NeoExec for Active Directory Administrators' Guide

Version 1.1

Liability Notice

Information in this manual may change without notice and does not represent a commitment on the part of NeoValens.

The software described in this manual is provided by NeoValens under a license agreement. The software may only be used in accordance with the terms of the agreement.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of NeoValens.

NeoValens claims copyright in this program and documentation as an unpublished work, revisions of which were first licensed on the date indicated in the foregoing notice. Claim of copyright does not imply waiver of other rights by NeoValens.

Copyright 2003-2004 © NeoValens S.A.
All rights reserved.

Trademarks

NeoExec is a registered a trademark of NeoValens S.A.
All other trademarks recognized.

NeoValens S.A.
66, Rue de Luxembourg
L-4221 Esch-sur-Alzette
Luxembourg

Email: support@neovalens.com
Web: www.neovalens.com

Published on: September 2004

Contents

Introduction

This document describes how to administer NeoExec for Active Directory (NeoExec/AD).

NeoExec/AD is an innovative solution that allows you to manage your desktops and servers in a more secure manner. There exists a number of applications that require elevated privileges to run and, without NeoExec, the only solution available was to grant such privileges to the user. NeoExec on the other hand allows you to set the privileges on a per-application basis thereby helping in creating a more secure, manageable environment.

Privileged Applications Vs Privileged Accounts

Members of the local Administrators group have privileges that allow them to perform any action on a computer. Users are often made members of the Administrators group because some applications require elevated privileges to run. The problem is twofold: users often abuse of those privileges to install new applications and/or to modify the configuration of their computer and, possibly even more important, users with elevated privileges are more vulnerable to viruses and trojans. Most malware requires elevated privileges to be installed and to replicate, and members of the local Administrators group are the primary target.

The principle of *least privilege* states that users should be granted the most restrictive set of privileges needed for the performance of authorized tasks. Application of this principle limits the damage that can result from accident, error, or unauthorized use of an information system (IS).

NeoExec helps in applying the least privilege principle by restricting elevated privileges to selected applications: we call such applications *Privileged Applications*.

NeoExec/AD components

NeoExec/AD consists of two components:

- NeoExec/AD MMC Snap-In.
- NeoExec/AD kernel driver and group policy extension

The NeoExec/AD MMC console allows you to define the NeoExec policies, that is, the list of Privileged Applications and their parameters. The MMC console plugs-in the Group Policy mechanism of Windows.

The MMC console can be used to define policies at the Site, Organizational Unit and Domain level. You can either install the NeoExec/AD MMC console on one or more Domain Controller or Windows 2000/XP workstations with the Administration Pack installed. Either way, the NeoExec node, named "Application Execution Policies", will appear in the Group Policy editor as a new node of the Computer Configuration node. The NeoExec/AD policies will automatically be deployed to the client computers by means of the built-in GPO deployment mechanism.

The MMC console can also be used to define local policies and it appears as a node of the local Group Policy snap-in (gpedit.msc). Such snap-in can be invoked from the NeoExec/AD shortcut menu or from the Run window by typing "gpedit.msc". You only need to install the NeoExec/AD MMC console on the computers for which you want to define local policies, typically computers not part of a domain.

The NeoExec/AD kernel driver reads the policies from the registry and applies them as the users executes Privileged Applications. When an end user launches a privileged application NeoExec modifies the process token on the fly adding the groups that you have set in the policies, thereby allowing the user to run the application as if he/she was a member of such group.

New for NeoExec/AD

- MMC administrative console integrates into Windows 2000/2003 Group Policy management
- Automatic policy distribution by means of Group Policy Objects
- Wizard based interface
- Simplified "command lines" management for built-in MMC snap-ins and Control Panel applets
- Ability to select a group other than Administrators (*Privileged Group*)
- Ability to restrict execution of Privileged Applications to selected users by means of ACLs (*Authorized Users*).

New for v1.1

- You can now have more than one Privileged Group. This allows you to mix both well-known groups such as Administrators and Power Users with both local and domain groups as required. Having more than one group allows you to create per-application ACLs based on such groups.
- You can now remove one or more groups from the process token. This allows you to remove sensitive groups when executing critical applications. This is especially useful when users with elevated privileges need to access the internet as NeoExec can revoke privileges of your choice for selected applications (such as Internet Explorer and Outlook for instance).
- You can now set per-process operating system privileges. Some processes require a number of privileges in order to work. Up to now such privileges would have been granted to the user but it is now finally possible to grant them on a per-application basis. By default, NeoExec will use
- Support for Windows Server 2003

NeoExec/AD MMC Console

NeoExec/AD is managed by means of a snap-in that fully integrates into Windows 2000 and Windows 2003 Group Policy based administration. When installed, NeoExec appears as a new node of the Group Policy editor as show below. The node is named *Application Execution Policies*.

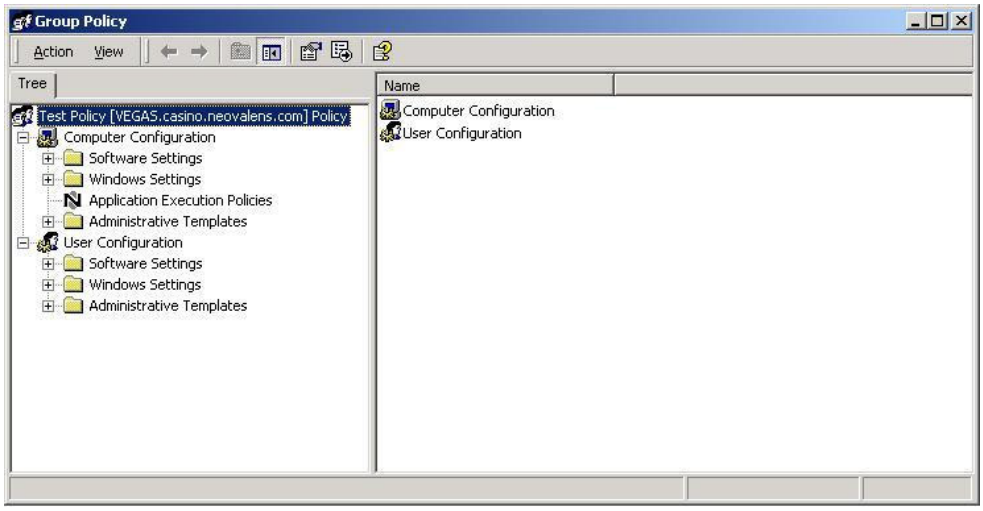


Figure 1: Group Policy editor

Local policies can also be set by means of the Group Policy editor by running "gpedit.msc" from the Run window or the command line.

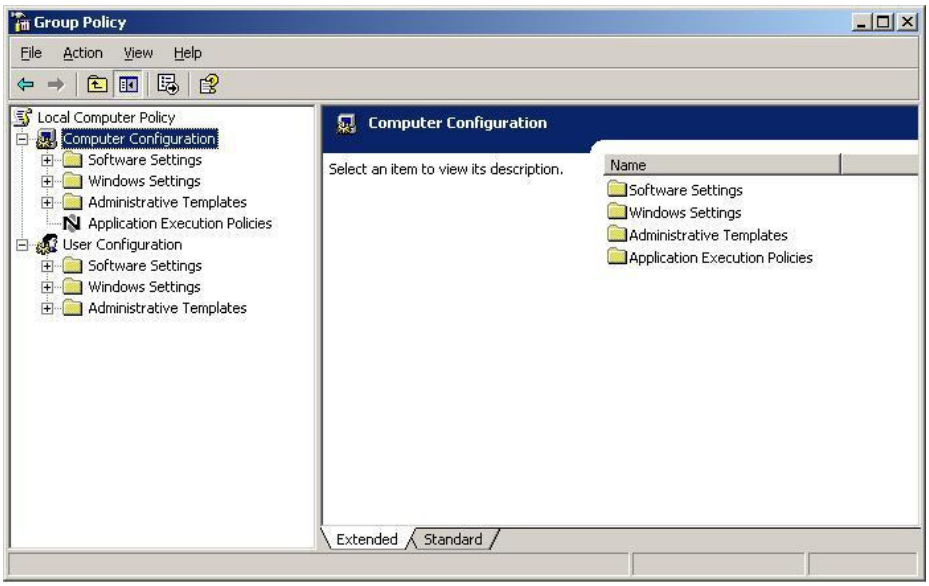


Figure 2: Local Group Policy editor

NeoExec policies consists of selecting the applications you want your users to execute with special privileges and setting their parameters, if any. The entire process is guided by a wizard as shown next.

The rule creation Wizard

To create a new rule select the NeoExec node and right-click on it and then select *New -> Privileged Application...* as shown in Figure 3.

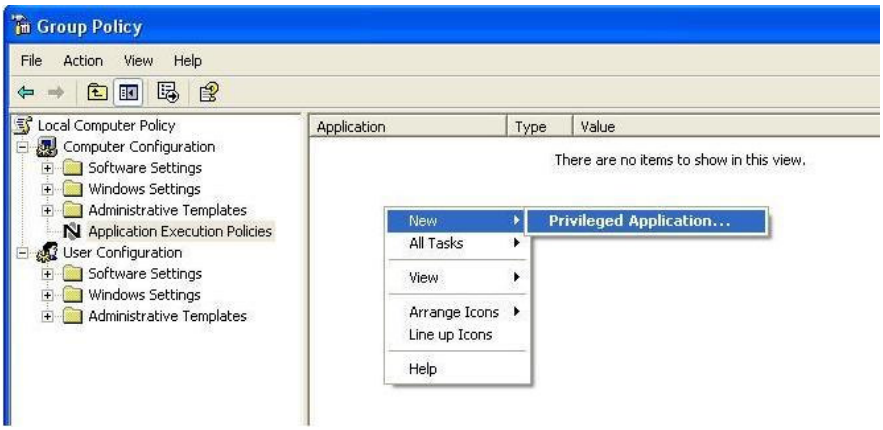


Figure 3: invoking the rule creation wizard

The wizard will guide you through the different steps: help is always available by clicking on the Help button.

Selecting a Rule type

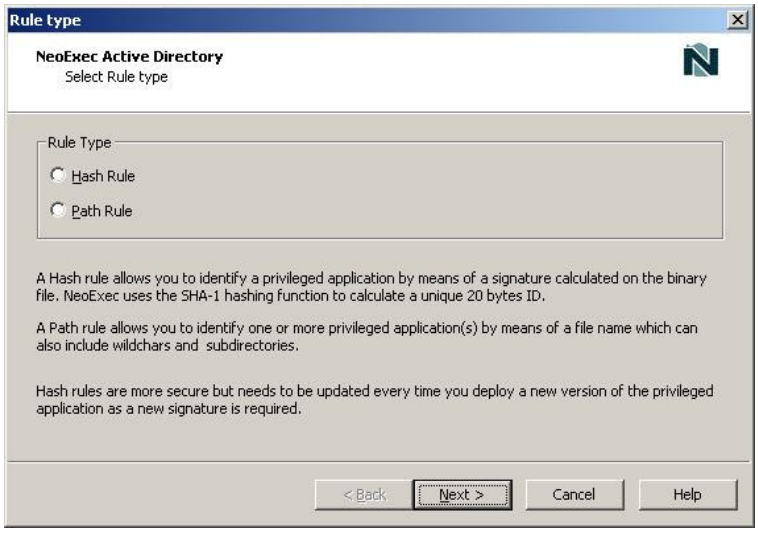


Figure 4: selecting a rule type

NeoExec supports two rules types: by Hash and by Path.

The Hash rule is the most secure as it calculates a unique identifier (SHA-1 message digest) of the executable file. The Hash rule is therefore independent of the file location but the rule will need to be updated whenever you will deploy a new image of the file. The Hash rule can only be applied to one executable file at the time.

The Path rule allows you to select one or more executable files by means of wild chars specifiers. The path rule is ideal when you need a rule for a number of applications that are subject to frequent changes and that are stored in a location under your control (e.g.: network share).

Once you have selected the rule type click on next to set its basic attributes.

Privileged Applications

Once you have selected the rule type it is time to select the application which will become a Privileged Application. For a hash rule all you need to do is to click on the *Browse* button an select the executable file. An example of a hash rule is shown in Figure 5.

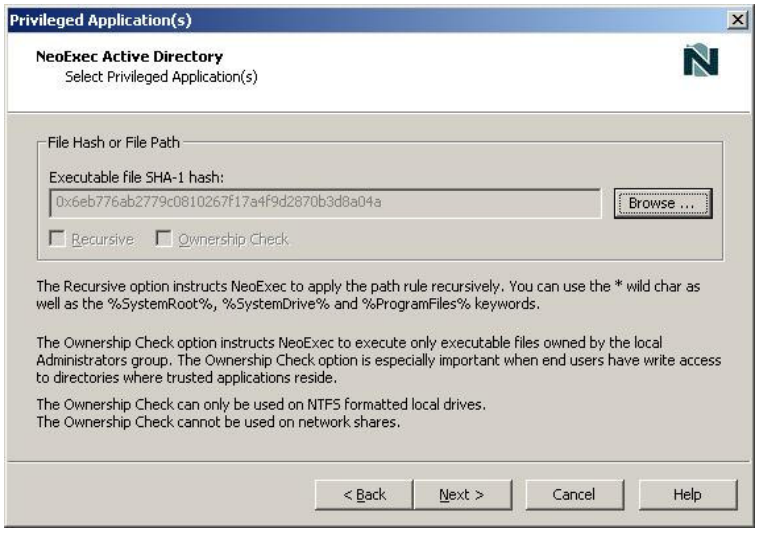


Figure 5: example of hash rule

The *Recursive* and *Ownership Check* options are disabled for Hash rules as they apply only to Path rules.

A Path rule, as shown in Figure 6, can apply to a single file. The process is identical to the one of the Hash rule and all you need to do is to click on the *Browse* button and select an executable file.

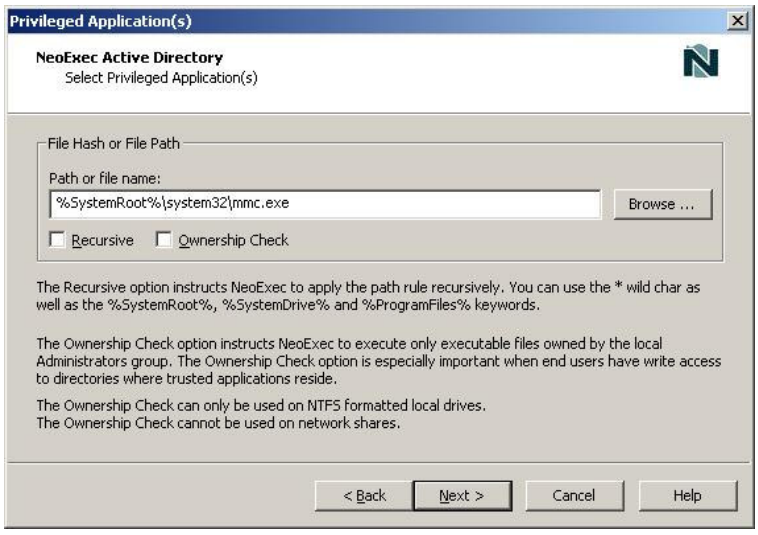


Figure 6: example of path rule

Path rules can also use wild chars specifiers. The following are example of valid path rules:

Rule example	Rule target
C:\Windows\regedit.exe	for regedit.exe
C:\Windows\System32\mmc.exe	for mmc.exe
C:\Program Files*.exe	for all executables found in the C:\Program Files directory.
%SystemRoot%\System32\net.exe	The rule targets is the net.exe application. This rule does not depend on the system root name (WINNT, WINDOWS..)
%SystemDrive%mydir\myApp.exe	This rule targets the application named myapp.exe located under the mydir directory on the system drive. The system drive is the drive where Windows is installed.
%ProgramFiles%\Internet Explorer\iexplore.exe	This (sample!) rule allows you to execute Internet Explorer. This rule uses %ProgramFiles% and it will work on all versions of Windows 2000 and Windows XP.

If you are targeting a single file, and the file location on both your computer and the target computer is the same, it is recommended to use the *Browse* button. In all other cases it is better to enter the File Path by hand.

> **Note**: when targeting multiple files in a directory you must append the appropriate wild char to the directory specifier. For example, C:\test*.exe is a valid path while c:\test is not.

Keywords

%SystemDrive% Use it in place of the system drive (usually C:).

%SystemRoot% Use it in place of a hard-coded system root such as c:\winnt or c:\windows.

%Program Files% Use it when targeting the Program Files directory.

The ***keywords are case-sensitive***.

Network Drives

When targeting Network drives you need to specify the file location by using *server**sharename**directory* rather than using mapped network drives.

Ownership Check

The Ownership Check option, when selected, instructs the NeoExec kernel driver to check who is the owner of the executable file. Only files owned by the local Administrators group will be trusted. The Ownership Check option is especially important when end users have *write access* to directories where privileged applications reside.

Please note that the Ownership Check can only be used on NTFS formatted local drives. The Ownership Check cannot be used on network shares.

> Beware that files created by the user Administrator will be owned by the Administrator (the user) and will fail the ownership check. The files must be owned by the Administrators group. The same applies to any other member of the local Administrators group.

Recursive

The Recursive option, when checked, instructs the NeoExec Driver to apply the path rule recursively.

Click on next to optionally add any command line.

Command Lines

You may want to restrict the execution of certain privileged applications so that they can only be launched with a specific command line argument. The most obvious examples are container applications that allow you to load and/or launch other modules. For example, all Control Panel applets are either .cpl extensions run by means of rundll32 or shortcuts to administrative MMC snap-ins run by means of mmc.exe. It would be unsafe to grant end users unrestricted access to those applications. A much safer approach is to limit the execution of such applications to run only when invoked with particular command line arguments.

The Command Lines main window, as shown in Figure 7, allows you to add, delete and edit command lines. To Delete and Edit a command line you need to select it first.

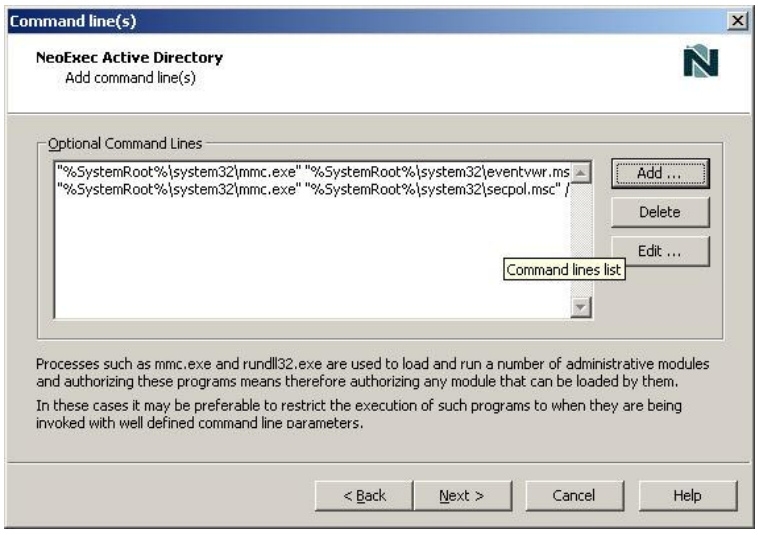


Figure 7: command lines list

Clicking on Add will bring up the *Get Command Line* window. The window has three tabs:

Running Processes

All (user mode) running processes are shown. One easy way to determine an application command line is to run it and then inspect the command line in this window. Systems Processes will not be shown (and a warning given) if you do not have Debug Privileges. To select a running process command line click on it in the list and the full command line will appear in the *Command Line* text box. Clicking on OK will add it to the command lines list.

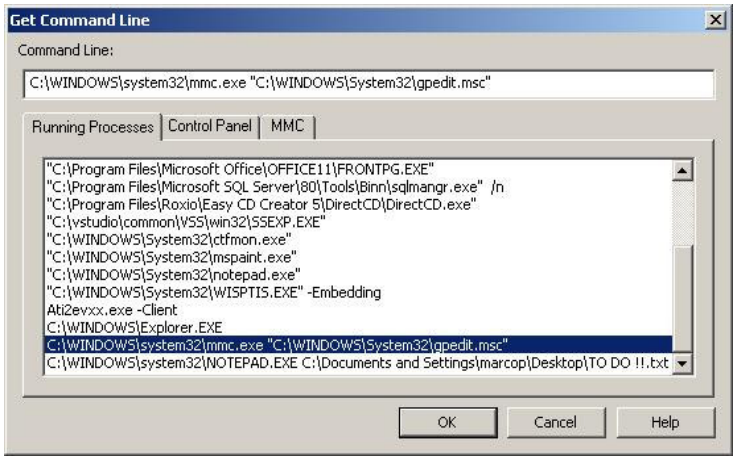


Figure 8: running processes example

Control Panel applets

This window lists all known control panel applets.

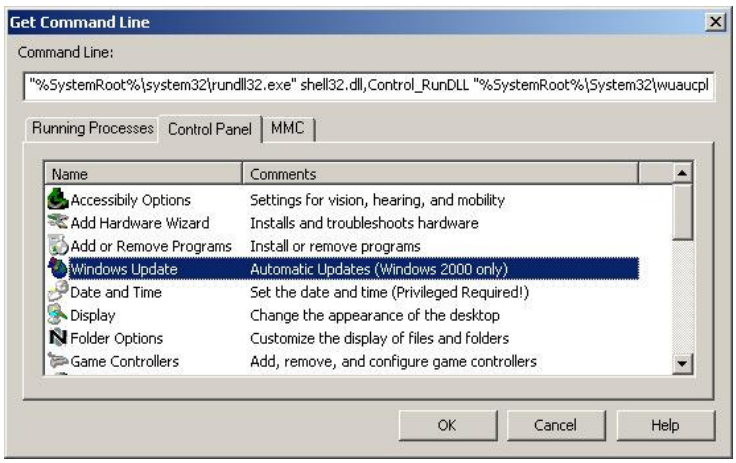


Figure 9: Control Panel applets example

MMC snap-ins

This window lists a sub-set of known MMC snap-ins. Management snap-ins that operate across the network are not listed.

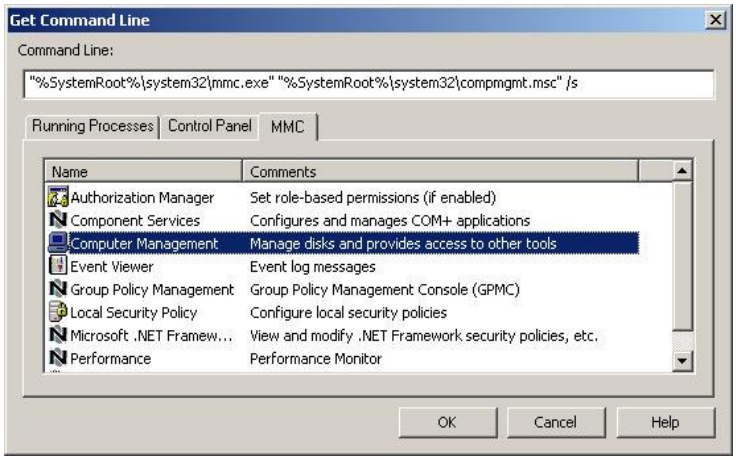


Figure 10: MMC snap-in example

Warning: one common mistake is to select one application as Privileged Application and then a command line for a different application. Unless the command line refers to the same application you have selected in the Privileged Application screen it will never run,

Click on Next to set the NT Group to be added to the end user token.

Privileged Groups

The Privileged Group window allows you to set the NT group SID that will be added to the end user token for the Privileged Application. The default is Administrators but it can be set to any Local and Domain group.

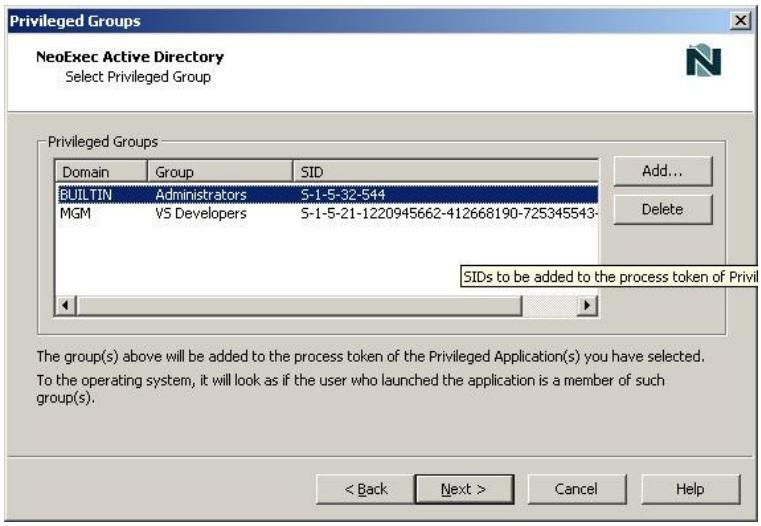


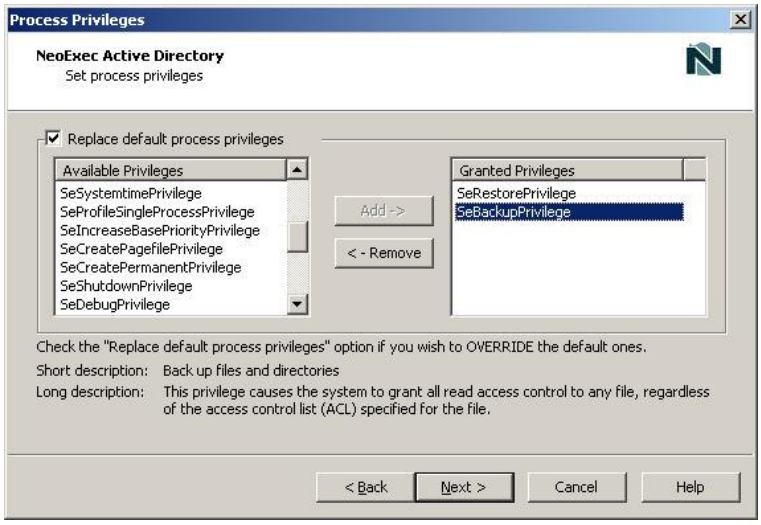
Figure 11: Privileged Group example

The Privileged Group is a very powerful feature that opens up a number of new possibilities. For example:

- You can restrict one or more applications to specific directories
- You can restrict one or more applications to read/write (or not) to specific keys
- When used together with SecureWave SecureNT, you can restrict applications to selected IO ports
- You can audit usage of Privileged Applications (as opposed to audit all applications)
- You can execute one or more applications as Power User rather than making the user a member of the group
- You can allow users to run Windows Update as opposed to make them Administrators

Process Privileges

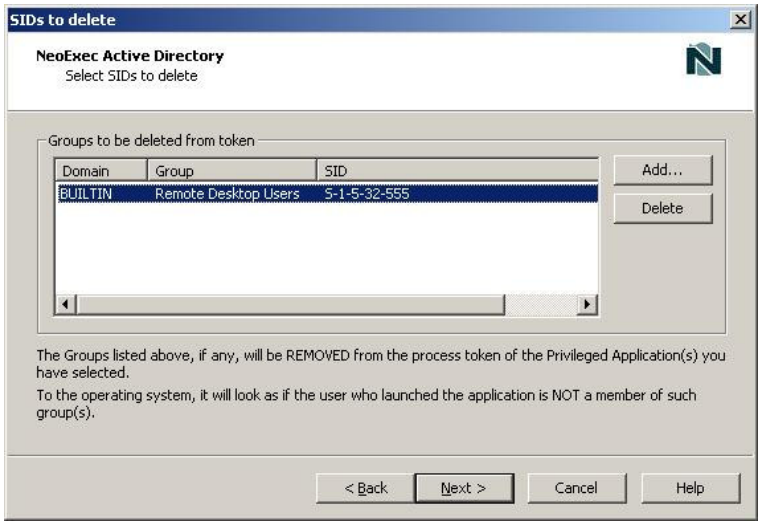
Some processes require a number of privileges in order to work. Up to now such privileges would have been granted to the user but it is now finally possible to grant them on a per-application basis. By default, NeoExec will use the user privileges but by checking the "Replace default process privileges" check box you can override the default behavior.



Click on the Add button to grant a privilege and on the Remove button to delete the privilege from the "Granted Privileges" list.

Groups to delete

NeoExec allows you also to optionally remove one or more groups from the process token. Removing groups is useful for sensitive application run by users with elevated privileges. Common examples are email clients such as Outlook and Internet browsers such as Internet Explorer run by users with Administrative privileges. The solution, for those users that require such privileges, is to remove the un-necessary groups for these applications.



Authorized Users

The Authorized Users lists allows you to define who can run Privileged Applications with elevated privileges; users excluded from the list below will still be able to run such applications but no privilege will be granted to them.

This window is especially useful for local policies as there is no mechanism to filter to whom a policy should apply. It can also be used at the Site/OU/Domain level in conjunction, or in place of filtering Group Policy according to group membership.

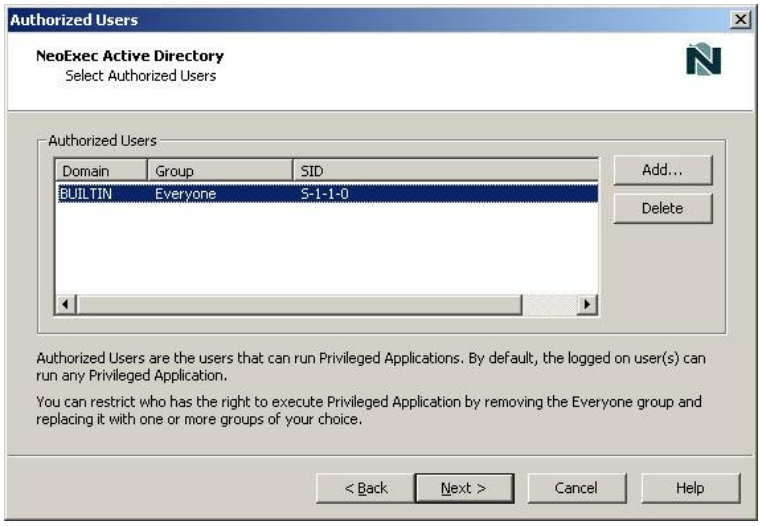


Figure 12: Authorized Users example

By default, as shown in Figure 12, the rule applies to the Everyone group, that is, to all users. This means that all users logged on interactively will be able to execute the Privileged Application and be granted the related privileges. To control who should be able to run the Privileged Application you should remove the Everyone group and replace it with one or more NT groups.

Click on Next to move to the last screen.

Saving the new rule

The rule is complete. Clicking on Finish will save the rule in the registry. Local Policies will be applied immediately while Site/OU/Domain policies will be replicated and deployed according to your Active Directory site topology and settings.

Editing a rule

To edit a rule click on the *Edit* button located to the right-side of the rule you want to edit. Clicking on the *Edit* button will invoke the Rule Editor for the selected rule.

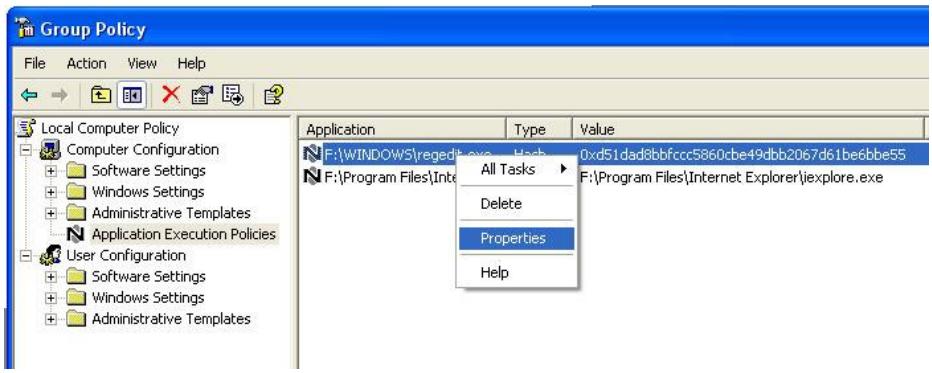


Figure 13: Editing a Path Rule

For a Path Rule, you can change all attributes but the path itself. For a Hash Rule, you can only add or remove command lines.

Deleting a rule

To delete a rule click on the Delete button located to the right side of the rule you want to delete.

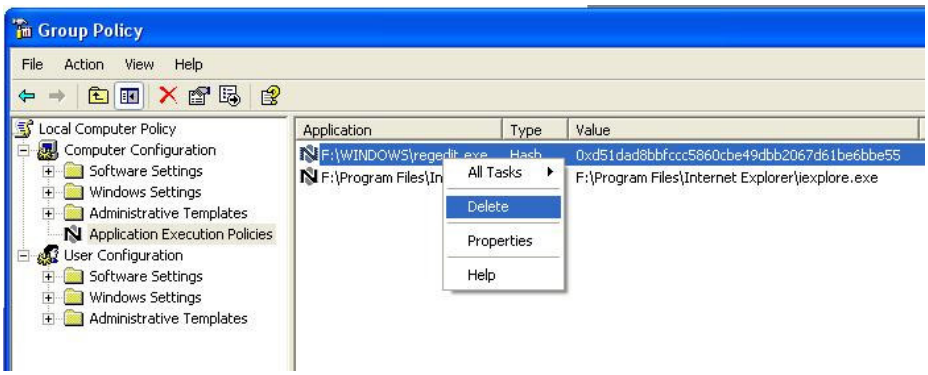


Figure 14: deleting a rule

Licensing

NeoExec/AD can be tested for evaluation purposes for a maximum of thirty (30) days after which a license must be purchased or you must uninstall the product. When you purchase a license you will receive a license file that you will then need to deploy to all workstations. The deployment is automatic and all you need to do is to import the license file from the MMC console as shown in figure 15.

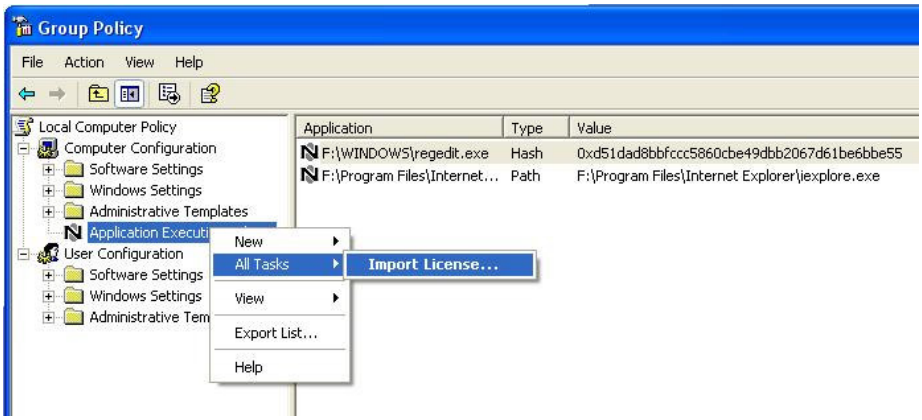


Figure 15: importing the license file

Warning: you need to pay special attention to the *scope* of the Group Policy Object that will carry the License file. You may want to create a new GPO specifically for the license file and no rule and apply it to the Site/OU/Domain according to the number of licenses you have purchased.

NeoExec Monitor

The NeoExec Monitor (NeoMon) application is typically used when in the process of creating NeoExec policies as it shows all processes being executed as well as when a NeoExec rule is applied. NeoMon is particularly useful when one wants to restrict a particular process to a specific command line but such command line is not obvious or a rule does not seem to be applied or have any effect on a process. As NeoMon shows the full command line used to start the process one needs only to copy the line from the NeoMon window for the process and paste it into the Command Lines property page in the NeoExec MMC snap-in.

NeoMon will also show on screen when a process being started matches a given Hash or Path rule.

- For a Path rule match, NeoMon will show: **Path rule match: .. process_name.exe**
- For a a Hash rule match, NeoMon will show: **Hash rule match: 0x34AF23607AD5AFA9E61B6A96CEC811E6BDC50B4A**

If the rule has no command line then NeoExec will apply the rule and **process_name.exe has been NeoExec[ed]** will be shown on screen following the rule-match line in the NeoMon window as shown above.

If one or more command lines have been set then NeoExec will examine them one by one and a log will be shown in the NeoMon window. The example below shows the output for the rundll32.exe process when started from the "Accessibility Options" control panel applet.

```
Checking command lines ...

 Input command line: "C:\WINDOWS\system32\rundll32.exe"
C:\WINDOWS\system32\shell32.dll,Control_RunDLL
"C:\WINDOWS\system32\access.cpl",Accessibility Options

 Checking command line: "C:\WINDOWS\system32\rundll32.exe"
C:\WINDOWS\system32\shell32.dll,Control_RunDLL
"C:\WINDOWS\System32\access.cpl",Accessibility Options

Command line match: "C:\WINDOWS\system32\rundll32.exe"
C:\WINDOWS\system32\shell32.dll,Control_RunDLL
"C:\WINDOWS\System32\access.cpl",Accessibility Options

rundll32.exe has been NeoExec[ed]
```

Appendix A: Events logged to the System event log

The NeoExec kernel driver logs a number of events to the System event log. The following table describes the events and the conditions that trigger them.

ID	Type	Message	Condition
28672	Error	Unsupported Operating System detected. NeoExec Professional supports only Windows 2000 and Windows XP.	This message is logged when NeoExec Professional detects an Operating System other than Windows 2000 or Windows XP.
28673	Informational	NeoExec loaded	The driver was loaded and started successfully.
28675	Warning	NeoExec is in trial mode due to the lack of a valid license. NeoExec will be disabled after 30 days unless an appropriate license is obtained. In the latter case, no reboot is required	Could not find license file (neoexec.lic) or the license is not valid.
28676	Error	NeoExec is in Trial Mode and has been disabled due to the lack of a valid license. To enable NeoExec, replace the current license (if any) with an appropriate license. No reboot is required.	This message is given once the trial period has elapsed. NeoExec will no longer function until a valid license file is provided.
28677	Error	NeoExec has loaded but is disabled due to a license violation. To enable NeoExec, replace the current license with an appropriate license. No reboot is required.	The number of computers running NeoExec is greater than the number licensed. Contact NeoValens to obtain a new license.
28678	Informational	NeoExec license check OK.	The license file was found and verified successfully.
28679	Warning	No public key file has been found: NeoExec will use the default one.	This error should never occur. Please contact NeoValens if this error is ever logged.
28680	Error	Cannot read public key file.	The public key was found but an error occurred while reading it.
28681	Error	Invalid signature detected in file neoexec.cfg.	The signature was found but is invalid. The configuration file will be ignored.
28682	Error	No signature detected in file neoexec.cfg.	The configuration file does not appear to be signed at all and will be ignored.
28683	Warning	NeoExec config file has changed.	A new configuration file has been copied to the Neo directory.
28684	Warning	No configuration file found. Save neoexec.cfg under SystemRoot\System32\Neo.	This event is typically logged when you start NeoExec for the first time. This message will no longer be logged once you supply a configuration file.
28685	Error	Unsupported Operating System detected. NeoExec Active Directory supports only Windows 2000, Windows XP and Windows 2003.	This message is logged when NeoExec/AD detects an Operating System other than Windows 2000, Windows XP, or Windows 2003.
28686	Warning	NeoExec registry policy changed.	This event is logged by

			NeoExec/AD kernel driver when it detects a change in the policies.

Appendix B: Privileges

Operating System privileges and logon rights are referred to as *User Rights*. User rights are assigned by using the Group Policy MMC snap-in. After you have started MMC and opened the Group Policy snap-in, use the console tree pane to ocate the User Rights Assignment folder. It is located under Local Computer Policy/Computer Configuration/Windows Settings/Security Settings/Local Policies. The following list shows the privileges that you can assign to a user by setting user rights.

- Act as part of the operating system
- Add workstations to a domain
- Back up files and directories
- Bypass traverse checking
- Change the system time
- Create a token object
- Create permanent shared objects
- Create a pagefile
- Debug programs
- Enable trusted for delegation on user and computer accounts
- Force shutdown from a remote system
- Generate security audits
- Increase quotas
- Increase scheduling priority
- Load and unload device drivers
- Lock pages in memory
- Manage auditing and security log
- Modify firmware environment values
- Profile a single process
- Profile system performance
- Replace a process-level token
- Restore files and directories
- Shut down the system
- Take ownership of files or other objects
- Unlock a laptop

Source: Microsoft Developer Network (MSDN)

By default, members of the local Users group have the following privileges:

Privilege	Description	Status
SeChangeNotifyPrivilege	Bypass traverse checking	enabled
SeShutdownPrivilege	Shut down the system	disabled
SeUndockPrivilege	Remove computer from docking station	enabled

Disabled privileges can be enabled programmatically.

Members of the local Administrators group have the following privileges:

Privilege	Description	Status
SeChangeNotifyPrivilege	Bypass traverse checking	enabled
SeSecurityPrivilege	Manage auditing and security log	disabled
SeBackupPrivilege	Back up files and directories	disabled
SeRestorePrivilege	Restore files and directories	disabled
SeSystemtimePrivilege	Change the system time	disabled
SeShutdownPrivilege	Shut down the system	disabled
SeRemoteShutdownPrivilege	Force shutdown from a remote system	disabled
SeTakeOwnershipPrivilege	Take ownership of files or other objects	disabled
SeDebugPrivilege	Debug programs	disabled
SeSystemEnvironmentPrivilege	Modify firmware environment values	disabled
SeSystemProfilePrivilege	Profile system performance	disabled
SeProfileSingleProcessPrivilege	Profile single process	disabled
SeIncreaseBasePriorityPrivilege	Increase scheduling priority	disabled
SeLoadDriverPrivilege	Load and unload device drivers	enabled
SeCreatePagefilePrivilege	Create a pagefile	disabled
SeIncreaseQuotaPrivilege	Adjust memory quotas for a process	disabled
SeUndockPrivilege	Remove computer from docking station	enabled
SeManageVolumePrivilege (*)	Perform volume maintenance tasks	disabled

(*) Not available under Windows 2000 Professional

Applications that require privileges beyond those granted to the members of the Users group usually enable them on the fly and display an error message should that fail. If an error arises, grant the required privilege(s) to the user by means of the Group Policy MMC snap-in as described above.

Appendix C: Security considerations

Synthetic SID

NeoExec adds two SIDs to the token of each instance of privileged applications: the local Administrators group (S-1-5-32-544) and a synthetic one (S-1-21-101010101-21030440) used primarily to track privileged applications. In this document we will refer to S-1-21-101010101-21030440 as the NE-SID.

The NE-SID should be used to prevent users executing privileged applications from accessing certain areas of the file system or registry. Access to these areas is governed by ACLs and one needs only to add, where necessary a deny ACE or an audit ACE.

For example, the NeoExec Professional kernel driver setup adds a deny ACE for the NE-SID on the *\SystemRoot\System32\Neo* directory to prevent end users from writing to the directory.

The setup also adds an audit ACE for the Everyone user to the *\SystemRoot\System32\Neo* directory as well as a deny ACE for NE-SID to the MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows registry key in order to prevent the user from injecting code into Privileged Applications by means of AppInit_DLLs.

Code Injection

There still exist a number of techniques for injecting code into a running process that an end user (with Debug privileges) could use to execute a portion of unauthorized code at elevated privileges. The fact that privileged applications are vulnerable to code injection does not make NeoExec less useful or less secure as, without it, end users would be running all processes with elevated privileges.

NTFS Permissions

Where Path rules are used you should ensure that regular users cannot write to privileged applications. Failing to protect privileged applications could result in end users substituting them with other applications. Privileged applications can be protected by removing the write permission on the executables.

Appendix D: Glossary of Terms

access control entry
(ACE) An entry in an access control list (ACL). An ACE contains a set of access rights and a security identifier (SID) that identifies a trustee for whom the rights are allowed, denied, or audited.

access control list
(ACL) A list of security protections that applies to an object. (An object can be a file, process, event, or anything else having a security descriptor.) An entry in an access control list (ACL) is an access control entry (ACE). There are two types of access control list, discretionary and system.

access token
An access token contains the security information for a logon session. The system creates an access token when a user logs on, and every process executed on behalf of the user has a copy of the token. The token identifies the user, the user's groups, and the user's privileges. The system uses the token to control access to securable objects and to control the ability of the user to perform various system-related operations on the local computer. There are two kinds of access token, primary and impersonation.

privilege
The right of a user to perform various system-related operations, such as shutting down the system, loading device drivers, or changing the system time. A user's access token contains a list of the privileges held by either the user or the user's groups.

process
The security context under which an application runs. Typically, the security context is associated with a user, so all applications running under a given process take on the permissions and privileges of the owning user.

security context
The security attributes or rules that are currently in effect. For example, the current user logged on to the computer or the personal identification number entered by the smart card user. For SSPI, a security context is an opaque data structure that contains security data relevant to a connection, such as a session key or an indication of the duration of the session.

security descriptor
A structure and associated data that contains the security information for a securable object. A security descriptor identifies the object's owner and primary group. It can also contain a DACL that controls access to the object, and a SACL that controls the logging of attempts to access the object.

security identifier
(SID) A data structure of variable length that identifies user, group, and computer accounts. Every account on a network is issued a unique SID when the account is first created. Internal processes in Windows refer to an account's SID rather than the account's user or group name.

Source: Microsoft Developer Network (MSDN)