone' read permissions. To solve these problems, ExtremeZ-IP 6.0.2 and later support 'DefaultGlobalUmask' and 'DefaultGlobalPermissions' string values in the 'Refreshable' registry key and 'DefaultUmask' and 'DefaultPermissions' values in the volume multi-string.

Whenever a SetFileParms or SetDirParms command would set the UNIX permissions, we see if we have a default umask to apply. We first check to see if the volume default umask exists and if so, use it; if not we check to see if the default global umask exists and if so, use it; otherwise we do no umask processing. Basically, a volume default umask overrides the default global umask. And if neither exists, we do nothing.

After the default umask processing, we check to see if there is a default permissions setting to apply. Again, first we check to see if the volume default permissions exists and if so, use it; if not, we check to see if the default global permissions exists and if so, use it; otherwise we do not modify the permissions being set. The volume specific default permissions override the default global permissions.

The default umask is subtractive just like it is on the Mac. A good use case would be where the customer does not want 'Everyone' to have access to files but the Mac is setting permissions to 644 meaning 'Everyone' has read permissions. By adding either the 'DefaultGlobalUmask' set to 007 or the volume specific 'DefaultUmask' set to 007, all permissions for 'Everyone' would be removed from the setting being applied by the Mac.

The default permissions are additive and are applied after the default umask. The default permissions would be used in a case where the Mac is not properly setting the UNIX permissions, as is the case with Photoshop. The customer wants the group to have read and write access but the user's umask is removing the ability for the group to write to the file. While the administrator wants permissions of 660, the Mac is setting them to 644. By using a 'DefaultGlobalPermissions' value of 020 or volume specific 'DefaultPermissions' value of 020, write permissions for the group can be enabled. The permissions are or'd with (added to) the permissions being set by the Mac.

Use the default umask to turn OFF permission settings, and use the default permissions to turn ON permission settings.

Then in the case where the administrator doesn't want 'Everyone' having any permissions but the Mac is turning on permissions for 'Everyone', and the Mac is turning off write permissions for the group but the administrator wants it on, the solution would be to set the volume registry multistring to include:
DefaultUmask=007
DefaultPermissions=020.
We will take the UNIX permissions setting that we receive from the Mac, remove all permissions for 'Everyone' and then turn on write permissions for the group.

They could just as well set the DefaultGlobalUmask and DefaultGlobalPermissions values in 'Refreshable' to these same values. However, the values would then apply to all volumes and any setting of UNIX permissions.

3.3.3 Important notes about this feature:

- The settings are applied when the Mac sets the UNIX permissions. Therefore, any permissions that are already set on any file or folder before these keys are enabled will not change.

- There isn't a lot of checking of the values. They are specified as a three character string. Be sure the values are correct 3-character octal UNIX permission settings.

- There are no UI changes at this time for these settings. They must be entered manually. Group Logic Support can instruct customers on how to set these up on a case-by-case basis.

- The default settings in no way affect any ACEs in the ACL that are inherited. As a result, if the ACL includes an inherited ACE for 'Everyone', we will not modify it in any way. It will still be inherited according to Windows rules of inheritance.

- The default values ONLY affect the explicit ACEs we add to the ACL for supporting UNIX permissions.

4 General Security notes

ACLs containing SIDs that are local to a computer on the network are often very slow to enumerate. Even though we cache the security descriptor, when the Mac gets a UUID it does not have a name cached for, it will attempt to get the name from AD. This can take 2-3 seconds per check and it might check for each item it enumerates if that UUID is the group or owner UUID. Snow Leopard asks the ExtremeZ-IP server for the name resulting in faster enumeration of ACLs.

If the owner is not a member of a domain then the primary group will be the <local computer name>\None group. The <local computer name>\None group has no domain rights. As a result, if there is no specific ACE for this group, no group rights will be granted.

The Mac will not be able to look up local computer SIDs but may still ask AD for them and will fail. This is slow. Local computer SIDs should be avoided as much as possible.

4.1 How to set up security

If you want to use *only* Windows security then set the following:

- Allow Mac clients to change folder permissions – NOT checked
- Reset permissions on move – NOT checked

- Support UNIX permissions and ACLs – NOT checked
- Support ACLs on all volumes/Volume Supports ACLs – NOT checked

If you want to use full Mac security (UNIX permissions and ACLS) then set the following:

- Allow Mac clients to change folder permissions – checked
- Reset permissions on move – NOT checked
- Support UNIX permissions and ACLs – checked
- Support ACLs on all volumes/Volume Supports ACLs – checked

If you want Mac security (UNIX permissions only) but do not want Mac clients to see or modify the windows ACL then set the following:

- Allow Mac clients to change folder permissions – checked
- Reset permissions on move – NOT checked
- Support UNIX permissions and ACLs – checked
- Support ACLs on all volumes/Volume Supports ACLs – NOT checked

If you want Mac security (UNIX permissions and ACLs) but do not want Mac clients to modify the UNIX permissions or ACL then set the following:

- Allow Mac clients to change folder permissions – checked
- Reset permissions on move – NOT checked
- Support UNIX permissions and ACLs – checked
- Support ACLs on all volumes/Volume Supports ACLs – checked

4.2 Advanced troulbeshooting information

Group Logic Support may request certain information to troubleshoot difficult permissions issues so that they can be escalated to the Development team. This information may include:

- ICACLS of the file/folder in question and its parent folder.
- Debug log with AfpSecurityACL set to DEBUG. Caution, this generates a lot of data. Enable it, do the test, and turn it off.
- A packet capture, ideally from the server point of view. Capturing from the client frequently drops response packets from the server making it impossible to see what action ExtremeZ-IP took with the request and if an error was returned.
- The name of the file or folder that is causing the problem so we can find it in the logs and packets. The complete path is required as other copies of the same file may exist on the system and therefore appear in the log and packets.

5 General Security Questions

Why can't I see the security for a file or folder on Windows? I'm an admin. It looks like the ACL is gone. I can only take ownership.

This is actually a normal security condition. It depends very much on what security was inherited from the object's parent folder. In a simple scenario, let's assume that only SYSTEM=FC was inherited. The Mac user creates a folder or file and the permissions get set to 700 by the user or the application on the Mac. The ACL will have two ACEs…SYSTEM=FC and <user>=FC. The owner will be the user. When you try to look at the security on Windows you will not see any ACL because you do not have permission unless you are that user. Administrators on Windows have no inherent rights to files or folders other than change ownership. Once an administrator takes ownership they can add to the ACL so they can have permission to read/write/delete. Windows does not convey these rights to an owner automatically as it does in UNIX. The only default owner rights are change ownership, change permissions, and read permissions.

Why can a script/app on ExtremeZ-IP be executed by any user even though only the owner has execute permission?

This appears to be a general AFP bug and GroupLogic has submitted a bug report to Apple. As long as the user can read the file and anyone higher (to the left) in the UNIX permissions has execute permission, i.e. 'owner' has execute permission and 'other' has read permission, then the user can execute the file.

How come if I assign the group no permissions or only read permissions I see full access for the group, i.e.rwx, when I do an 'ls -l'command? (ExtremeZ-IP 6.0.2)

OS X and some applications still make decisions about who can read, write, or execute a file based on the UNIX permissions even though the object has an ACL, which would allow the user or group access to the object. In some cases, ExtremeZ-IP cannot determine the effective rights a group has to a file or folder because the Windows API used for this purpose is deficient and will return an error rather than the effective rights. In order to satisfy the Mac apps that still rely on UNIX permissions, we return full access rather than no access. This allows the Mac to continue to attempt the desired action such as reading the file. However, if the user really has no access, the request to read the file will fail with afpAccessDenied (-5000) regardless of the UNIX permission settings.

Why won't the Mac allow a member of a group to read and write to the file? I added an ACE to the ACL for the group and gave them full control but the Mac still won't let the user use the file.

The published order of security evaluation on the Mac is first to see if the ACL allows the desired access to the file. If not, then it is supposed to see if the UNIX permissions would allow access. If not then access is denied.

In fact, the Mac CatSearch and some applications such as Photoshop check the UNIX permissions first and don't even bother to check the ACL. The result is that the user is denied access to a file or folder that they should actually be allowed to access.

In the case of CatSearch, the Mac looks at the UNIX permissions returned with the search enumeration and if the user does not have access granted by the UNIX permissions, the file is not displayed in the results. So if a file in the result set had UNIX permissions of 704 and the user was a member of the group but not the owner, they would not have access to the file and the file would not appear in the results.

In the case of Photoshop, the umask causes the group write permission to be removed. When the UNIX permissions show that the group member cannot write to the file, no further checking is done. However, there may very well be another ACE in the ACL that grants the user write permissions to the file. This will be totally ignored because the software on the Mac will not attempt to read or write to the file even though it would be allowed to proceed.

5.1 How inherit-only ACEs work

Any ACE marked as inherit-only (shown as (IO) in cacls and icacls) is not used as part of the calculation of access rights for the object in which it is found. Instead it is passed on to child objects based on the inheritable flag settings of object inherit (shown as (OI) in cacls and icacls) and container inherit (shown as (CI) in cacls and icacls). You will only find inherit-only ACEs in ACLs of folders. If, when looking at an ACL for a folder, you found an ACE for 'Administrators' that was inherit-only, inheritable by files and inheritable by folders, that had full control AND you found an ACE for 'Administrators' that was NOT inherit-only and only had read and traverse folder access, then a member of the 'Administrators' group could read the contents of the folder because the effective ACE would grant that access right. The inherit-only ACE would be passed to child objects with the inherit-only flag removed, and the 'Administrators' group would be granted full access to all child files and folders.

There are two special inherit-only ACEs recognized by the Mac and Windows. These are CREATOR_OWNER ('owner' on the Mac) and CREATOR_GROUP ('group' on the Mac). These ACEs are place-holders and are not used in the computation of access rights when an ACL is evaluated since they are inherit-only ACEs. However, Windows (and ExtremeZ-IP) will add an ACE to any descendants of the folder with the rights granted by the inherit-only ACE but with the SID of the owner, or SID of the owner's primary group, respectively. The CREATOR_OWNER or CREATOR_GROUP ACE will also then be passed on to descendant folders. This method is used to provide rights for the creator (owner) of the file or folder, typically, since in Windows, the owner has no inherent rights such as read or write for files or folders it creates.

6 GLOSSARY:

ACE - **access control entry** permissions attached to an object for a specific user or group. The ACE specifies what access is allowed or denied for the user or group to the object, the type of inheritance to allow, and the SID of the user or group to which the access applies.

ACL - **access control list** (**ACL**) is a list of access control entries (ACEs).

Effective ACE – an ACE that is used when calculating access to an object.

Effective rights - the rights Windows grants to the user or group based on its evaluation of all entries in the ACL for the SID of the user or group.

Explicit ACE – an ACE that was not inherited from a parent object.

Inherited ACE – an ACE that was inherited from a parent object.

Inherit-only ACE – an ACE that is not used in the calculation of access rights but which results in an effective ACE being added to child objects. CREATOR_OWNER is a common inherit-only ACE.

Mac Access rights – the folder access rights (OS 9 style) when no UNIX permissions or ACLs are enabled.

Owner – the security ID (SID) of the entity considered by Windows as the owner of the object. The owner has certain rights in Windows such as change permissions which cannot be revoked.

Primary Group – the security ID (SID) of the group entity that is assigned to the owner in Active Directory. The primary group for most users is 'Domain Users'. However, this is easily changed in AD.

SID – (Security Identifier) Every user, group, and computer has a unique SID identifying it within the security system.

SD – (Security descriptor) Every secured object in Windows, i.e. file or folder, has a security descriptor that identifies the SID of the owner, the SID of the primary group, and the discretionary ACL.

Umask - (abbreviated from ***user mask***) is a command and a function in POSIX environments which sets the default permission modes for newly created files and directories of the current process. Umask is subtractive and removes access rights from the UNIX permissions. If the UNIX

permissions being set are 777 and the umask is 022, then the resulting UNIX permissions are 755.

UNIX Permissions - There are three specific permissions on Unix-like systems that apply to each class:

The *read* permission, which grants the ability to read a file. When set for a directory, this permission grants the ability to read the **names** of files in the directory (but **not** to find out any further information about them, including file type, size, ownership, permissions, etc.)

The *write* permission, which grants the ability to modify a file. When set for a directory, this permission grants the ability to modify entries in the directory. This includes creating files, deleting files, and renaming files.

The *execute* permission, which grants the ability to execute a file. This permission must be set for executable binaries (for example, a compiled c++ program) or shell scripts (for example, a Perl program) in order to allow the operating system to run them. When set for a directory, this permission grants the ability to traverse its tree in order to access files or subdirectories, but not see files inside the directory