

DATA ACCESS GOVERNANCE WITH NTFS PERMISSIONS

**USING PERDEMIA'S PERMISSION ANALYZER TO
GAIN INSIGHT INTO CURRENT NTFS RIGHTS
WITHIN THE NETWORK**

11 April 2021



- 1. Introduction 3
 - Information requirements 4
- 2. Standard Windows tooling 5
 - PowerShell 5
- 3. Approach with Perdemia's Permission Analyzer 7
 - Scanning the network..... 7
 - Creating reports..... 8

1. Introduction

It is common knowledge that most cases of data theft occur within the organization. No hacks from the outside, but people within the organization who (accidentally) unfairly have access to sensitive data. Apart from the financial damage, damage to reputation, this leads in many cases to legal violations, loss of intellectual property and problems in the customer relationships. See this [article](#) on [Digital Guardian](#) from this year in which 47 data security experts share their vision when it comes to protecting data and how the greatest risks come from within the organization. Employees are already cleared for most security measures and are already logged on to the network domain, they know the organization and can easily ask for additional rights. Although external attacks partly focus on weaknesses in the system, many hackers focus on existing authorized credentials through social engineering and phishing emails.

One of the complexities in the protection of data is that the solution doesn't rely on just one department. The business departments have the knowledge about the necessary authorizations and the sensitivity of data, while the IT Department has the knowledge about user groups, the rights structure and tooling. Both parties must find a balance between protection of data and supporting the business processes in which constant changes take place with accompanying pressure to ease the security principles. Who is now responsible for the protection of data and monitoring of policy?

Each organization starts with a clean design in the form of user groups, directory structures and processes, but as time passes this design blurs. Users request additional rights, projects use new approaches, departments reorganize or network data is restructured. Gradually there is a proliferation of rights and no longer can they be checked to see if the original design still applies, with all its consequences. Is the data still sufficiently protected? Do the defined groups still have the correct rights? And are temporarily acquired rights and group changes properly cleaned up? In other words, can we still rely on the design on paper with which strategic business goals, financial results, personal data, internal reviews and intellectual property are protected? We call this **Data Access Governance (DAG)**. Many companies in recent years have given priority to DAG and the protection of 'unstructured data', such as spreadsheets, presentations, documents and other data made by employees carrying out their day to day work. Gartner indicates that 80% of all data is unstructured. The first step in protecting this is the clarification, clean-up and organization of the current data and rights granted.

In this article we look at how Perdemia's Permission Analyzer can be used to support Data Access Governance and the way in which the below information requirements are addressed.

INFORMATION REQUIREMENTS

If we wish to gain insight into the current access rights, the following questions arise:

1. what are the effective rights for a specific user or group (including rights obtained through nested group memberships)?
2. where do the rights come from for a user? Through which folders and group memberships are the rights granted?
3. which groups and users have indirect access to a specific folder?
4. what happens if someone is placed in a group?
5. what are the rights for **all** users of a specific group and what exceptions apply?
6. can we validate and maintain consistency in the original security design for data access?

These information requirements increase in complexity, but even the most simple question, where someone has rights, is difficult to determine for a system administrator. The last two points are perhaps the most interesting because they contribute heavily to the validation of our security principles. We want to be able to characterize/segment users (everyone in an AD Group, a department in an LDAP organizational unit, or a private selection of employees) and we want to check the rights for a large number of users and this group. This group is allowed to view and edit certain data, or this group has absolutely **no** rights to certain data. This means that for each user the nested group memberships must be taken into account as, after all, the access rights can be obtained through nested groups. Furthermore, the overview must reveal what exactly the exceptions are and where the cause of the unwanted rights lies. In the end we want to be able to say in full confidence that **the policy for data access within the network is still applicable and consistent.**

2. Standard Windows tooling

If we rely on the standard Windows tooling then right away we encounter the first problems. NTFS rights are granted to directories and files via one of the thirteen "*special*" rights, or several "*simple*" rights which amount to a combination of special rights. The users, groups, and permissions (Access Control Entries) together constitute the Access Control List for a file or directory. Because this information is directly linked to a directory or file but not centrally stored, a network administrator will state the need to check all the directory's one by one to come to an overview of rights. In addition, a user can inherit rights through membership in groups, which makes the whole thing even more complex and time consuming to get a realistic overview. Current versions of Windows in the directory properties offer a tab to determine the effective rights of a user, but unfortunately this option is only for an individual directory. There is also a number of command-line tools available that we will evaluate later.

POWERSHELL

We have the option to use PowerShell, a scripting solution from Microsoft that enables special tasks to be performed by the operating system. One of these tasks is the Get-Acl cmdlet to get the security descriptor of a file or directory. In combination with the Get-ChildItem cmdlet to browse a folder structure, we can use a script to export all Access Control List Entries (ACE's) to a CSV file. The command powershell can be typed in the command line. PS appears before the command prompt and now the entire powershell script can be pasted and performed:

```
Get-ChildItem "C:\MyFolder" -Recurse | Sort-Object FullName | %{
$Path = $_.FullName
$IsDirectory = $_.PsIsContainer
(Get-Acl $Path) | Select-Object `
@{n='Path';e={ "$Path, d=$IsDirectory" }},
@{n='Access';e={ [String]::Join("`n", $( $_.Access | %{
"$($_.IdentityReference), $($_.AccessControlType), $($_.IsInherited),
$($_.InheritanceFlags), $($_.PropagationFlags), $($_.FileSystemRights)" )))
}}
} | Format-list | Out-File -FilePath C:\temp\permission_export.txt -
Encoding UTF8
```

A major advantage of PowerShell scripts is that the job is run on the remote server. Only the result is sent to the requesting machine through the network. We were disappointed with the performance of these tasks, scanning 500.000 local files took 32 minutes, 30% CPU, 1 GB memory and resulted in a CSV file of 150 MB. Other tools, such as Perdemia's Permission Analyzer, scans the same file information in 20 minutes as well as writing it to a database. PowerShell offers a lot of options and cmdlets to apply filters or additional tasks. For example, to skip inherited rights we can apply the line below in the preceding script:

```
@{n='Access';e={ [String]::Join("`n", $( $_.Access | ?{!$_ .IsInherited} |  
%{
```

And to exclude files from the export we can use the following:

```
Get-ChildItem "C:\MyFolder" -Recurse | Sort-Object FullName | ?{  
$_ .PsIsContainer } | %{
```

But then? Then we have all rights in a CSV file, without considering the group membership. We can of course do some basic checks and there are also ways in a PowerShell to read the Active Directory for group memberships, but even then it remains difficult to get the right information and requires substantial manual development.

If we look at the available tooling then we encounter a number of comprehensive management applications, as well as tools purely focused on reading access rights (ranging between \$200 and \$2500 USD depending on the license form) and a number of small free tools. The large packages are distinguished mainly in the extensive functionality, but do they also help us to provide the stipulated information requirements?

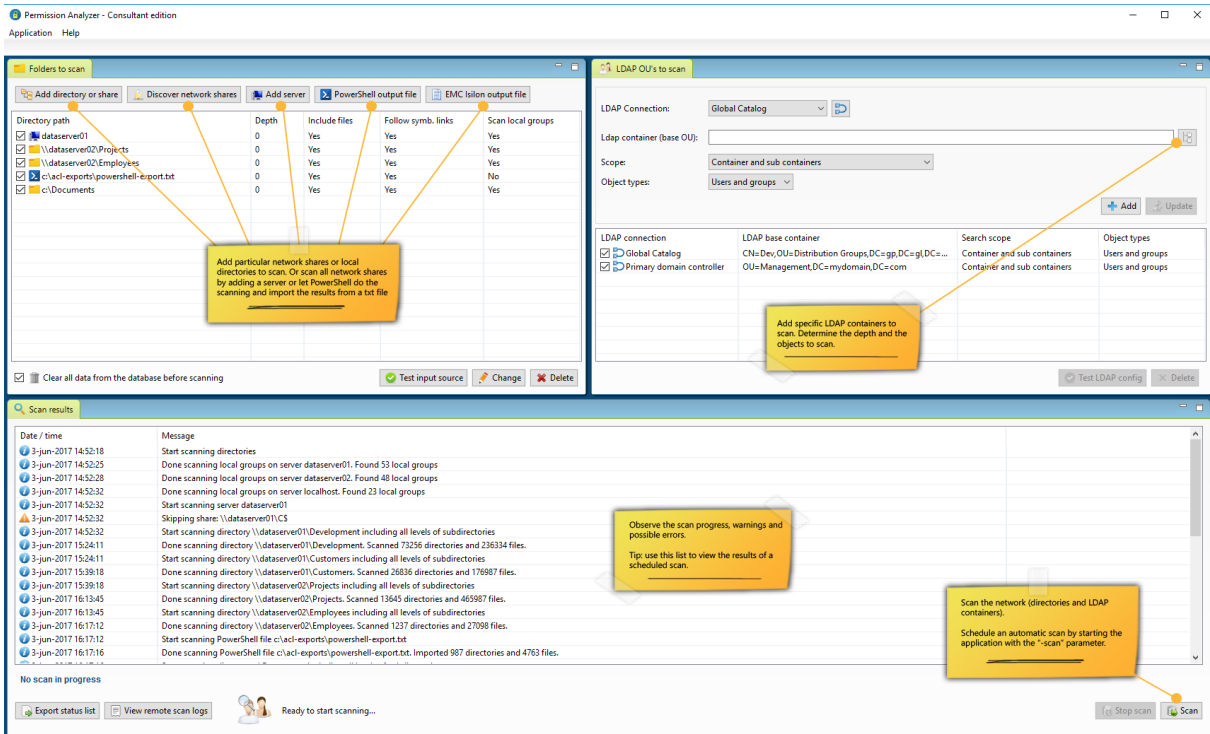
3. Approach with Perdemia's Permission Analyzer

Perdemia's Permission Analyzer is a targeted tool to gain control on the proliferation of NTFS rights. It offers a unique way to apply filters, look at results from different angles and can automatically check on the consistency of security principles by setting its own rules. Here we'll discuss some examples to see how the application works.

More than ten years ago, Perdemia's [Permission Analyzer](#) started as an application to reveal the NTFS rights per user or group. Now, the application has been completely rebuilt to version 2, in order to better fulfill information requirements stated in the [Introduction](#). Permission Analyzer was rebuilt with the focus on performance and large amounts of data. The directory information and user information is therefore not all loaded into memory and it does not scan the network for every search, but it uses a database to gather all the information together and to apply filters. It's standard to use an embedded database (H2), but out-of-box the application also supports MSSQL Server, Oracle, MySQL, DB2, PostgreSQL and Derby.

SCANNING THE NETWORK

The application provides two 'views' from the menu, the Scan View and the Report View. In the Scan View the scan is configured by selecting the desired directories and OU's from the Active Directory. The directories can be added as network share, local directory, simply a server name (a search for network shares will automatically happen), or a text/zip file generated by a Powershell script included or EMC Isilon export from a NetApp filer. Incidentally, a [Scan Agents](#) can also be used that run locally on a file server and store the scan results in a central database. The Active Directory OU's are used to scan for group memberships. A scan can also be started automatically with various [command-line parameters](#), which periodically refreshes the data.



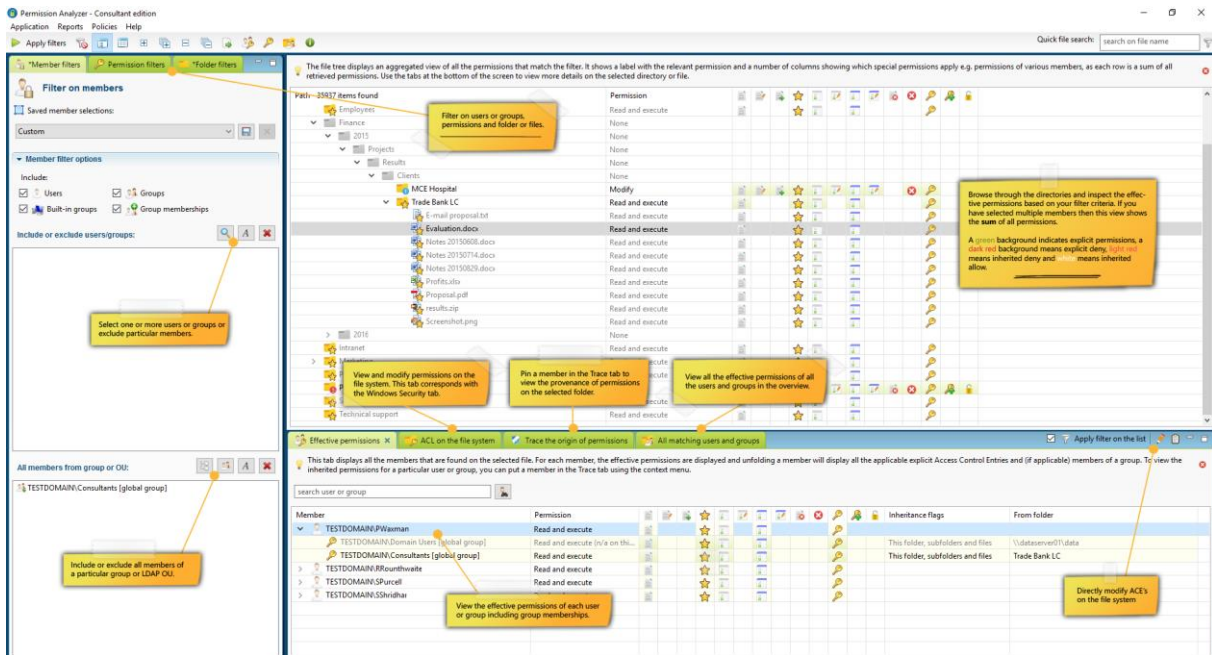
The screenshot displays the 'Permission Analyzer - Consultant edition' interface, divided into three main sections:

- Folders to scan:** A tree view on the left lists directories like 'dataserver01', 'dataserver02\Projects', and 'c:\laci-exports\powershell-export.txt'. A callout box instructs: "Add particular network shares or local directories to scan. Or scan all network shares by adding a server or let PowerShell do the scanning and import the results from a text file." Buttons for 'Add directory or share', 'Discover network shares', 'Add server', 'PowerShell output file', and 'EMC Isilon output file' are visible.
- LDAP OUs to scan:** A configuration panel on the right for 'LDAP Connection' (Global Catalog) and 'LDAP container (base OU)'. A callout box says: "Add specific LDAP containers to scan. Determine the depth and the objects to scan." The 'Object types' section lists 'Users and groups'.
- Scan results:** A log window at the bottom shows a timeline of scan activities from 3-jun-2017 14:52:18 to 16:17:16. A callout box notes: "Observe the scan progress, warnings and possible errors. Tip: use this list to view the results of a scheduled scan." Another callout box at the bottom right says: "Scan the network (directories and LDAP containers). Schedule an automatic scan by starting the application with the '-scan' parameter."

CREATING REPORTS

After the scan is complete the database can be used to make overviews. The Report View of the application displays the initial folder tree containing a summation of the rights found per directory or file. The folder icon indicates what rights are found and the columns next to a directory show the special rights. Without applying filters, the permission will mainly be set to "Full Control" with all the special rights, but after applying [filters](#) on a user or group, the unwanted rights can be viewed at a glance. In order to not clutter the folder tree the users are not instantly displayed in the folder tree, but rather across in [four tabs](#) under the folder tree

- **Effective permissions:** displays the effective rights of all users and groups in the selected folder that match the filter criteria. Each user or group can be folded out to see where the rights originate from.
- **ACL on the file system:** shows the rights (Access Control Entries) as they appear on the file system. This corresponds with the Security tab in Windows Explorer and includes the possibility to [adjust rights](#) from within the application.
- **Trace the origin of permissions:** provides the ability to create a user or group to pin down through which group memberships and parent folders the rights have been obtained.
- **All matching users and groups:** the previous tabs display information for the selected directory. This tab displays all unique users and groups that are found in the overview. Each user or group can be folded out to see where the rights are found and through which group.



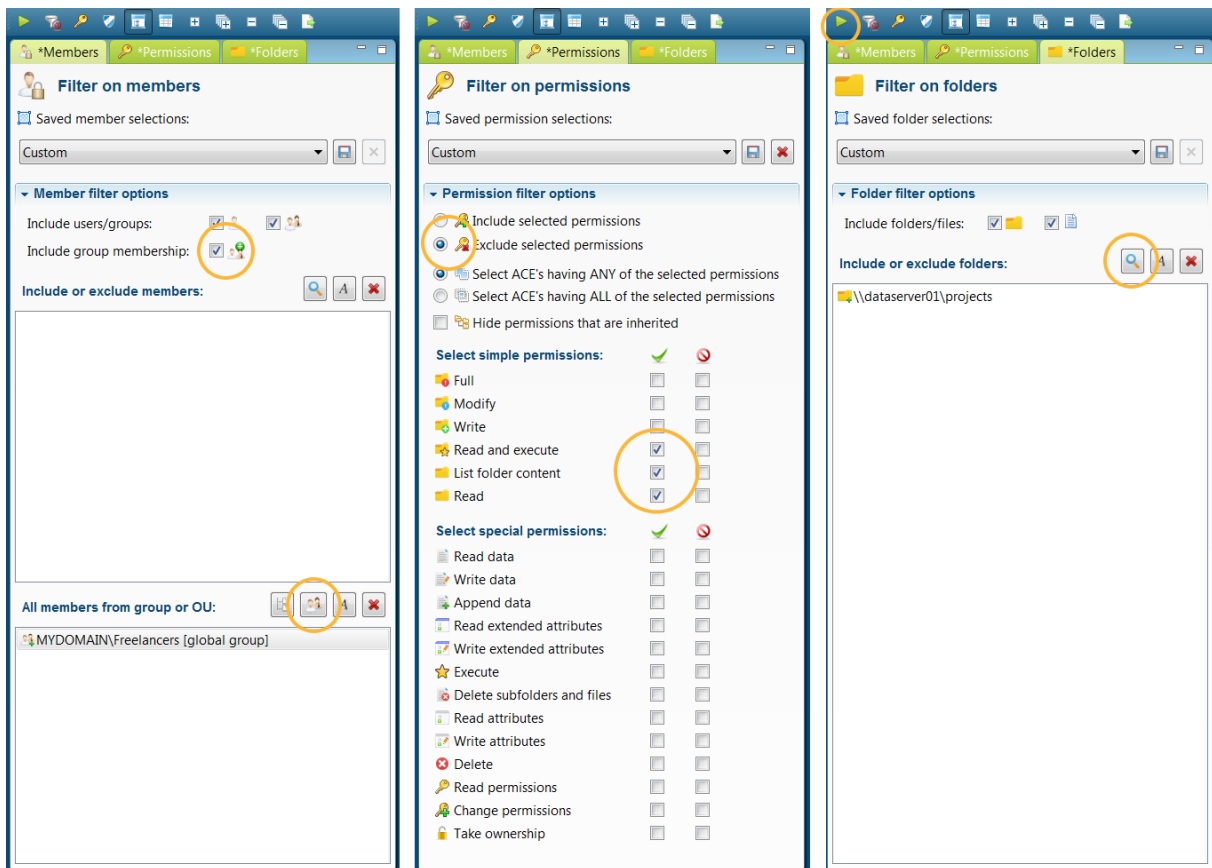
Through these four angles we can easily get answers to the first five information requirements. We can instantly view effective rights on all directories, we see per directory and user where the rights come from and we can filter out all the users of a particular group without losing the relationship between the rights and users. The latter helps us with the 5th information requirement ("What are the right for all users of a specific group and where are the exceptions?"). Finally, we want to know which users or groups are involved in the unwanted rights within the search results. By including multiple users/groups as a filter, or a filter of the "All members from group" type, in one bulk check an overview is created for a large group of users including nested group memberships. With the *Effective Permissions* and *All matching users and groups* tabs, we can subsequently zoom in on individual users.

The last information requirements is to run checks to keep the access rights consistent. Instead of notifying system administrators for each change, Permission Analyzer offers another solution. A set of filters can be stored as Policy in order to periodically check for unwanted rights. If search results are found on a Policy, the system administrator can be notified by email, with an HTML report of the rights found as an attachment. Also, the list of Policies with their status is available within the application. Loading of a Policy gives the possibility to zoom in on the results and thus solve the cause.

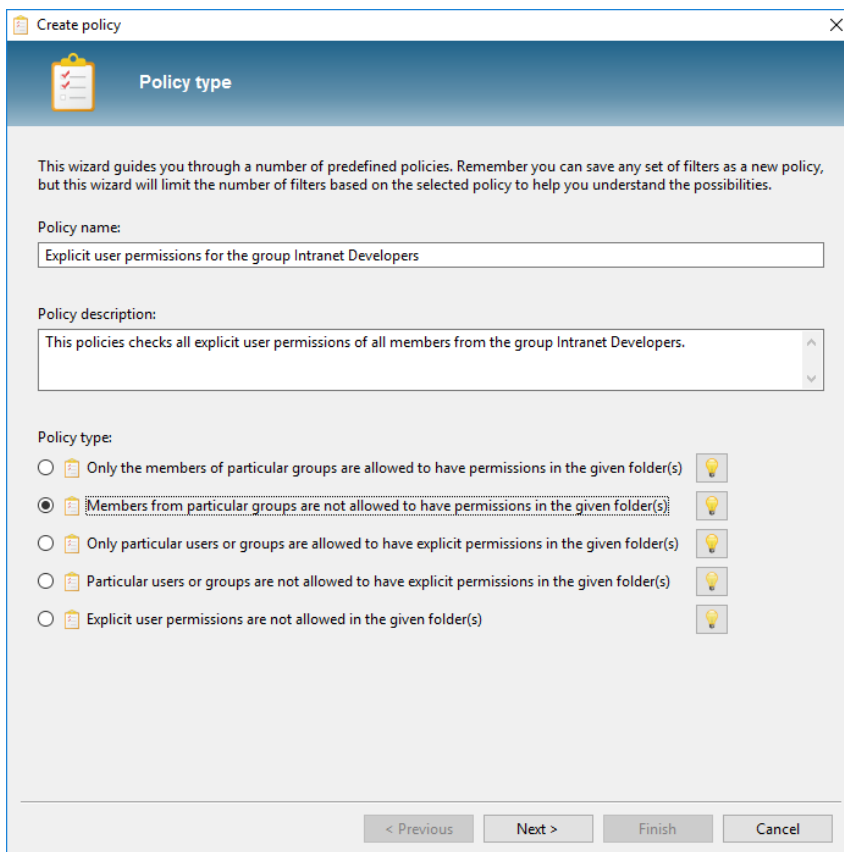
Example:

We make a policy in which it is stated: "Nobody from the group Freelancers can edit project data"

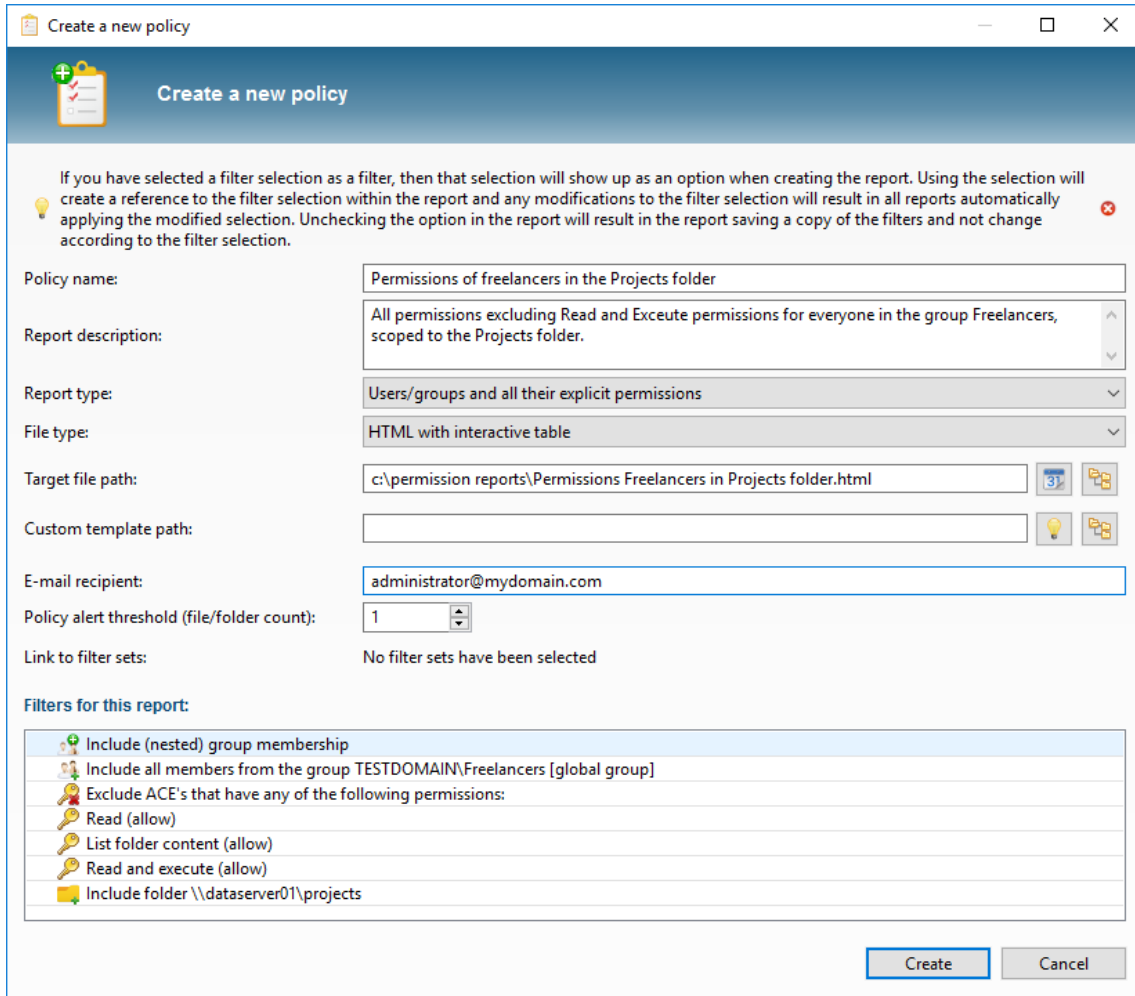
We want to create a report in which we get to see the unwanted rights so that we can save the filters as Policy. We add a member filter of the type "All members from group" and we select the group Freelancers. Then we add additional filters to the rights (we want to allow read access), and we report on the scopes folder \\\\dataserver01\\projects:



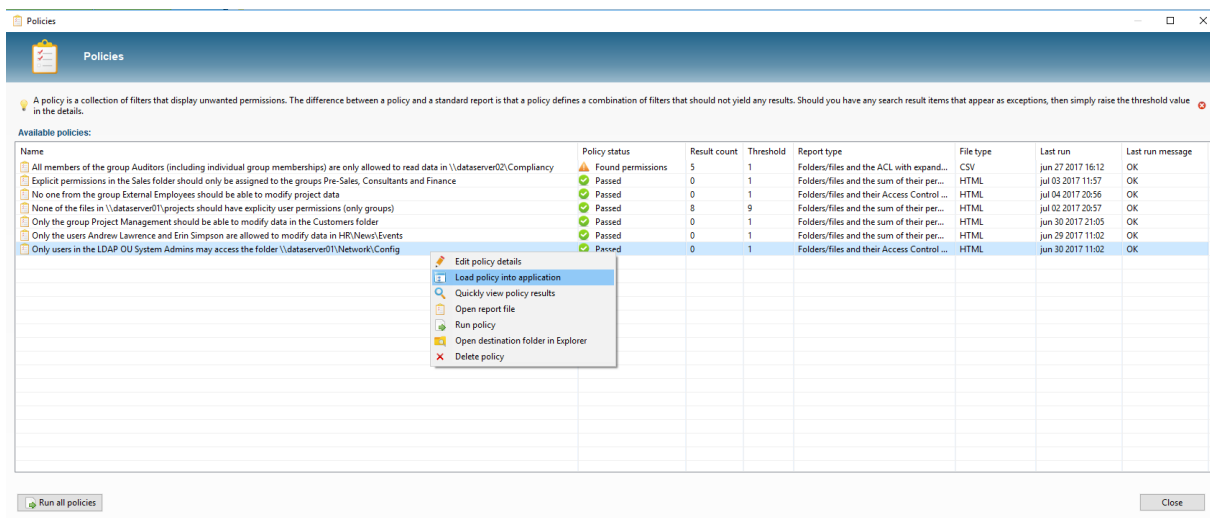
Tip: Use the Policy Wizard to get acquainted with the various filter options:



The listing now shouldn't display any search results, which means that no unwanted rights have been found. We now store the filters as Policy:



By regularly running the policies via the [parameters](#) -scan and -allPolicies, we can periodically check the rights for consistency, without having to check each change in the folder structure manually. In the overview of Policies we can see the status of our rules at a glance and we can zoom in on the unwanted rights by loading a policy in the application:



AUDIT DASHBOARD

In addition to regularly running policies we can immediately discover a number of improvements for the Data Access Governance via the Audit Permission Analyzer's dashboard. This dashboard shows 18 different charts including a top 25 of users who have the most explicit rights (rights that may have to be regulated through a group membership), a top 25 of users with the most group memberships, the ratio between direct and indirect group membership, directories with the most explicit rights and much more. Many of these statistics instantly detect where the possible risks are if someone manages to get unauthorized access to the system.



The price of Permission Analyzer depends on the number of users and groups that is scanned and varies between \$149 and \$499 per year. A license is valid for a single machine on which the application is installed (not the number of servers scanned). A trial version is available online, as well as time-limited versions of each of the Permission Analyzer editions.