VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION SYSTEMS

ACCESS CONTROL

Information Technology Security Methods

Prepared by: Simonas Riška

Checked by: lect. [redacted]

VILNIUS 2024

Practical assignment No. 1

Windows

1. **Set access permissions for users according to scenario**
    a. **Creating User Groups**

In PowerShell I created array with group names:

```
$groups = @('Systems Administrator', 'Boss', 'Administration', 'Managers',
'Unknown')
```

```
PS C:\Windows\system32> $groups = @('Systems Administrator', 'Boss', 'Administration', 'Managers', 'Unknown')
```

Then I used `foreach` loop to create groups and `net localgroup <group name> /add` command. `foreach` loop looked like this:

```
foreach ($groupName in $groups) { net localgroup $groupName /add }
```

```
PS C:\Windows\system32> foreach ($groupName in $groups) { net localgroup $groupName /add }
The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.
```

These commands created 5 groups – *Systems Administrator*, *Boss*, *Administration*, *Managers* and *Unknown*.

b. Creating User Accounts and Adding Them to Groups

I added user 'GodMode' which was created at Windows installation process to 'Systems Administrator' group with command:

```
net localgroup "Systems Administrator" GodMode /add
```

```
PS C:\Windows\system32> net localgroup "Systems Administrator" GodMode /add
The command completed successfully.
```

I decided that when creating new users they should get random temporary password that they could change after logging in. I created a function to generate random password:

```
function Generate-RandomPassword {
        $minLength = 8
        $maxLength = 14
        $length = Get-Random -Minimum $minLength -Maximum $maxLength
```

– `Get-Random` function randomly selects length between 8 and 14 characters (inclusive of 8 but exclusive of 14)

```
        $upperCase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.toCharArray()
        $lowerCase = 'abcdefghijklmnopqrstuvwxyz'.toCharArray()
        $numbers = '0123456789'.toCharArray()
        $special = '!@#$%^&*()-_=+[]{};:,.<>?'.toCharArray()
        $allCharacters = $upperCase + $lowerCase + $numbers + $special
```

`$password = @()` – initializing `$password` as an empty array

`$password += $upperCase | Get-Random` – randomly selects upper case letter and adds it to array

`$password += $numbers | Get-Random` – randomly selects number and adds it to array

`$password += $special | Get-Random` – randomly selects special character and adds it to array

`for ($i = 3; $i -lt $length; $i++) {` – `for` loop stars from 3, because it already added three characters

`$password += $allCharacters | Get-Random` – randomly selects character from all characters and adds it to array

`}`

`$shuffledPassword = ($password | Get-Random -Count $password.Length) -join ‘’` – from `$password` array it selects random values (but not repeated because of `-Count`) and joins (`-join ‘’`) it to final shuffled password. It gets repeated `$password.Length` times.

`return $shuffledPassword`

`}`

```
PS C:\Windows\system32> function Generate-RandomPassword {
>> $minLength = 8
>> $maxLength = 14
>> $length = Get-Random -Minimum $minLength -Maximum $maxLength
>> $upperCase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.toCharArray()
>> $lowerCase = 'abcdefghijklmnopqrstuvwxyz'.toCharArray()
>> $numbers = '0123456789'.toCharArray()
>> $special = '!@#$%^&*()-_=+[]{};:,.<>?'.toCharArray()
>> $allCharacters = $upperCase + $lowerCase + $numbers + $special
>> $password = @()
>> $password += $upperCase | Get-Random
>> $password += $numbers | Get-Random
>> $password += $special | Get-Random
>> for ($i = 3; $i -lt $length; $i++) {
>> $password += $allCharacters | Get-Random
>> }
>> $shuffledPassword = ($password | Get-Random -Count $password.Length) -join ''
>> return $shuffledPassword
>> }
>>
PS C:\Windows\system32> Generate-RandomPassword
P+9e^MU!(
PS C:\Windows\system32> Generate-RandomPassword
K2C.2CZ+
PS C:\Windows\system32> Generate-RandomPassword
l=O]}!4Ln
```

Also, I decided that the newly created user with temporary password should be stored in a file accessible only to ‘Systems Administrator’ group:

`$logDir = “C:\AdminLogs”`

`$logFile = “$logDir\UserPasswords.txt”`

`New-Item -Path $logDir -ItemType Directory` – creates new directory which was specified in `$logDir`.

`New-Item -Path $logFile -ItemType File` – creates new file which was specified in `$logFile`.

`icacls $logDir /grant "Systems Administrator:(OI)(CI)F" /inheritance:r` – grants `$logDir` full control `(F)` to ‘Systems Administration’ group with object and container inheritance `(OI)(CI)` that for each file and directory created inside the folder ‘Systems Administrator’ group would have full control `(F)` and remove folder inheritance from C:\ with `/inheritance:r`

icacls command structure looks like this – `icacls <directory name> <operation to perform, such as reset or grant permissions> <group/username>:<permission> <options, such as /T to apply the change to all subfolders and files recursively>`

```
PS C:\Windows\system32> $logDir = "C:\AdminLogs"
PS C:\Windows\system32> $logFile = "$logDir\UserPasswords.txt"
PS C:\Windows\system32> New-Item -Path $logDir -ItemType Directory


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/13/2024   4:44 PM                AdminLogs


PS C:\Windows\system32> New-Item -Path $logFile -ItemType File


    Directory: C:\AdminLogs


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/13/2024   4:44 PM              0 UserPasswords.txt


PS C:\Windows\system32> icacls $logDir /grant "Systems Administrator:(OI)(CI)F" /inheritance:r
processed file: C:\AdminLogs
Successfully processed 1 files; Failed processing 0 files
```

I created array with usernames of users:

`$users = @('CEO', 'Alice', 'Gabi', 'Anthony', 'Elisa', 'Jolie', 'Tom', 'Supreme')`

```
PS C:\Windows\system32> $users = @('CEO', 'Alice', 'Gabi', 'Anthony', 'Elisa', 'Jolie', 'Tom', 'Supreme')
```

Then, I used foreach loop to generate random passwords, then for each user create account with generated random passwords, force the user to change password on first login and log the username and password to the log file.

`foreach ($user in $users) {` – iterates through $users array

`$password = Generate-RandomPassword` – generates random password and saves it into $password

`net user $user $password /add` – creates new user with random password

`net user $user /logonpasswordchg:yes` – forces user to change password on first login

`Add-Content -Path $logFile -Value "$user : $password"` – adds username and password to log .txt file

`}`

```
PS C:\Windows\system32> foreach ($user in $users) {
>> $password = Generate-RandomPassword
>> net user $user $password /add
>> net user $user /logonpasswordchg:yes
>> Add-Content -Path $logFile -Value "$user : $password"
>> }
>>
The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.

The command completed successfully.
```

With **Get-Content** command I can clearly see that it worked successfully:

Get-Content -Path "C:\AdminLogs\UserPasswords.txt"

```
PS C:\Windows\system32> Get-Content -Path "C:\AdminLogs\UserPasswords.txt"
CEO : #+nYAp4DP
Alice : qj&L2A=qB_8-t
Gabi : M:H34b%C)<4
Anthony : l{ZCq1f<7G
Elisa : (x)G5DiYd
Jolie : OWN:JJF57Ng
Tom : {S#^pW1U1}OB
Supreme : QW%!]$lvZ0yCX
```

If I want to get, let's say, only Tom password, I can use command

(Get-Content -Path "C:\AdminLogs\UserPasswords.txt" | Select-String "Tom").ToString().Split(":")[1].Trim()

Get-Content -Path "C:\AdminLogs\UserPasswords.txt" reads the content of the file.
Select-String "Tom" finds the line that contains **"Tom"**.
.ToString() converts the result to a string.
.Split(":")[1] splits the line at the colon (:) and selects the second part, which is the password.
.Trim() removes any extra whitespace or newline characters.

```
PS C:\Windows\system32> (Get-Content -Path "C:\AdminLogs\UserPasswords.txt" | Select-String "Tom").ToString().Split(":")
[1].Trim()
{S#^pW1U1}OB
```

I defined a hashtable with users as keys and groups as values:

```
$userGroups = @{
‘CEO’ = ‘Boss’
‘Alice’ = ‘Administration’
‘Gabi’ = ‘Administration’
‘Anthony’ = ‘Managers’
‘Elisa’ = ‘Managers’
‘Jolie’ = ‘Managers’
‘Tom’ = ‘Managers’
‘Supreme’ = 'Unknown’
}
```

```
PS C:\Windows\system32> $userGroups = @{
>> ‘CEO’ = ‘Boss’
>> ‘Alice’ = ‘Administration’
>> ‘Gabi’ = ‘Administration’
>> ‘Anthony’ = ‘Managers’
>> ‘Elisa’ = ‘Managers’
>> ‘Jolie’ = ‘Managers’
>> ‘Tom’ = ‘Managers’
>> ‘Supreme’ = 'Unknown’
>> }
```

Then, I used foreach loop to loop through each key-value pair in the hashtable and adding to each group each member:

```
foreach ($user in $userGroups.Keys) {
    $group = $userGroups[$user]
    Add-LocalGroupMember -Group $group -Member $user
}
```

```
PS C:\Windows\system32> foreach ($user in $userGroups.Keys) {
>> $group = $userGroups[$user]
>> Add-LocalGroupMember -Group $group -Member $user
>> }
```

`With Get-LocalGroupMember -Group <group name>` I can check if users are in correct groups:

```
PS C:\Windows\system32> Get-LocalGroupMember -Group "Boss"

ObjectClass Name           PrincipalSource
----------- ----           ---------------
User        win7-task1\CEO


PS C:\Windows\system32> Get-LocalGroupMember -Group "Administration"

ObjectClass Name             PrincipalSource
----------- ----             ---------------
User        win7-task1\Alice
User        win7-task1\Gabi


PS C:\Windows\system32> Get-LocalGroupMember -Group "Managers"

ObjectClass Name               PrincipalSource
----------- ----               ---------------
User        win7-task1\Anthony
User        win7-task1\Elisa
User        win7-task1\Jolie
User        win7-task1\Tom


PS C:\Windows\system32> Get-LocalGroupMember -Group "Unknown"

ObjectClass Name               PrincipalSource
----------- ----               ---------------
User        win7-task1\Supreme
```

c. Creating directories and granting permissions

i. General directory where everyone has full permissions

I created general directory where everyone has full permissions.

```
$companyDataPath = "C:\CompanyData"
$generalDirectoryPath = "$companyDataPath\GeneralDirectory"
New-Item -Path $companyDataPath -ItemType Directory
New-Item -Path $generalDirectoryPath -ItemType Directory
icacls $generalDirectoryPath /grant "${group}:(OI)(CI)F" /T
```

```
PS C:\Windows\system32> foreach ($group in $groups) { icacls $generalDirectoryPath /grant "${group}:(OI)(CI)F" /T }
processed file: C:\CompanyData\GeneralDirectory
Successfully processed 1 files; Failed processing 0 files
processed file: C:\CompanyData\GeneralDirectory
Successfully processed 1 files; Failed processing 0 files
processed file: C:\CompanyData\GeneralDirectory
Successfully processed 1 files; Failed processing 0 files
processed file: C:\CompanyData\GeneralDirectory
Successfully processed 1 files; Failed processing 0 files
processed file: C:\CompanyData\GeneralDirectory
Successfully processed 1 files; Failed processing 0 files
```

Alternative would be `icacls $generalDirectoryPath /grant "Everyone:(OI)(CI)F" /T`

ii. Directories for each group

Managers directory

Creating special directory C:\CompanyData\ManagersDirectory:

```
$managersDirectoryPath = "$companyDataPath\ManagersDirectory"
New-Item -Path $managersDirectoryPath -ItemType Directory
```

```
PS C:\Windows\system32> $managersDirectoryPath = "$companyDataPath\ManagersDirectory"
PS C:\Windows\system32> New-Item -Path $managersDirectoryPath -ItemType Directory


    Directory: C:\CompanyData


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/13/2024   6:46 PM                ManagersDirectory
```

Rules for group:

- Each user has full permissions to the objects he creates, and for other objects – read-only permissions.
- Users from a higher-level group (in this case ‘Administration’, ‘Boss’ and ‘Systems Administrator’) can read objects.
- CEO full permissions.

To grant permissions for *ManagersDirectory*, I used these commands:

```
icacls $managersDirectoryPath /grant "CREATOR OWNER:(OI)(CI)(IO)(F)"
icacls $managersDirectoryPath /grant "Managers:(OI)(CI)(R)"
icacls $managersDirectoryPath /grant "Managers:(OI)(CI)(W)"
icacls $managersDirectoryPath /grant "Administration:(OI)(CI)(R)"
icacls $managersDirectoryPath /grant "Boss:(OI)(CI)(R)"
icacls $managersDirectoryPath /grant "Systems Administrator:(OI)(CI)(R)"
icacls $managersDirectoryPath /grant "CEO:(OI)(CI)(F)"
icacls $managersDirectoryPath /inheritance:d
```

```
icacls $managersDirectoryPath /remove:g "BUILTIN\Users"
icacls $managersDirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"
```

`icacls $managersDirectoryPath /grant "CREATOR OWNER:(OI)(CI)(IO)(F)"` – gives full permissions to the person who creates files/folders inside the directory

`icacls $managersDirectoryPath /grant "Managers:(OI)(CI)(R)"`

`icacls $managersDirectoryPath /grant "Managers:(OI)(CI)(W)"` – provides Managers group read and write access for all files and folders inside

`icacls $managersDirectoryPath /grant "Administration:(OI)(CI)(R)"`

`icacls $managersDirectoryPath /grant "Boss:(OI)(CI)(R)"`

`icacls $managersDirectoryPath /grant "Systems Administrator:(OI)(CI)(R)"` – grants Administration, Boss and Systems Administrator groups read-only access to all content inside the directory

`icacls $managersDirectoryPath /grant "CEO:(OI)(CI)(F)"` – grants Administration, Boss and Systems Administrator groups read-only access to all content inside the directory

`icacls $managersDirectoryPath /inheritance:d` – stops inherited permissions from parent directories so only explicitly set permissions apply here

`icacls $managersDirectoryPath /remove:g "BUILTIN\Users"`

`icacls $managersDirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"` – remove any access rights for general and authenticated users groups

```
PS C:\Windows\system32> icacls $managersDirectoryPath /grant "CREATOR OWNER:(OI)(CI)(IO)(F)"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /grant "Managers:(OI)(CI)(R)"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /grant "Managers:(OI)(CI)(W)"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /grant "Administration:(OI)(CI)(R)"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /grant "Systems Administrator:(OI)(CI)(R)"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /grant "CEO:(OI)(CI)(F)"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /inheritance:d
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /remove:g "BUILTIN\Users"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $managersDirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"
processed file: C:\CompanyData\ManagersDirectory
Successfully processed 1 files; Failed processing 0 files
```

Administration directory

Creating special directory C:\CompanyData\AdministrationDirectory:

```
$administrationDirectoryPath = "$companyDataPath\AdministrationDirectory"
New-Item -Path $administrationDirectoryPath -ItemType Directory
```

```
PS C:\Windows\system32> $administrationDirectoryPath = "$companyDataPath\AdministrationDirectory"
PS C:\Windows\system32> New-Item -Path $administrationDirectoryPath -ItemType Directory


    Directory: C:\CompanyData


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/13/2024  10:44 PM                AdministrationDirectory
```

Rules for group:

- Each user has full permissions to the objects he creates, and for other objects – read-only permissions.
- Users from a higher-level group (in this case ‘Boss’ and ‘Systems Administrator’) can read objects.
- CEO full permissions

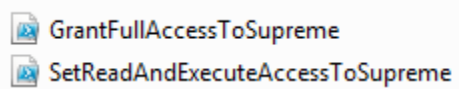
To grant permissions for *AdministrationDirectory*, I used these commands:

```
icacls $administrationDirectoryPath /grant "CREATOR OWNER:(OI)(CI)(IO)(F)"
icacls $administrationDirectoryPath /grant "Administration:(OI)(CI)(R)"
icacls $administrationDirectoryPath /grant "Administration:(OI)(CI)(W)"
icacls $administrationDirectoryPath /grant "Boss:(OI)(CI)(R)"
icacls $administrationDirectoryPath /grant "Systems Administrator:(OI)(CI)(R)"
icacls $administrationDirectoryPath /grant "CEO:(OI)(CI)(F)"
icacls $administrationDirectoryPath /inheritance:d
icacls $administrationDirectoryPath /remove:g "BUILTIN\Users"
icacls $administrationDirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"
```

```
PS C:\Windows\system32> icacls $administrationDirectoryPath /grant "CREATOR OWNER:(OI)(CI)(IO)(F)"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /grant "Administration:(OI)(CI)(R)"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /grant "Administration:(OI)(CI)(W)"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /grant "Boss:(OI)(CI)(R)"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /grant "Systems Administrator:(OI)(CI)(R)"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /grant "CEO:(OI)(CI)(F)"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /inheritance:d
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /remove:g "BUILTIN\Users"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $administrationDirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"
processed file: C:\CompanyData\AdministrationDirectory
Successfully processed 1 files; Failed processing 0 files
```

iii. Unknown complex permission scheme of my choice

For Supreme I created two files:



Both are PowerShell scripts.

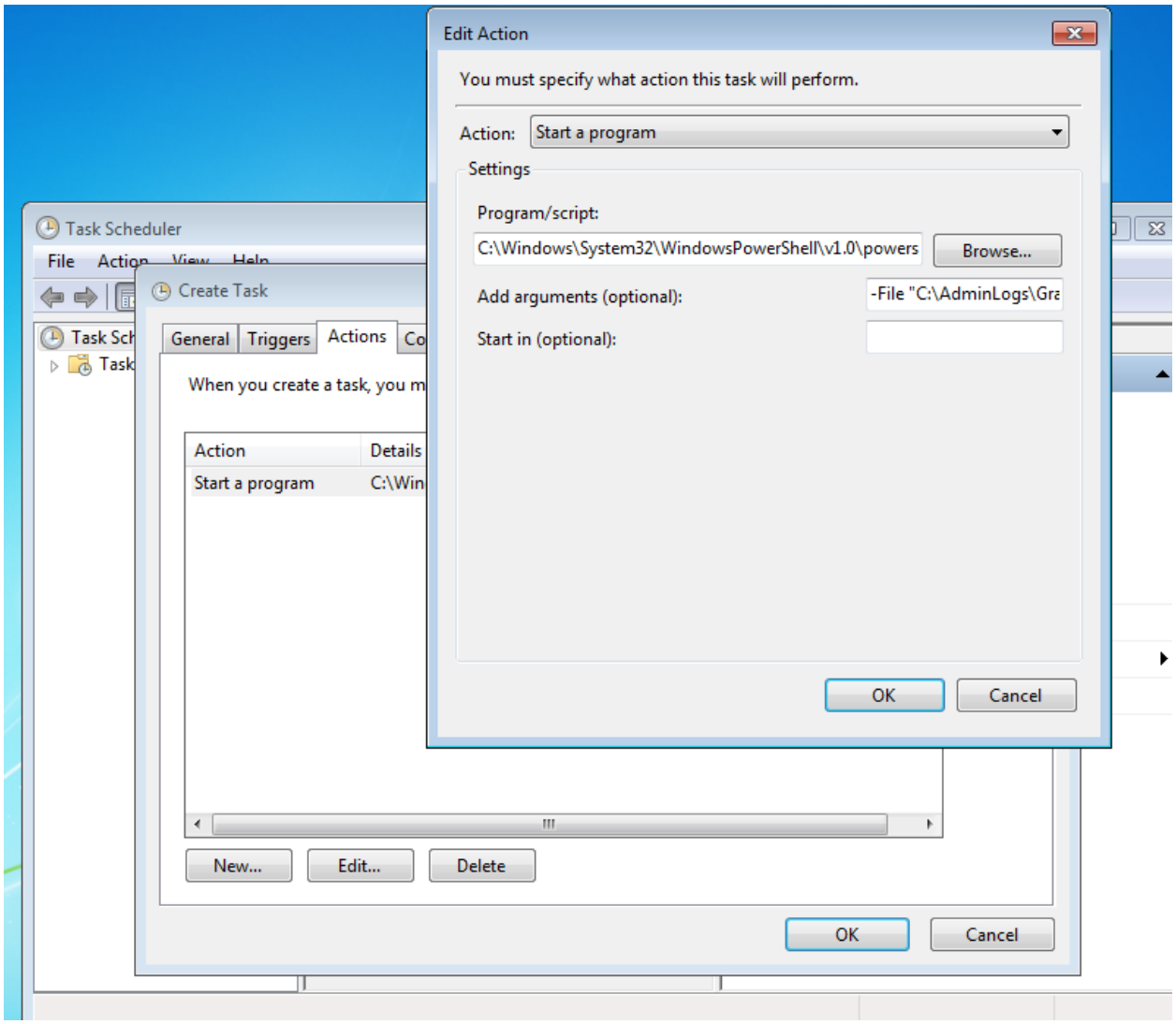
GrantFullAccessToSupreme.ps1 contains this code:

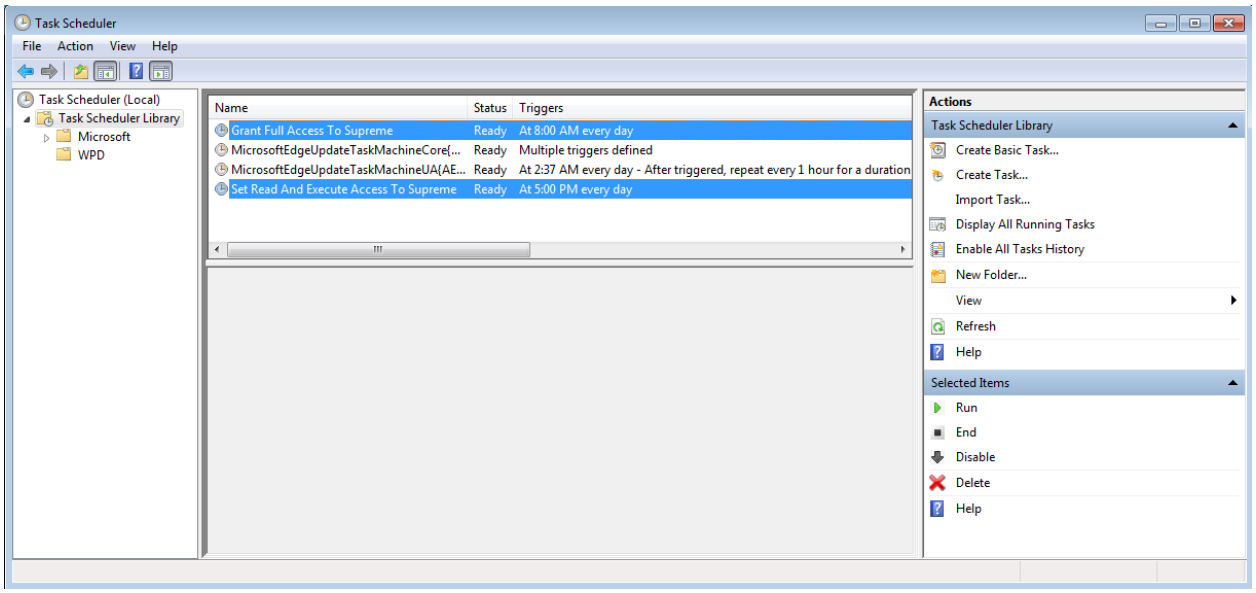
```
icacls C:\CompanyData /grant "Supreme:(OI)(CI)F" /T
```

SetReadAndExecuteAccessToSupreme.ps1 contains this code:

```
icacls C:\CompanyData /grant "Supreme:(OI)(CI)RX" /T
```

In `TaskScheduler` I made an actions that would use powershell.exe with arguments - File and those file, one at 8 a.m., another at 5 p.m.





2. **Select directory, which belongs to the CEO, and grant write permission to accountant or manager**

```
$CEODirectoryPath = "$companyDataPath\CEODirectory"
New-Item -Path $CEODirectoryPath -ItemType Directory
icacls $CEODirectoryPath /inheritance:d
icacls $CEODirectoryPath /remove:g "BUILTIN\Users"
icacls $CEODirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"
icacls $CEODirectoryPath /grant "CEO:(OI)(CI)(F)"
```

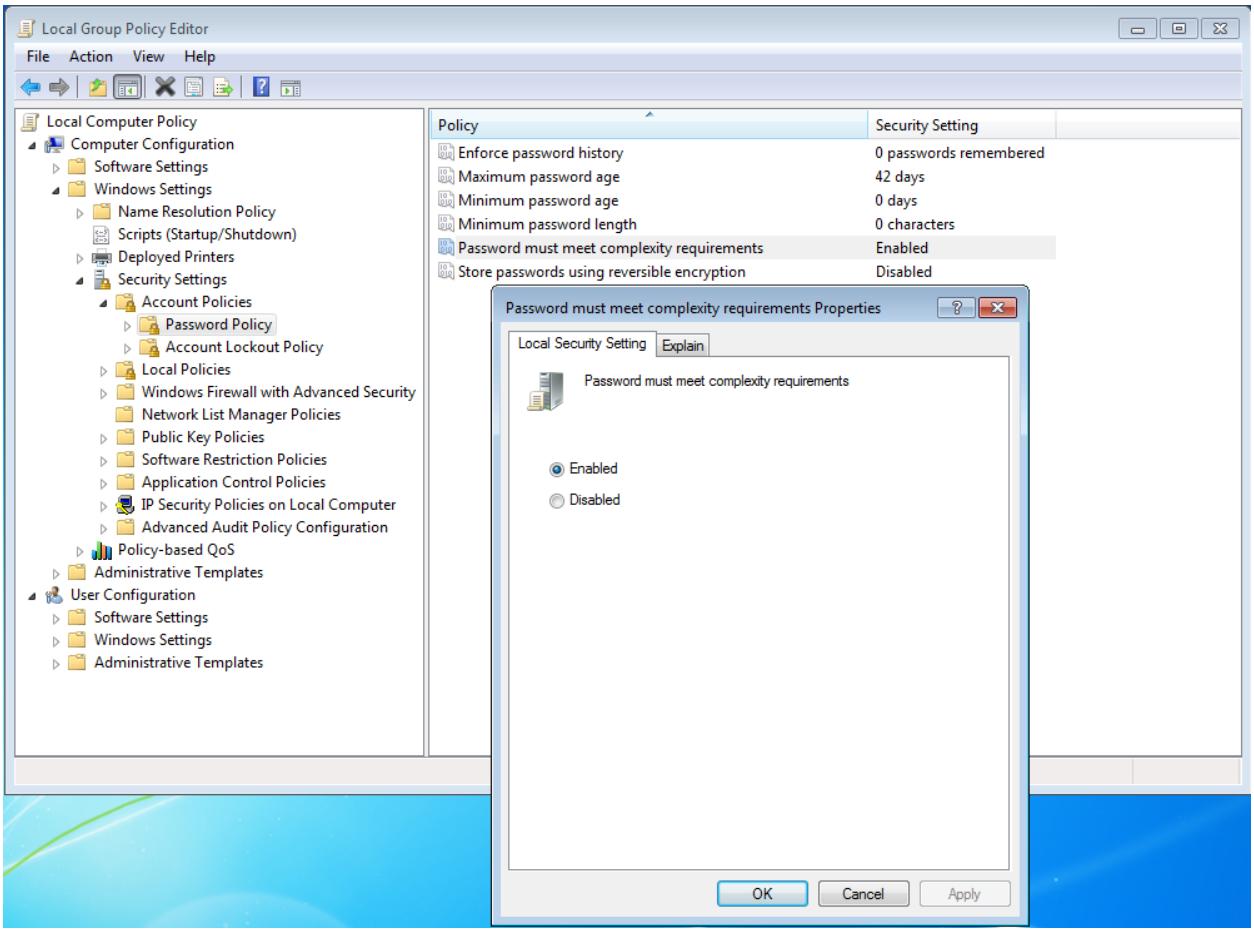
`icacls $CEODirectoryPath /grant "Alice:(CI)(OI)(IO)(W)"` – grants Alice write permissions on items within directory but not on folder itself. She won't be able to view, rename or delete CEODirectory folder, read or delete files, but she will be able to create and modify files and subfolders inside the folder.

```
PS C:\Windows\system32> $CEODirectoryPath = "$companyDataPath\CEODirectory"
PS C:\Windows\system32> New-Item -Path $CEODirectoryPath -ItemType Directory


    Directory: C:\CompanyData


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/14/2024  10:19 PM                CEODirectory


PS C:\Windows\system32> icacls $CEODirectoryPath /inheritance:d
processed file: C:\CompanyData\CEODirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $CEODirectoryPath /remove:g "BUILTIN\Users"
processed file: C:\CompanyData\CEODirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $CEODirectoryPath /remove:g "NT AUTHORITY\Authenticated Users"
processed file: C:\CompanyData\CEODirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $CEODirectoryPath /grant "CEO:(OI)(CI)(F)"
processed file: C:\CompanyData\CEODirectory
Successfully processed 1 files; Failed processing 0 files
PS C:\Windows\system32> icacls $CEODirectoryPath /grant "Alice:(CI)(OI)(IO)(W)"
processed file: C:\CompanyData\CEODirectory
Successfully processed 1 files; Failed processing 0 files
```

```
PS C:\Windows\system32> icacls $CEODirectoryPath /grant "Alice:(CI)(OI)(W)"
processed file: C:\CompanyData\CEODirectory
Successfully processed 1 files; Failed processing 0 files
```

3. **Set password policy according to current situation and modern requirements in IT world: password strength, minimum password length, maximum password age**

In Local Group Policy (**gpedit.msc**) I went to **`Computer Configuration -> Windows Settings -> Security Settings -> Account Policies -> Password Policy.`** There I enabled **`Password must meet complexity requirements`** – this enables that passwords must contain contain at least one uppercase letter, one lowercase letter, one digit, and one special character



For minimum password length, minimum/maximum password age I used following command:

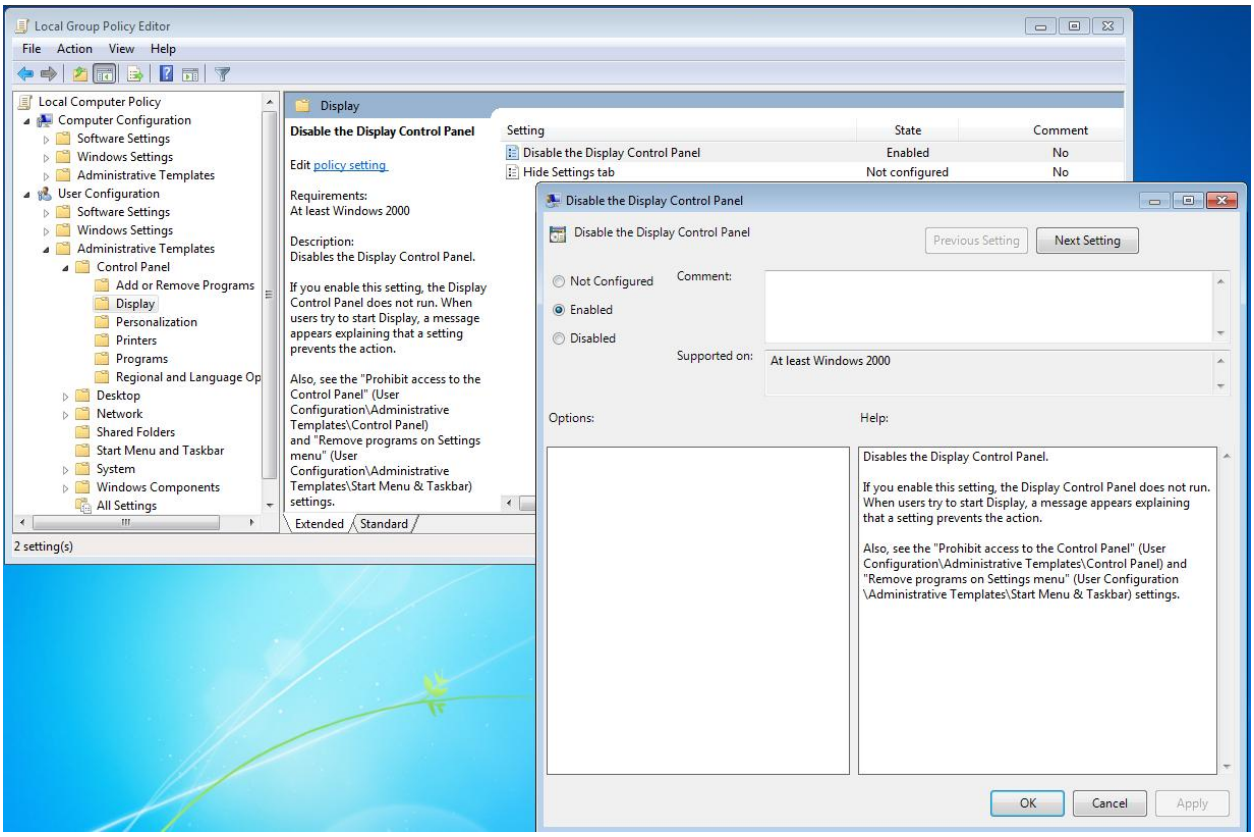
```
net accounts /minpwlen:12 /maxpwage:90 /minpwage:1
```

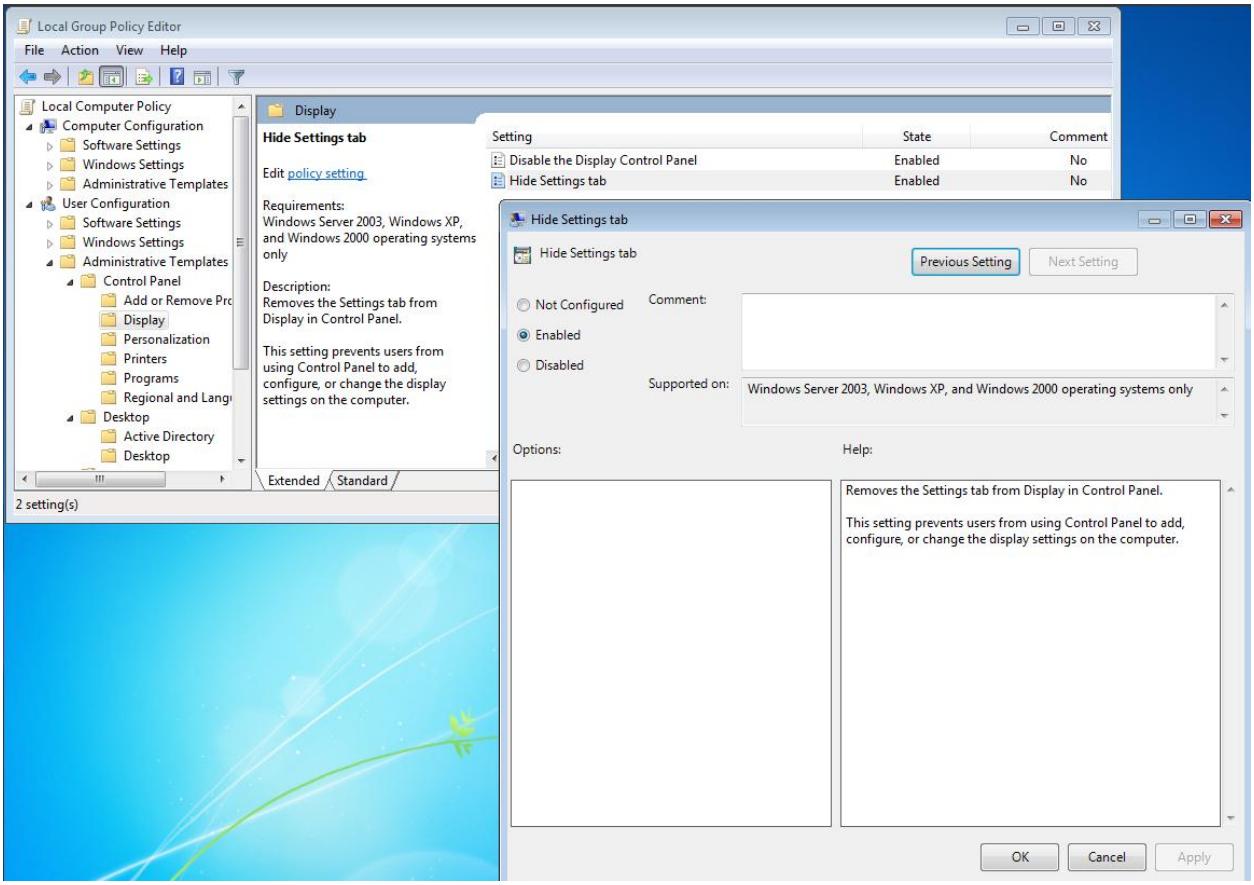
Minimum password age **`/minpwage:1`** prevents users from changing their passwords more than once a day. Maximum password age **`/maxpwage:90`** forces users to change their passwords periodically, in this case this sets the password expiration to 90 days. Minimum password length **`/minpwlen:12`** sets the minimum password length to 12 characters.

```
PS C:\Windows\system32> net accounts /minpwlen:12 /maxpwage:90 /minpwage:1
The command completed successfully.
```

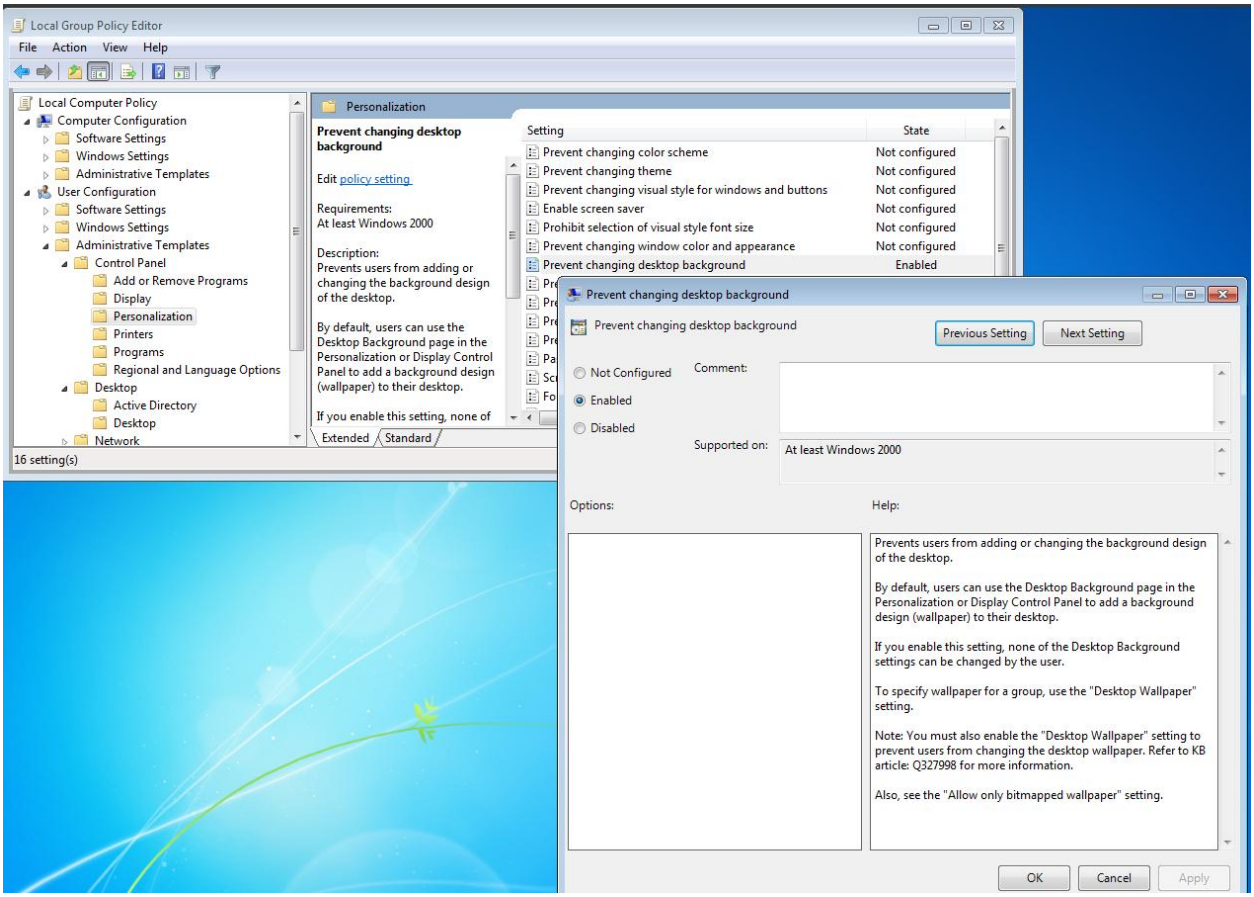
4. **Prohibit employees from customizing display settings, changing desktop wallpaper, shutting down computer, read removable drives**

In Local Group Policy (**gpedit.msc**) I went to **`User Configuration -> Administrative Templates -> Control Panel -> Display`** and there I enabled two settings – **`Disable the Display Control Panel`** and **`Hide Settings tab`**. This means users will no longer have access to the Display settings in the Control Panel or be able to see or modify display-related options within the Settings tab, effectively restricting their ability to change screen display configurations.

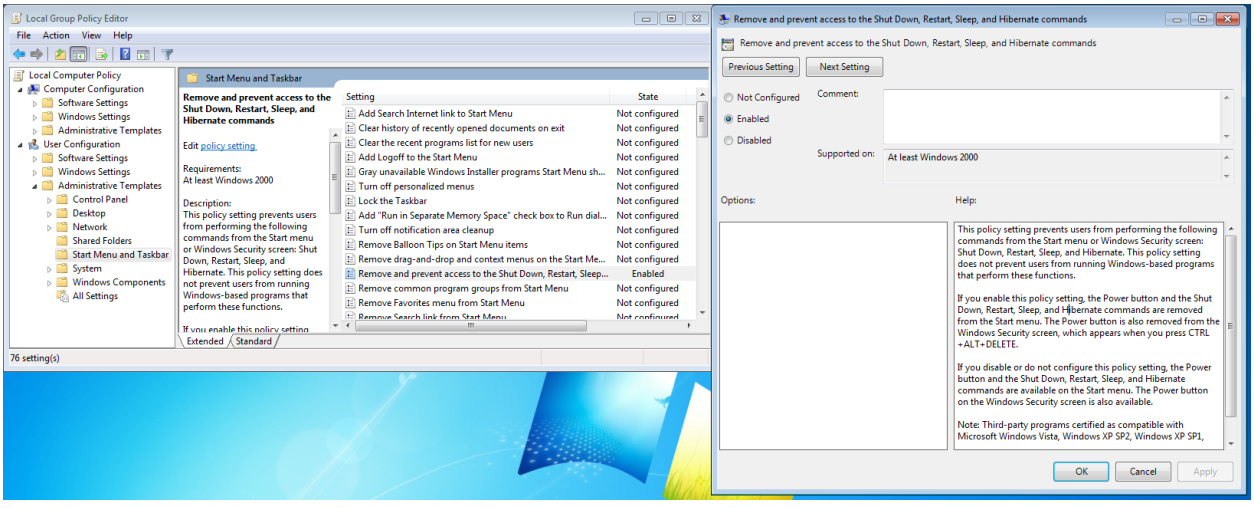

Local Group Policy Editor
File Action View Help
Local Computer Policy
Computer Configuration
Software Settings
Windows Settings
Administrative Templates
User Configuration
Software Settings
Windows Settings
Administrative Templates
Control Panel
Add or Remove Programs
Display
Personalization
Printers
Programs
Regional and Language Op
Desktop
Network
Shared Folders
Start Menu and Taskbar
System
Windows Components
All Settings
Display
Disable the Display Control Panel
Edit policy setting
Requirements:
At least Windows 2000
Description:
Disables the Display Control Panel.
If you enable this setting, the Display Control Panel does not run. When users try to start Display, a message appears explaining that a setting prevents the action.
Also, see the "Prohibit access to the Control Panel" (User Configuration\Administrative Templates\Control Panel) and "Remove programs on Settings menu" (User Configuration\Administrative Templates\Start Menu & Taskbar) settings.
Setting
State
Comment
Disable the Display Control Panel
Enabled
No
Hide Settings tab
Not configured
No
Extended
Standard
2 setting(s)
Disable the Display Control Panel
Disable the Display Control Panel
Previous Setting
Next Setting
Not Configured
Enabled
Disabled
Comment:
Supported on:
At least Windows 2000
Options:
Help:
Disables the Display Control Panel.
If you enable this setting, the Display Control Panel does not run. When users try to start Display, a message appears explaining that a setting prevents the action.
Also, see the "Prohibit access to the Control Panel" (User Configuration\Administrative Templates\Control Panel) and "Remove programs on Settings menu" (User Configuration\Administrative Templates\Start Menu & Taskbar) settings.
OK
Cancel
Apply


Local Group Policy Editor
File Action View Help
Local Computer Policy
Computer Configuration
Software Settings
Windows Settings
Administrative Templates
User Configuration
Software Settings
Windows Settings
Administrative Templates
Control Panel
Add or Remove Pro
Display
Personalization
Printers
Programs
Regional and Langu
Desktop
Active Directory
Desktop
Display
Hide Settings tab
Edit policy setting
Requirements:
Windows Server 2003, Windows XP, and Windows 2000 operating systems only
Description:
Removes the Settings tab from Display in Control Panel.
This setting prevents users from using Control Panel to add, configure, or change the display settings on the computer.
Setting
State
Comment
Disable the Display Control Panel
Enabled
No
Hide Settings tab
Enabled
No
Extended
Standard
2 setting(s)
Hide Settings tab
Hide Settings tab
Previous Setting
Next Setting
Not Configured
Enabled
Disabled
Comment:
Supported on:
Windows Server 2003, Windows XP, and Windows 2000 operating systems only
Options:
Help:
Removes the Settings tab from Display in Control Panel.
This setting prevents users from using Control Panel to add, configure, or change the display settings on the computer.
OK
Cancel
Apply

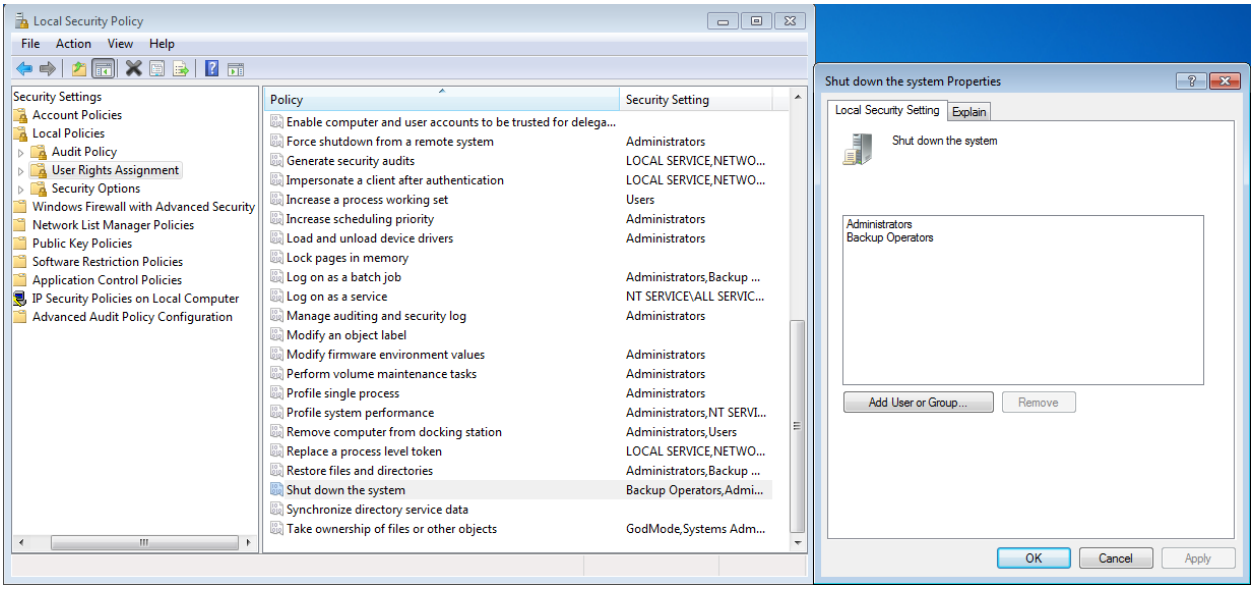
I also went to **`User Configuration -> Administrative Templates -> Control Panel -> Personalization`** and enabled setting **`Prevent changing desktop background`**.



Then I went to **`User Configuration -> Administrative Templates -> Start Menu and Taskbar`** and enabled setting **`Remove and prevent access to the Shut Down, Restart, Sleep, and Hibernate commands`**.

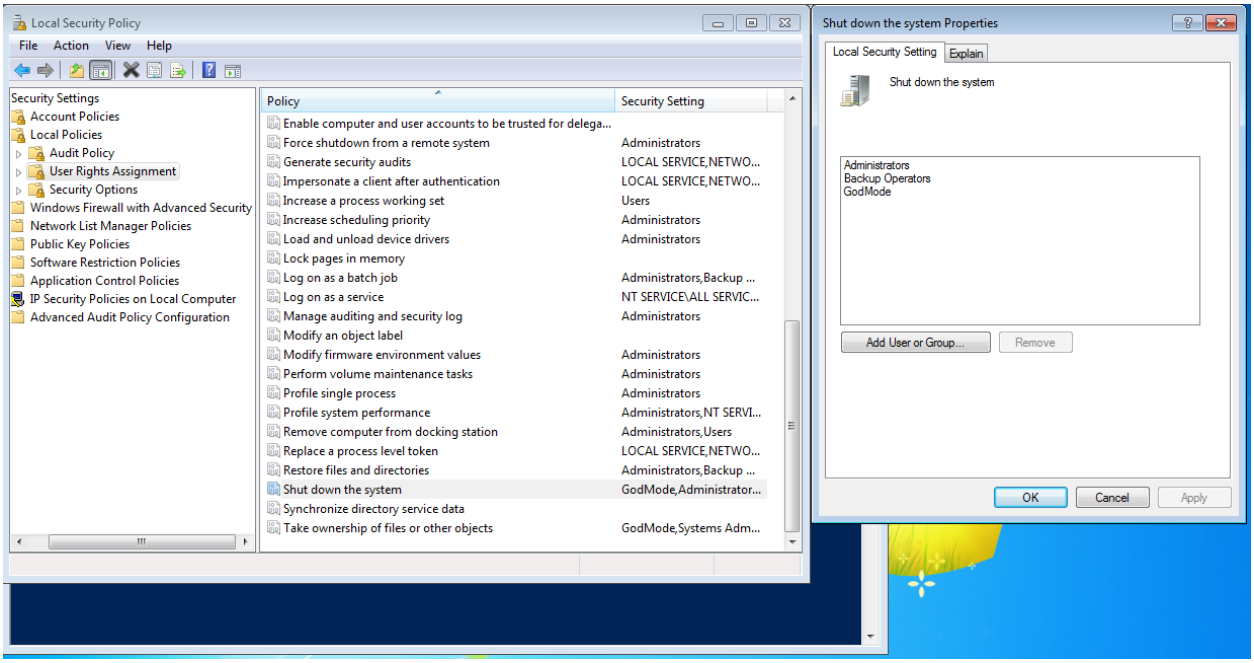


I also went to **`secpol.msc`** (Local Security Policy) **`Local Policies -> User Rights Assignment`** and removed Users group that would enable them to shut down the system (no option in start menu, but they can via Powershell with **`Stop-Computer`** cmdlet, for example)



I also tested out **`NTRIGHTS -u "GodMode" +r SeShutdownPrivilege`** command to see if that would add GodMode user to have shut down the system option (`+r` adds the right, `-r` removes).

```
PS C:\Windows\system32> NTRIGHTS -u "GodMode" +r SeShutdownPrivilege
Granting SeShutdownPrivilege to GodMode   ... successful
```



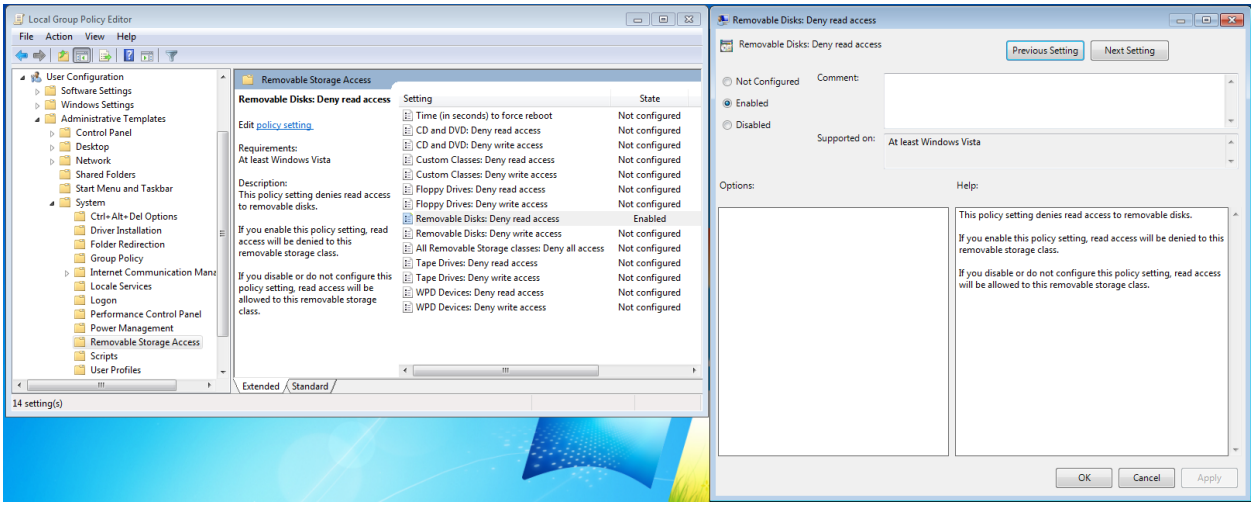
With CEO user I was unable to stop computer:

```
PS C:\Users\CEO> Stop-Computer -Force
Stop-Computer : Privilege not held.
At line:1 char:1
+ Stop-Computer -Force
+ ~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (win7-task1:String) [Stop-Computer], ManagementException
    + FullyQualifiedErrorId : StopComputerException,Microsoft.PowerShell.Commands.StopComputerCommand
```

With GodMode I was able to do that:

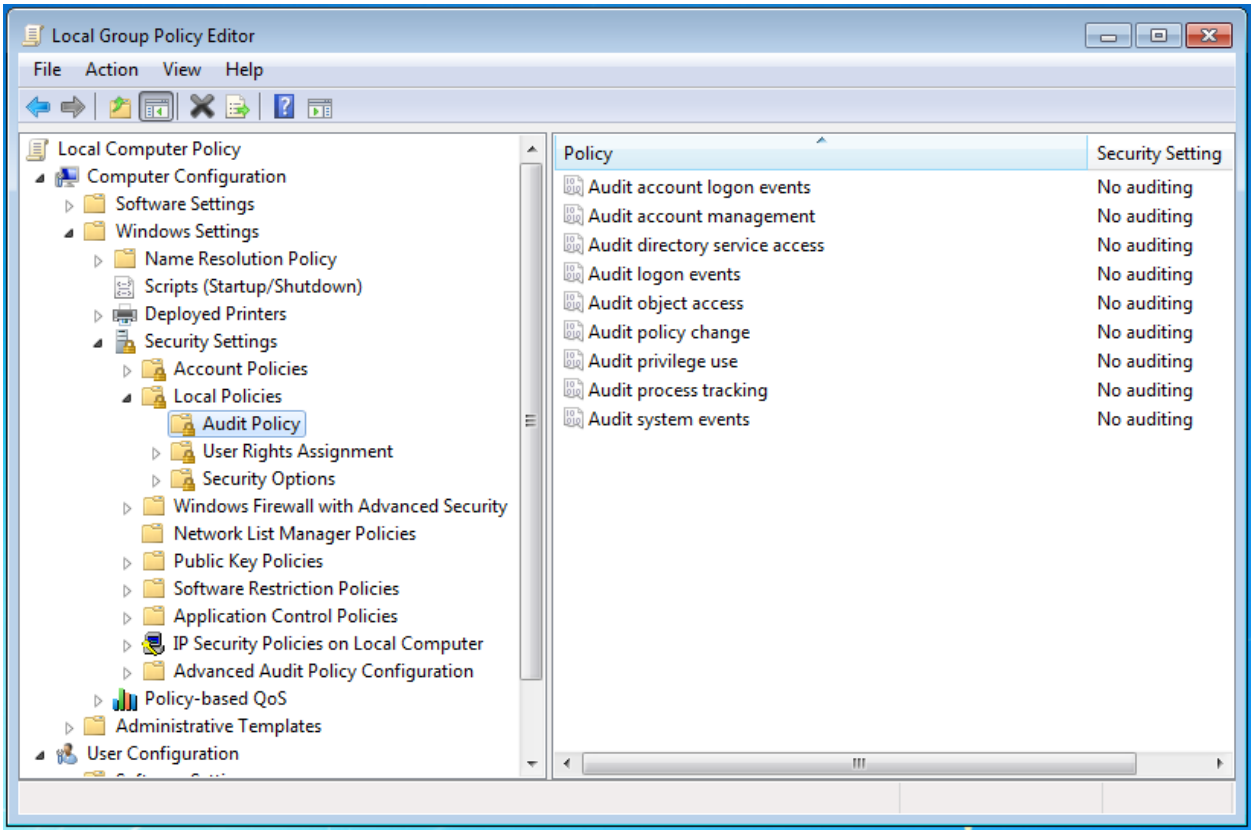
```
PS C:\Users\GodMode> Stop-Computer -Force
```

To deny read access for removable disks, I went to **`gpedit.msc`** (Local Group Policy), **`User Configuration -> Administrative Templates -> System -> Removable Storage Access`** and there I enabled setting called **`Removable Disks: Deny read access.`** That means that users will be prevented from reading or accessing any data stored on removable disks, such as USB drives, when connected to the system.

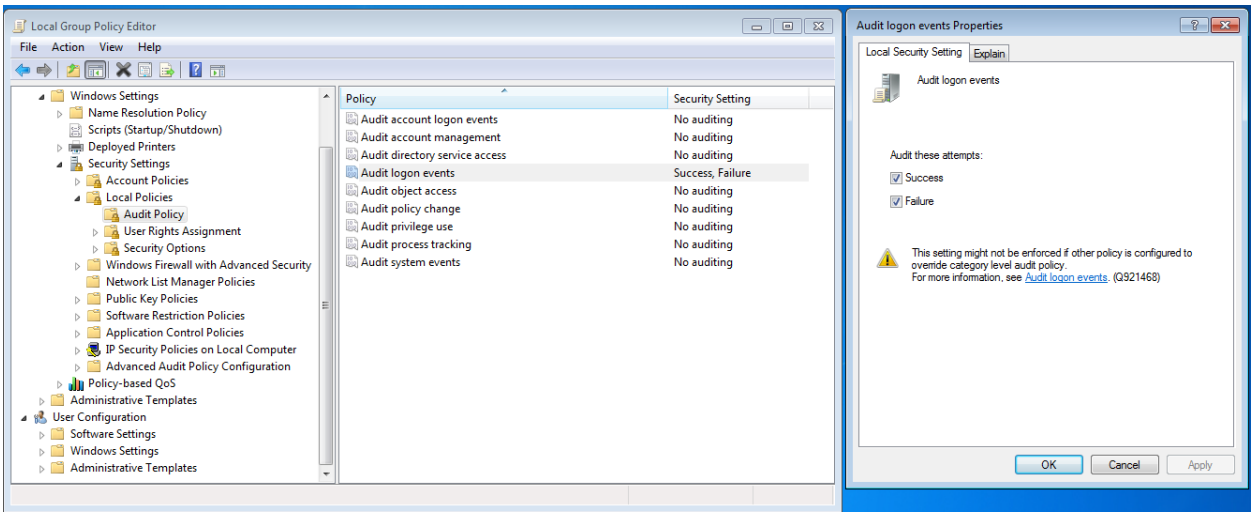


5. Activate event logging

In Local Group Policy Editor (**`gpedit.msc`**) I navigated **`Computer Configuration -> Windows Settings -> Security Settings -> Local Policies -> Audit Policy`**:



There I double-clicked on **`Audit logon events`**, selected both **`Success`** and **`Failure`** checkboxes (to audit for both successful and failed logon attempts), clicked **`Apply`** and **`OK`**:



As an alternative it could be done with **`auditpol`**:

```
auditpol /set /subcategory:"Logon" /success:enable /failure:enable
```

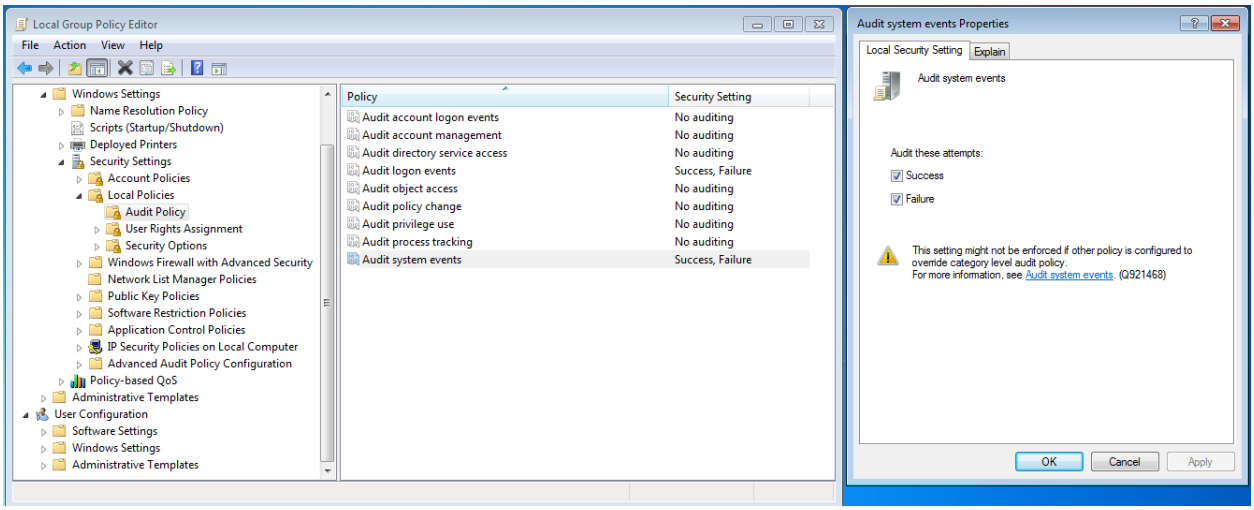
```
PS C:\Windows\system32> auditpol /set /subcategory:"Logon" /success:enable /failure:enable
The command was successfully executed.
```

And to check:

```
auditpol /get /category:"Logon/Logoff"
```

```
PS C:\Windows\system32> auditpol /get /category:"Logon/Logoff"
System audit policy
Category/Subcategory                      Setting
Logon/Logoff
  Logon                                   Success and Failure
  Logoff                                  Success
  Account Lockout                         Success
  IPsec Main Mode                         No Auditing
  IPsec Quick Mode                        No Auditing
  IPsec Extended Mode                     No Auditing
  Special Logon                           Success
  Other Logon/Logoff Events               No Auditing
  Network Policy Server                   Success and Failure
```

To activate audit system event (to enable auditing for both successful and failed system events (for example system startup, shutdown, etc)) I did the same steps for **`Audit system events`**:

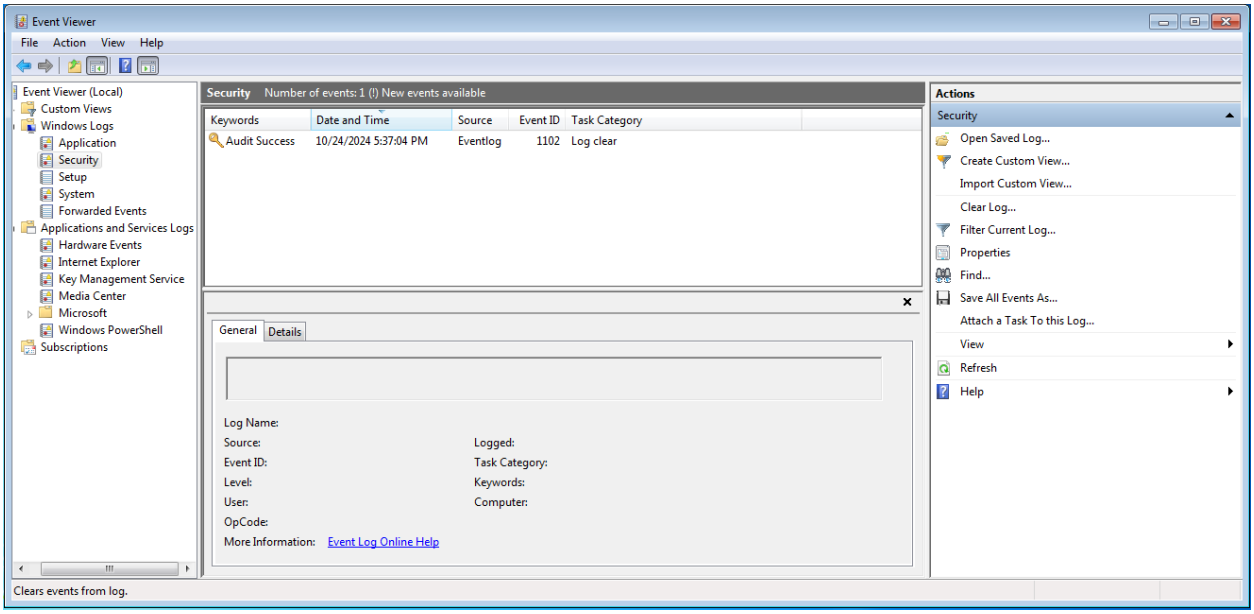


To ensure the new audit settings are applied immediately, I ran the following command in the Powershell with administrator privileges:

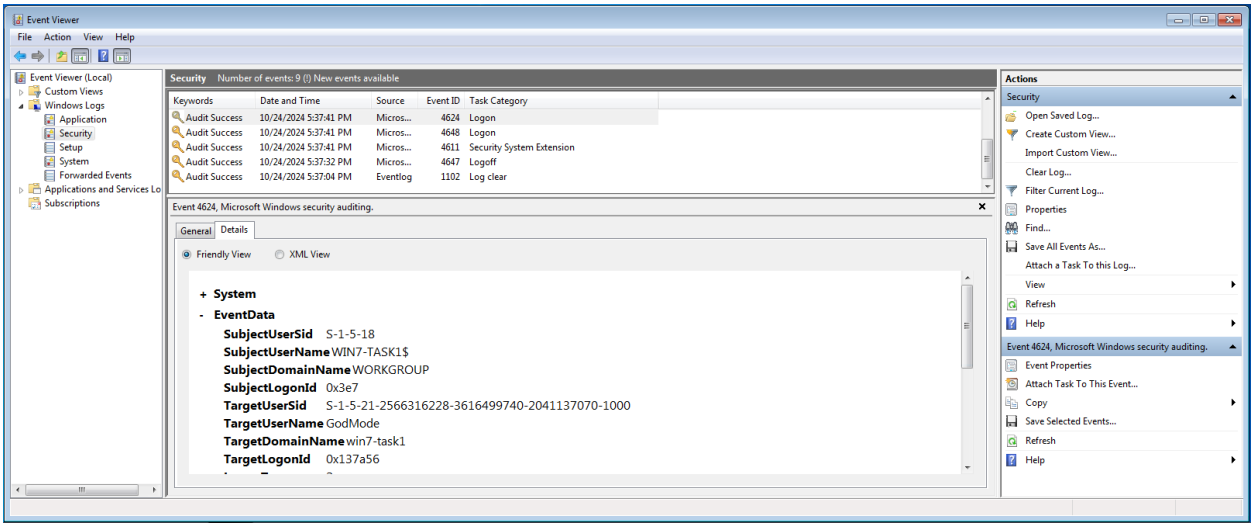
```
gpupdate /force
```

```
PS C:\Windows\system32> gpupdate /force
Updating Policy...

User Policy update has completed successfully.
Computer Policy update has completed successfully.
```

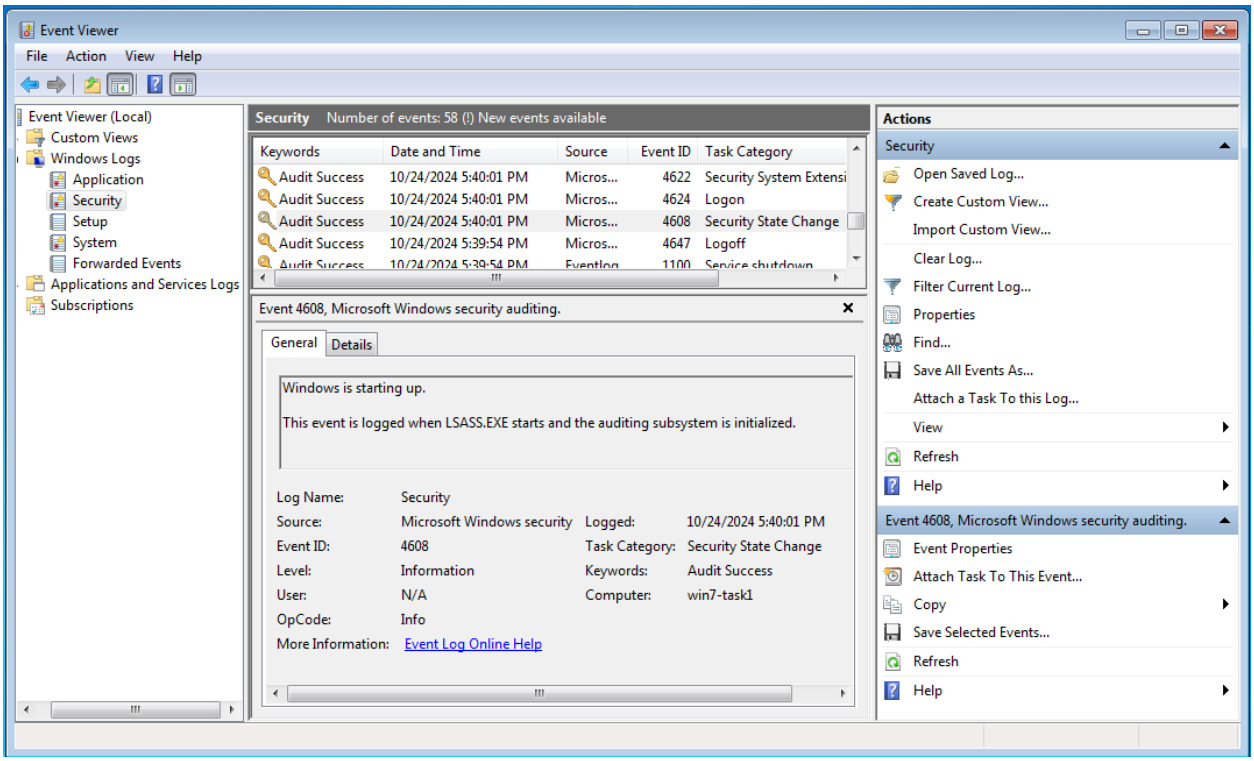
I verified that auditing was enabled by going to Event Viewer. In the Event Viewer, I navigated to **`Windows Logs -> Security`**:



I logged out of the system and logged back in to generate a logon event. I checked the Event Viewer under **`Security`** logs for a logon event (Event ID 4624 for successful logon, Event ID 4625 for failed logon):

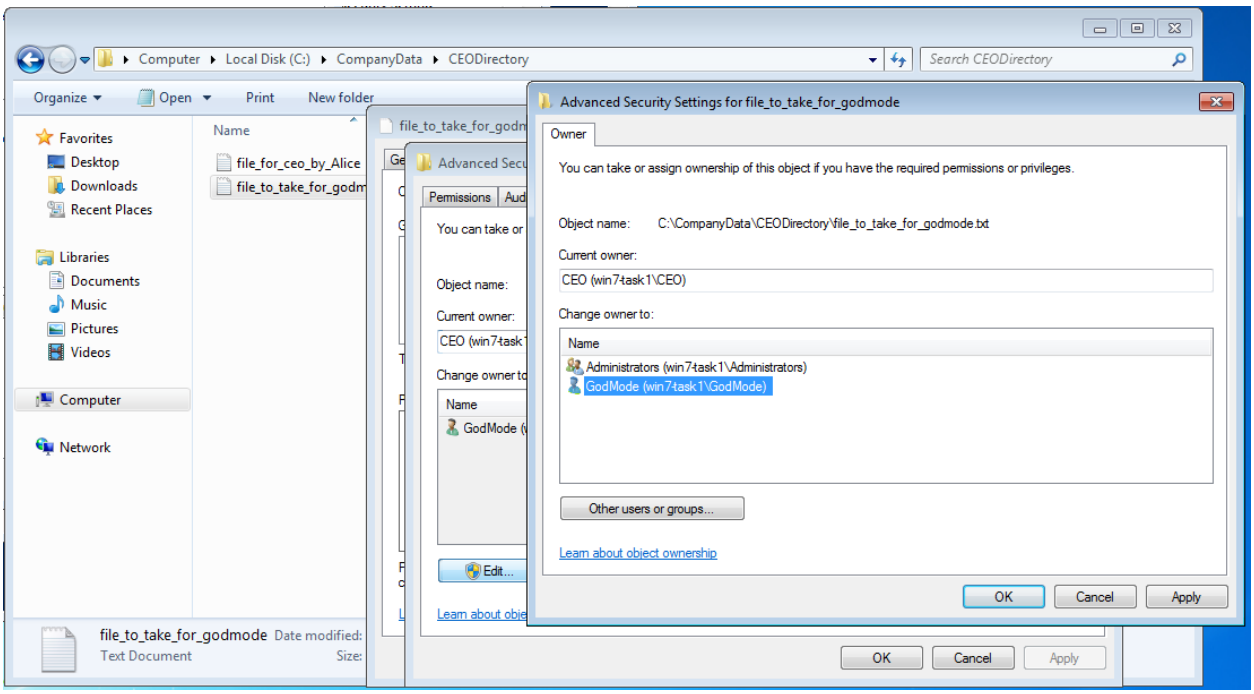


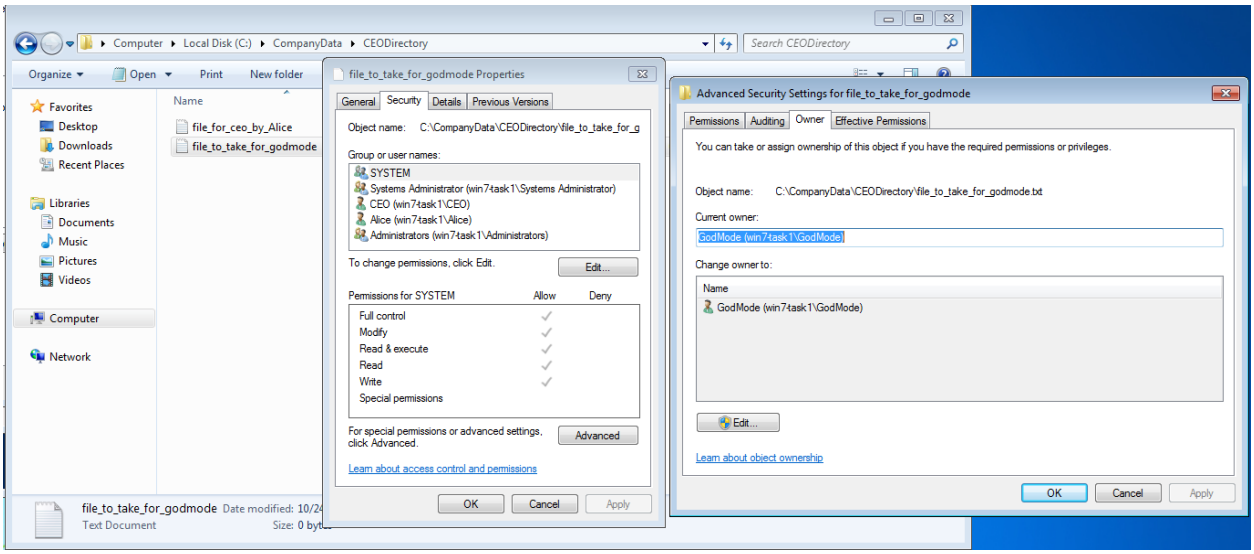
I restarted the computer to generate a system event. Then, I checked the Event Viewer under **`Security`** logs for a system event:



6. Take ownership of the selected files, which were created by other employees

With **`GodMode`** account I navigated to **`C:\CompanyData\CEODirectory`**. There I went to file properties, **`Security`** tab, pressed **`Advanced`** button and went to **`Owner`** tab. There I changed owner to **`GodMode`** account:





Also, I tried out **`Get-Acl`** command to get information about owner of **`C:\CompanyData\General Directory`**. There information showed that owner is Windows built-in **`Administrators`** group:

```
PS C:\Windows\system32> Get-Acl C:\CompanyData\GeneralDirectory

    Directory: C:\CompanyData

Path            Owner                  Access
----            -----                  ------
GeneralDirectory BUILTIN\Administrators win7-task1\Systems Administrator Allow  FullControl...
```

I tried to take ownership with this command:

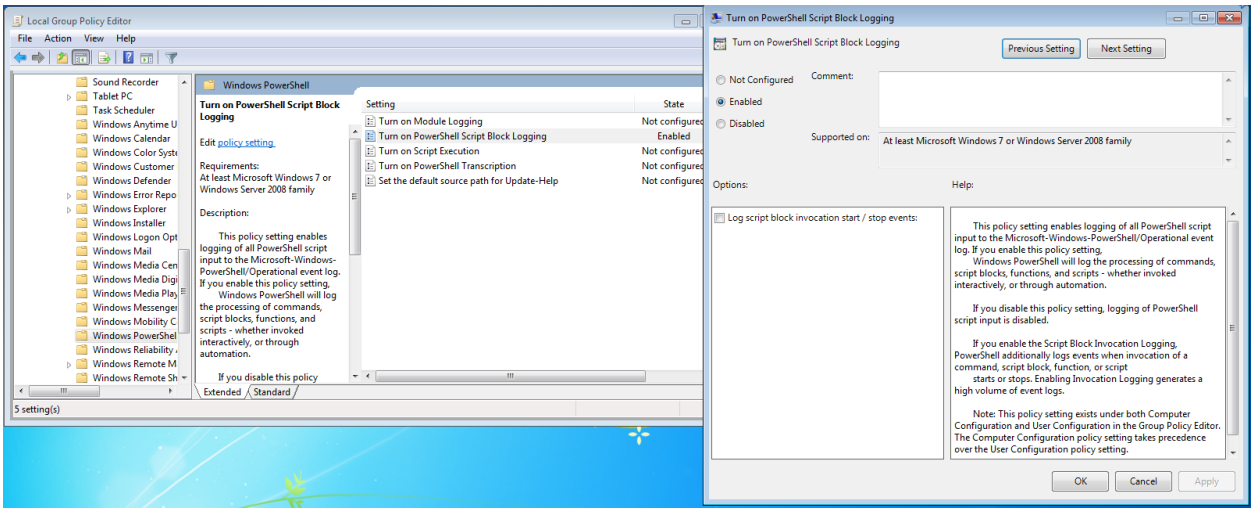
```
takeown /f "C:\CompanyData\GeneralDirectory" /r /d y
```

In this command **`/f`** specifies the filename or directory path to take ownership of, in this case – **`C:\CompanyData\GeneralDirectory`**. **`/r`** makes the command recursive, applying it to the specified folder and all files and subfolders within in. **`/d y`** automatically responds "Yes" to any confirmation prompts if the command encounters files that require confirmation to take ownership.

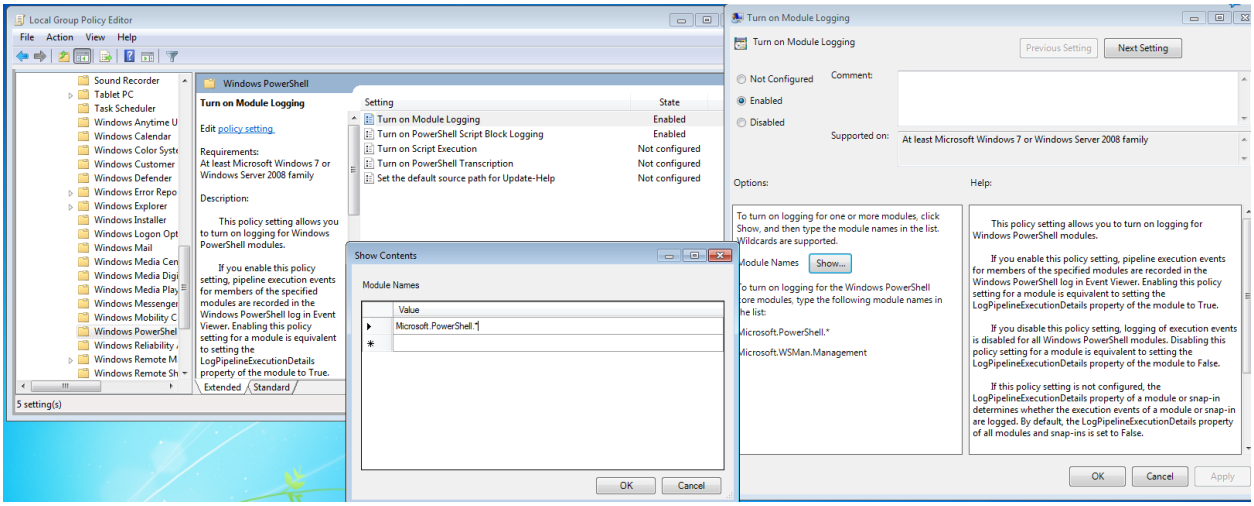
```
PS C:\Users\GodMode> takeown /f "C:\CompanyData\GeneralDirectory" /r /d y

SUCCESS: The file (or folder): "C:\CompanyData\GeneralDirectory" now owned by user "win7-task1\GodMode".

SUCCESS: The file (or folder): "C:\CompanyData\GeneralDirectory\AnthonyFileGeneral.txt" now owned by user "win7-task1\Go
dMode".
PS C:\Users\GodMode> Get-Acl C:\CompanyData\GeneralDirectory


    Directory: C:\CompanyData

Path            Owner              Access
----            -----              ------
GeneralDirectory win7-task1\GodMode win7-task1\Systems Administrator Allow  FullControl...
```

7. Implement Logging and Sessions recording for Privileged users

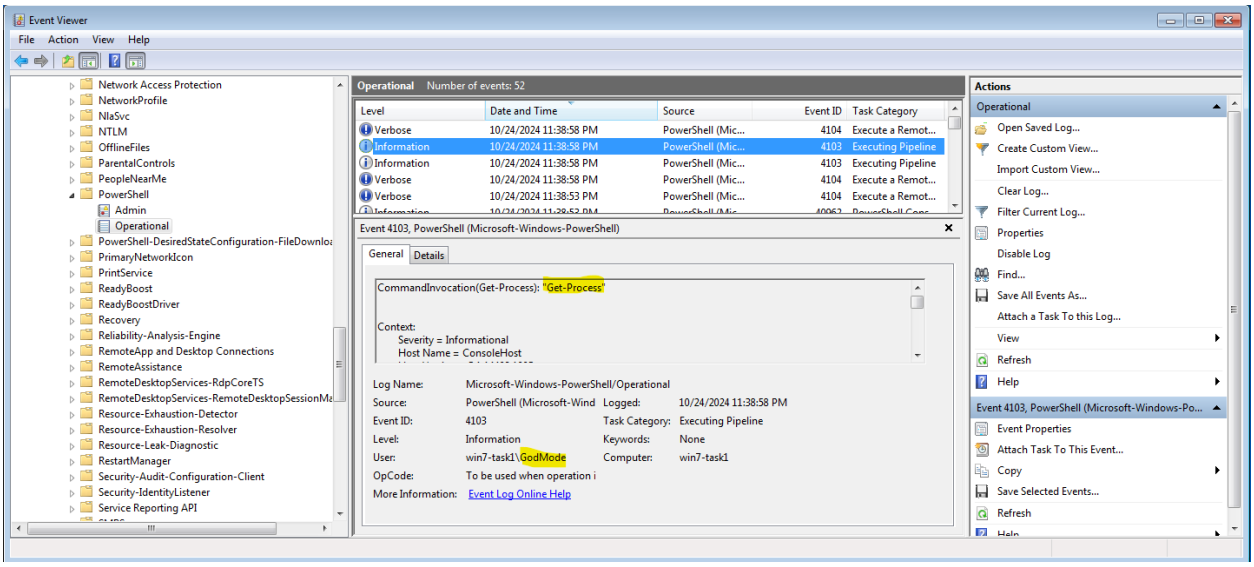
I opened Local Group Policy Editor (**`gpedit.msc`**) and navigated to **`Computer Configuration -> Administrative Templates -> Windows Components -> Windows PowerShell`**. There I double-clicked on **`Turn on PowerShell Script Block Logging`**. In the window that opened I selected **`Enabled`** and applied the setting. That ensures that PowerShell commands are logged in the event logs.



Still in the same **`Windows PowerShell`** section of the Local Group Policy Editor I double-clicked **`Turn on Module Logging`**, selected **`Enabled`**, In the **`Modules`** names, I entered an Microsoft.PowerShell with asterisk (*) (**`Microsoft.PowerShell.*`**) to log all modules. This logs PowerShell commands executed via modules.

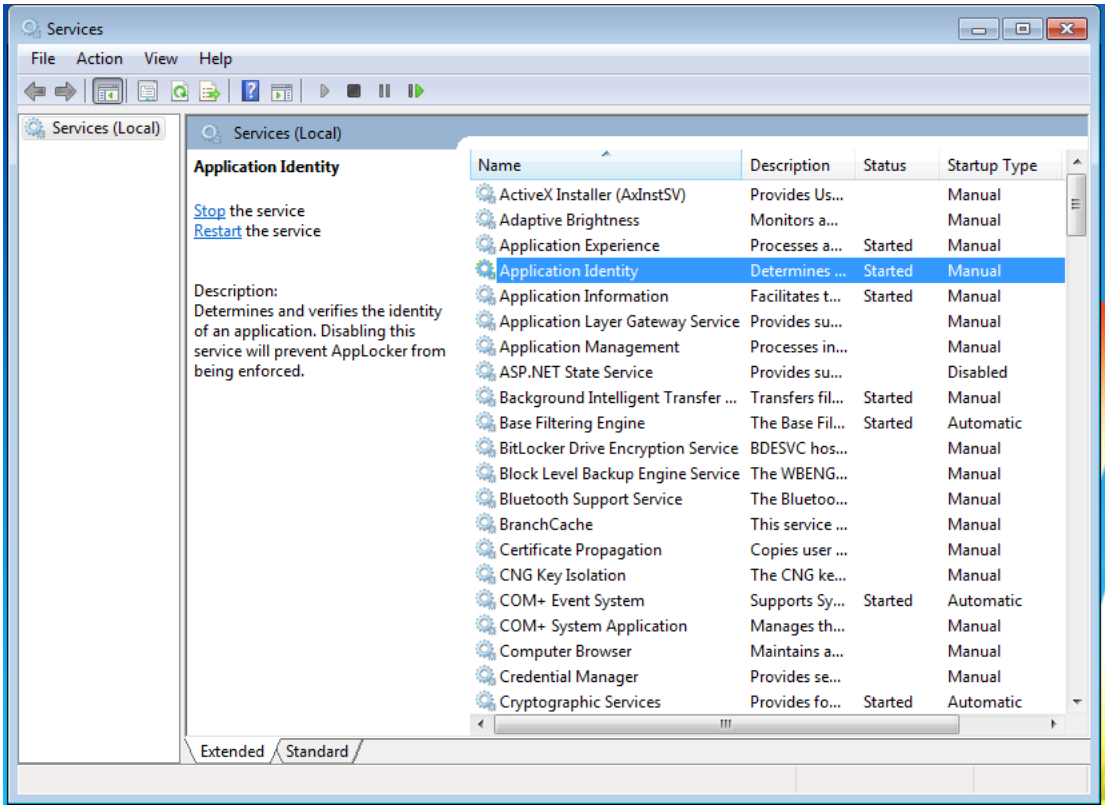


After applying these settings, logs of PowerShell sessions and any commands executed with administrator privileges can be viewed. I open PowerShell with administrator privileges and run some commands like **`Get-Process`**. Then, I check the **`Event Viewer`** and navigated to **`Applications and Services Logs -> Microsoft -> Windows -> PowerShell > Operational`** to see if the commands were logged in the **`PowerShell Operational`** log:

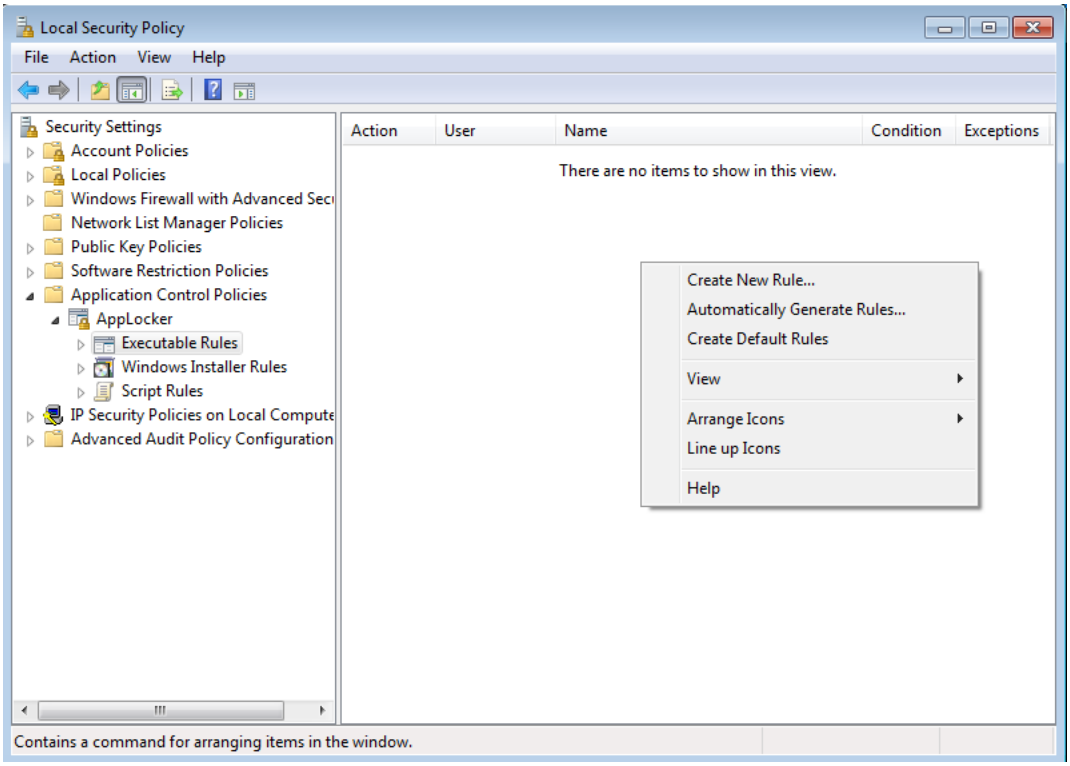


8. Limit user account with Applocker

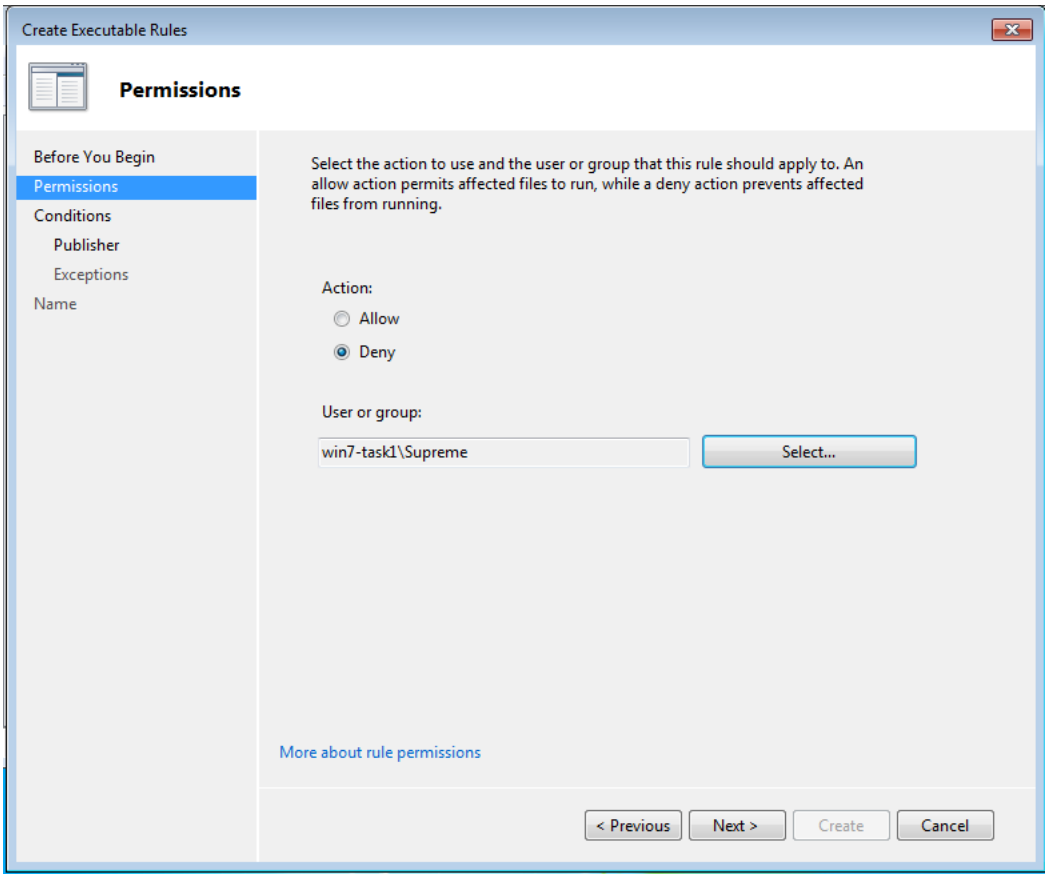
First I went to **`services.msc`**, found **`Application Identity`** in the list, right-clicked it, and selected **`Start`**. This is required for **`AppLocker`** to work:



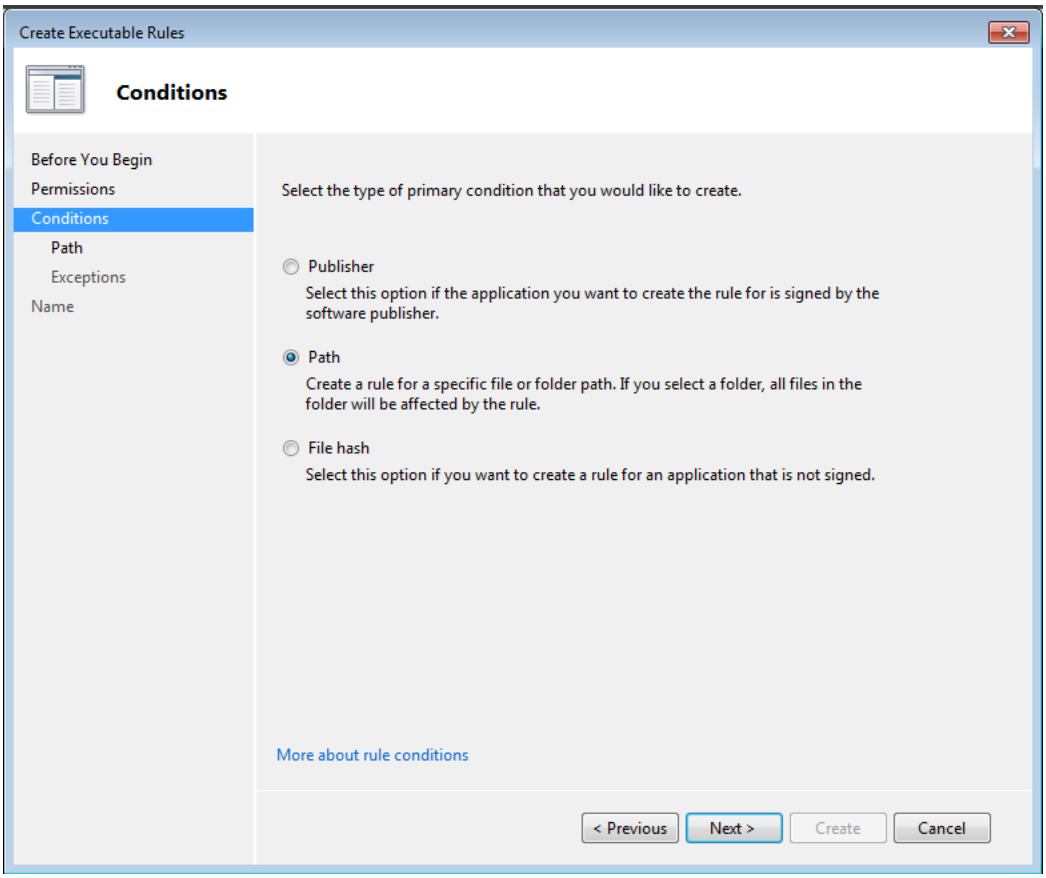
Then I opened **`secpol.msc`** (Local Security Policy). In the **`Application Control Policies -> AppLocker`** section, I right-clicked in **`Executable Rules`** and select **`Create New Rule`**:



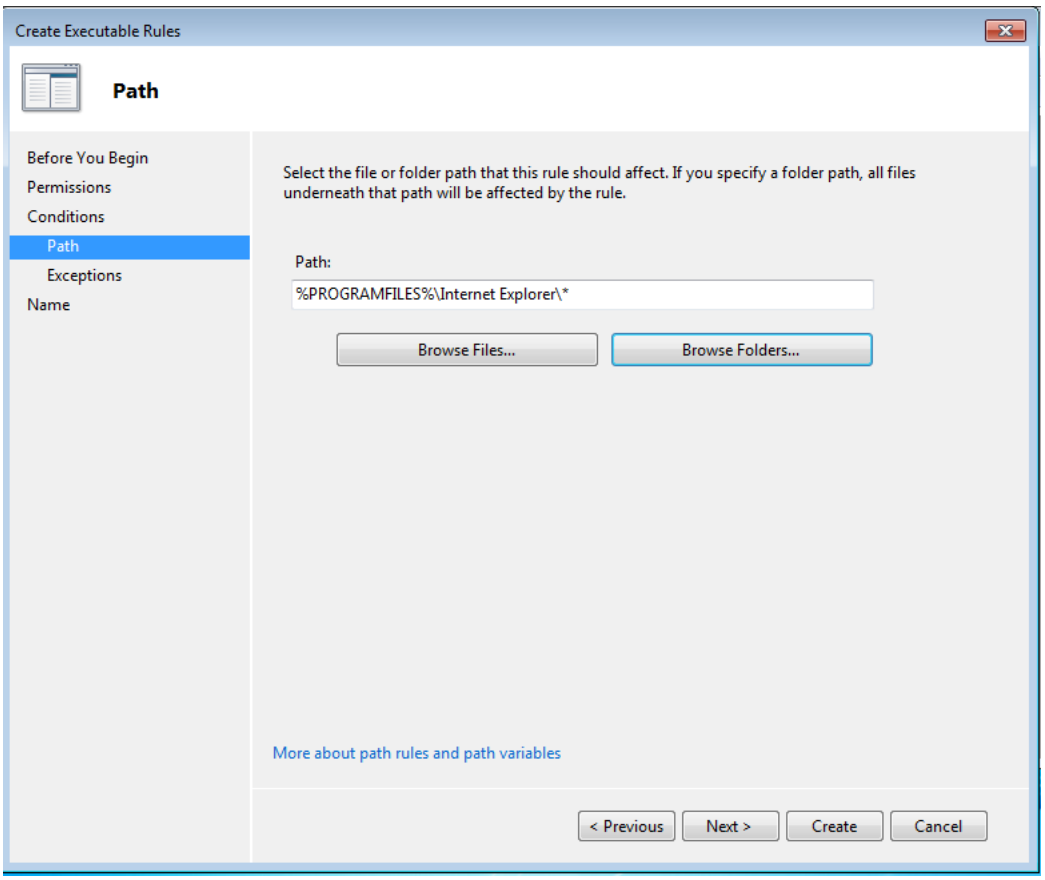
I chose **`Deny`** under **`Action`** - this prevents the user from running specific applications. Then I decided that I will limit **`Supreme`** user account:



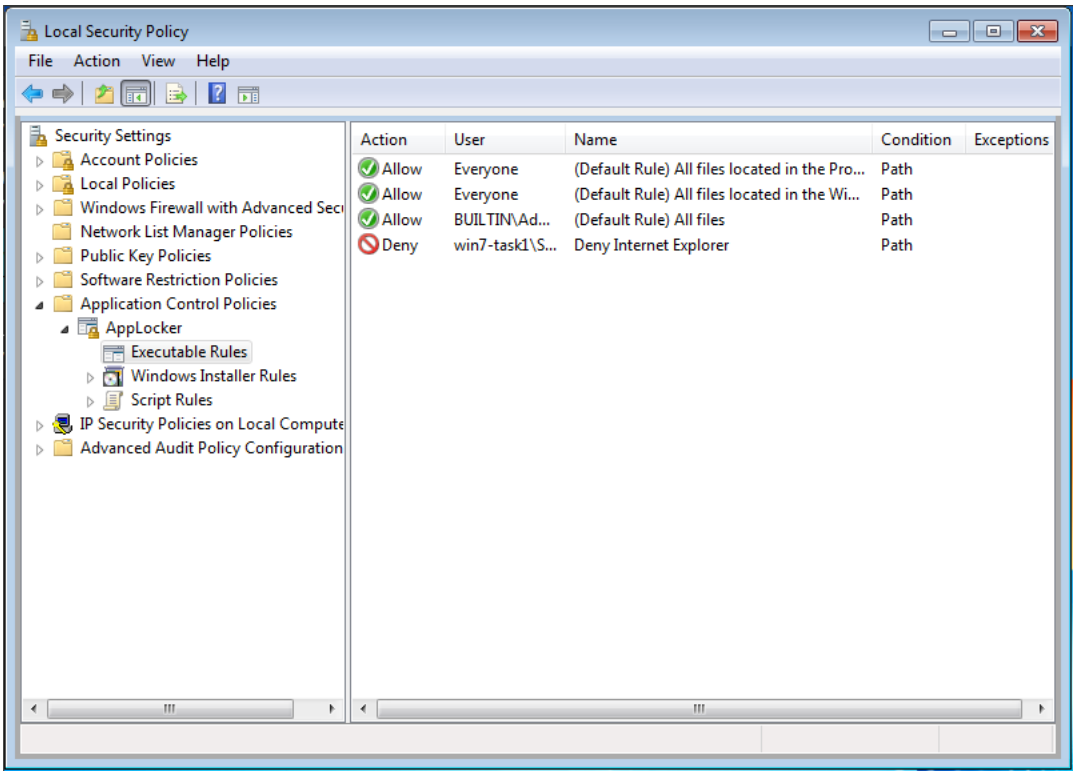
In conditions I selected **`Path`** to limit app by their file or folder path:



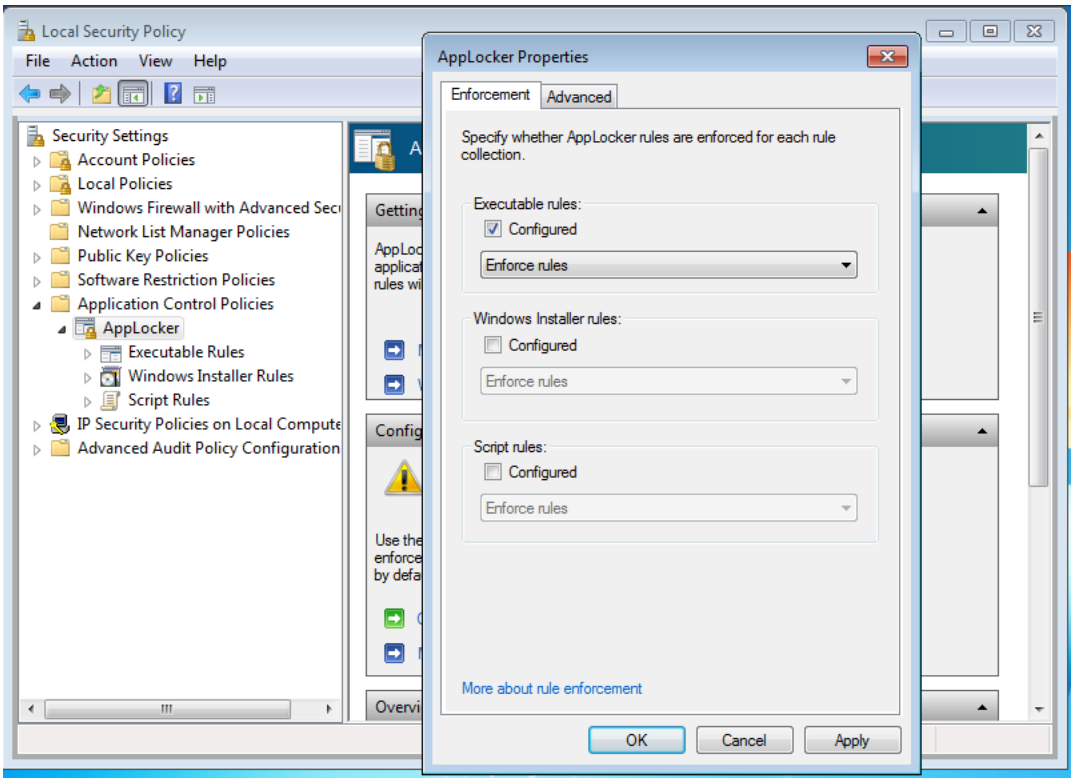
There I clicked **`Browse Folders`** and navigate to the **`Internet Explorer`** folder to block it:



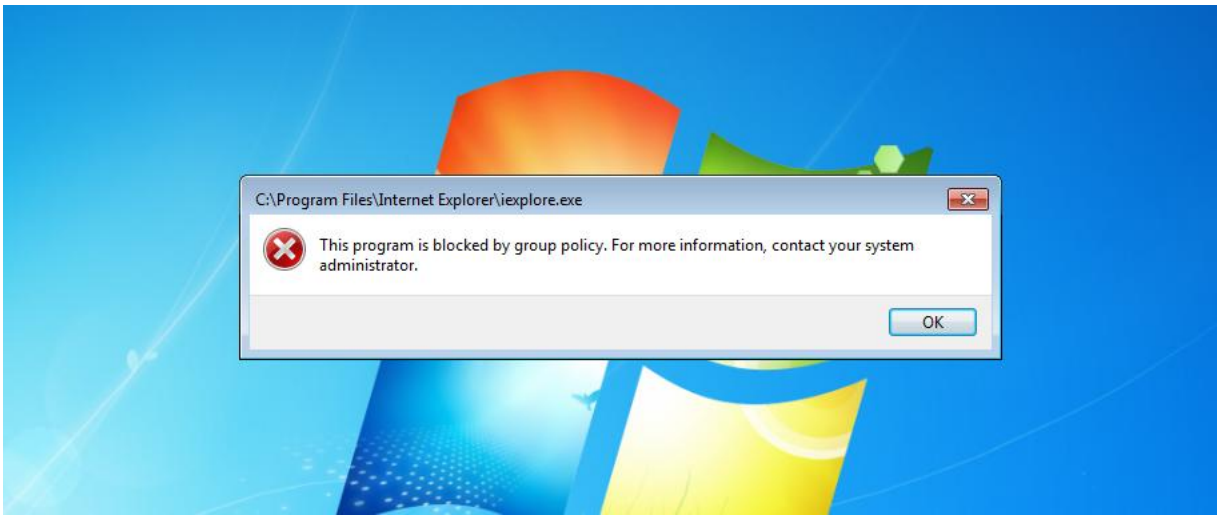
After finishing creation of rule the system asked me to add default rules to make it work properly. I added it.



Lastly, I right-clicked on **`AppLocker`**, selected **`Properties`** and checked that **`Executable rules`** are **`Configured`** and selected to **`Enforce rules`**:

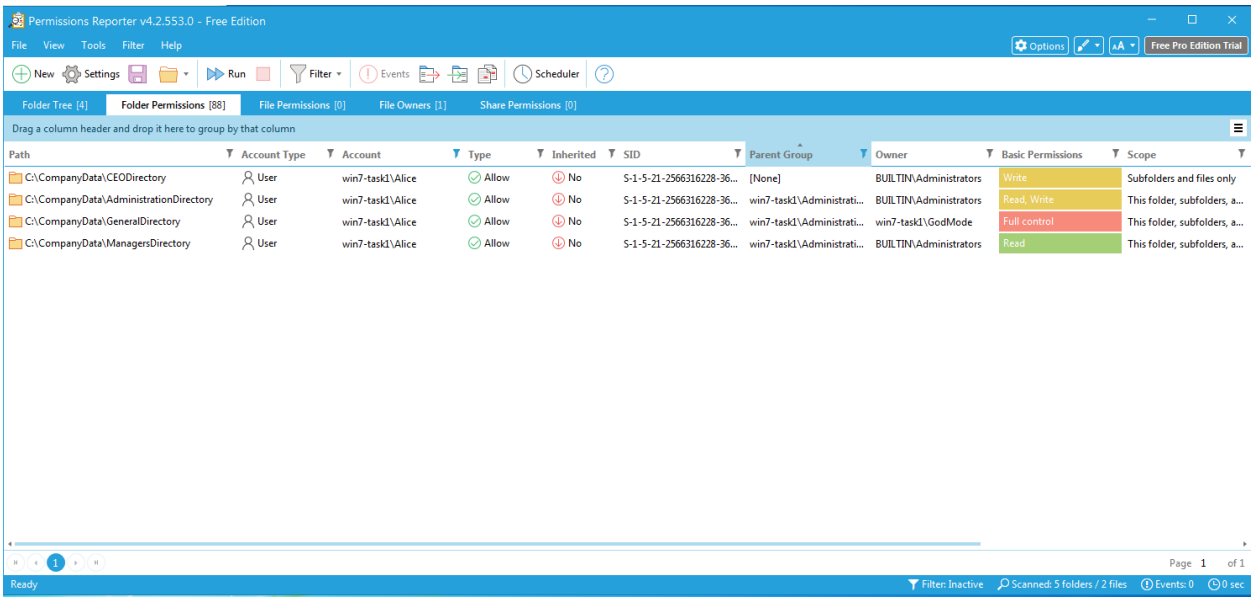


Then I logged in as **`Supreme`** and tried to access **`Internet Explorer`** app. It got me this error message:



9. **Utilize "NTFS Permissions Reporter" to check the permissions you have granted to one of the users**

I downloaded and installed **`NTFS Permissions Reporter`** from the official website. There I selected the folder of **`C:\CompanyData`** and scanned its folders. Then I navigated to **`Folder permissions`** tab and filtered by **`Alice`** account. It can be shown what permissions **`Alice`** has:



Linux

1. **Set access permissions for users according to scenario**
    a. **Creating User Groups**

```
sudo -i
groupadd <group name>
groupadd systems_administrator
groupadd ceo
groupadd administration
groupadd managers
groupadd unknown
```

`sudo -i` switched user to superuser mode (and there is no need to type sudo before each administrative command). `groupadd <group name>` adds group.

```
godmode@task1-simonas:~$ sudo -i
[sudo] password for godmode:
root@task1-simonas:~# groupadd systems_administrator
root@task1-simonas:~# groupadd ceo
root@task1-simonas:~# groupadd administration
root@task1-simonas:~# groupadd managers
root@task1-simonas:~# groupadd unknown
root@task1-simonas:~#
```

```
useradd -g ceo boss
useradd -g administration alice
useradd -g administration gabi
useradd -g managers anthony
useradd -g managers elisa
useradd -g managers jolie
useradd -g managers tom
useradd -g unknown supreme
```

`useradd -g <group user to be added> <user name>` adds a new user and assigns it to provided group.

```
root@task1-simonas:~# useradd -g ceo boss
root@task1-simonas:~# useradd -g administration alice
root@task1-simonas:~# useradd -g administration gabi
root@task1-simonas:~# useradd -g managers anthony
root@task1-simonas:~# useradd -g managers elisa
root@task1-simonas:~# useradd -g managers jolie
root@task1-simonas:~# useradd -g managers tom
root@task1-simonas:~# useradd -g unknown supreme
```

```
usermod -aG systems_administrator godmode
```

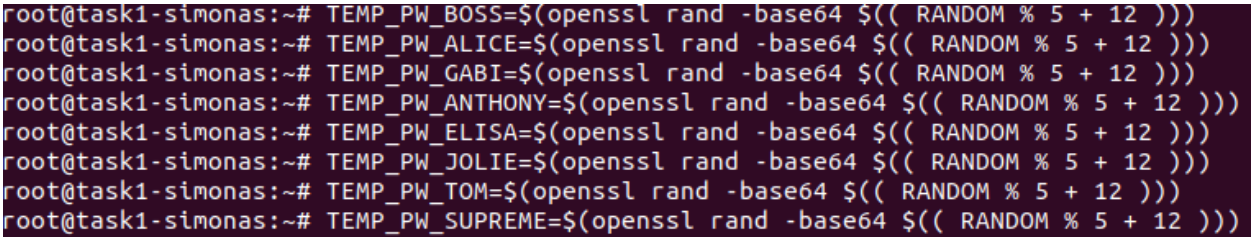
`usermod` command modify user's settings. `-a` option means append, to add user to a new group without removing it from groups it is a part of, without `-a` option usermod command would overwrite, `-G` specifies the group user to be added, then it is provided group name and user name.

```
root@task1-simonas:~# usermod -aG systems_administrator godmode
```

```
TEMP_PW_BOSS=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
TEMP_PW_ALICE=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
```

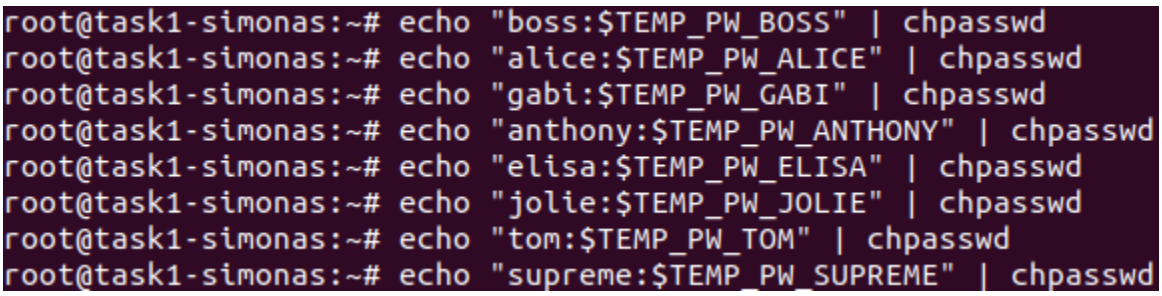
```
TEMP_PW_GABI=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
TEMP_PW_ANTHONY=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
TEMP_PW_ELISA=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
TEMP_PW_JOLIE=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
TEMP_PW_TOM=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
TEMP_PW_SUPREME=$(openssl rand -base64 $(( RANDOM % 5 + 12 )))
```

I created temporary passwords and assigned it into variables **TEMP_PW_<user>**. **${…}** this syntax runs a command and captures its output. **openssl rand** generates random data, **-base64** encodes the output in base64 format making it alphanumerc and suitable for passwords. **RANDOM** is a variable that generates a random integer, **%5** gives a remainder between 0 and 4, **+12** adjusts this range so the length is between 12 and 16.




```
echo "boss:$TEMP_PW_BOSS" | chpasswd
echo "alice:$TEMP_PW_ALICE" | chpasswd
echo "gabi:$TEMP_PW_GABI" | chpasswd
echo "anthony:$TEMP_PW_ANTHONY" | chpasswd
echo "elisa:$TEMP_PW_ELISA" | chpasswd
echo "jolie:$TEMP_PW_JOLIE" | chpasswd
echo "tom:$TEMP_PW_TOM" | chpasswd
echo "supreme:$TEMP_PW_SUPREME" | chpasswd
```

echo <user>:<temp password> outputs the string. The format is **username:password**, which is what **chpasswd** expects. Pipe operator (|) takes the output from the echo command and passes it as input to **chpasswd**. **chpasswd** reads input in the **username:password** format and updates the password for the specified user.




```
chage -d 0 boss
chage -d 0 alice
chage -d 0 gabi
chage -d 0 anthony
chage -d 0 elisa
chage -d 0 jolie
chage -d 0 tom
chage -d 0 supreme
```

chage is the command for managing password expiration and aging settings . **-d** option sets the date, when used with **0** it shows that the password is outdated and user must change it at the next login, then the user name is specified.

```
root@task1-simonas:~# chage -d 0 boss
root@task1-simonas:~# chage -d 0 alice
root@task1-simonas:~# chage -d 0 gabi
root@task1-simonas:~# chage -d 0 anthony
root@task1-simonas:~# chage -d 0 elisa
root@task1-simonas:~# chage -d 0 jolie
root@task1-simonas:~# chage -d 0 tom
root@task1-simonas:~# chage -d 0 supreme
```

```
mkdir -p /home/godmode/secure
```

`mkdir` created directory, `-p` creates folders for parents if they don't exist.

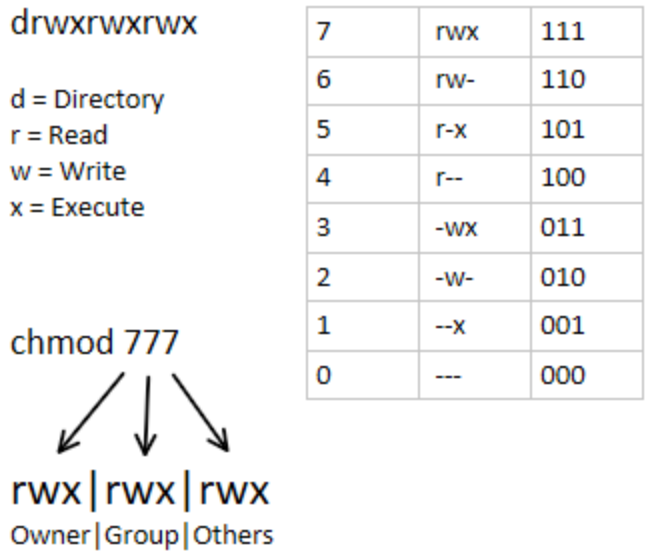
```
root@task1-simonas:~# mkdir -p /home/godmode/secure
```

```
chown godmode /home/godmode/secure
```

`chown` changes the ownership of a file or directory. Syntax for chown command is `chown <user to be the owner> <file or directory>`

```
root@task1-simonas:~# chown godmode /home/godmode/secure
```

```
chmod 700 /home/godmode/secure
```



7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000

`chmod` changes permissions of file or directory. In this case owner is given full permissions, no permissions to group or others, to make folder accessible only to owner (`godmode` as specified previously)

```
root@task1-simonas:~# chmod 700 /home/godmode/secure
```

```
SECURE_FILE="/home/godmode/secure/temp_passwords.txt"
```

Then I make variable of secure file the temporary passwords would be stored (in the folder where only `godmode` can access)

```
root@task1-simonas:~# SECURE_FILE="/home/godmode/secure/temp_passwords.txt"
```

```
touch $SECURE_FILE
```

With `touch` command I create a file in location specified in `SECURE_FILE` variable.

```
root@task1-simonas:~# touch $SECURE_FILE
```

```
chown godmode $SECURE_FILE
```

chown command sets owner to a file or directory. In this case I set **godmode** as owner for **SECURE_FILE**.

```
root@task1-simonas:~# chown godmode $SECURE_FILE
```

```
chmod 700 $SECURE_FILE
```

I also set **chmod** of **SECURE_FILE** to be accessible only to owner (godmode).

```
root@task1-simonas:~# chmod 700 $SECURE_FILE
```

```
echo -e "CEO: $TEMP_PW_CEO\nAlice: $TEMP_PW_ALICE\nGabi: $TEMP_PW_GABI\nAnthony:
$TEMP_PW_ANTHONY\nElisa: $TEMP_PW_ELISA\nJolie: $TEMP_PW_JOLIE\nTom:
$TEMP_PW_TOM\nSupreme: $TEMP_PW_SUPREME" > $SECURE_FILE
```

echo prints text, with **>** it outputs to file, in this case to **$SECURE_FILE**. **-e** option enables interpretation of escape characters within the text (**\n** is new file). Users with their temporary passwords are written in **$SECURE_FILE.**

```
root@task1-simonas:~# echo -e "CEO: $TEMP_PW_CEO\nAlice: $TEMP_PW_ALICE\nGabi: $TEMP_PW_GAB
I\nAnthony: $TEMP_PW_ANTHONY\nElisa: $TEMP_PW_ELISA\nJolie: $TEMP_PW_JOLIE\nTom: $TEMP_PW_T
OM\nSupreme: $TEMP_PW_SUPREME" > $SECURE_FILE
```

```
exit
```

I exited login shell as root user.

```
root@task1-simonas:~# exit
logout
```

```
cat /home/godmode/secure/temp_passwords.txt
```

With **godmode** account, which is owner of directory and file, I used **cat** command which displays the content of file.

```
godmode@task1-simonas:~$ cat /home/godmode/secure/temp_passwords.txt
CEO: wH0s8dTR4Yp8JDm9sfweoA==
Alice: XWLR333KNafXOLzTftTlvA==
Gabi: NIRlmikDcPqUYCE8wUipXQ==
Anthony: agl0S354enlfzcLqVTQ=
Elisa: wbSj9EmbSk9OMBcenzA=
Jolie: SggP0sfJqk9AvuYlxaw=
Tom: lmcIG2NMNCNKdYFMg767
Supreme: dLyi2h9aEyvNpjuPUa1N
godmode@task1-simonas:~$
```

General directory

```
mkdir -p /home/general
```

```
root@task1-simonas:~# mkdir -p /home/general
```

```
chmod 777 /home/general
```

```
root@task1-simonas:~# chmod 777 /home/general
```

Special directory

```
mkdir -p /home/special
```

```
root@task1-simonas:~# mkdir -p /home/special
```

```
chown godmode:systems_administrator /home/special
```

I assigned **godmode** as folder user owner and **systems_administrator** group as folder group owner.

```
root@task1-simonas:~# chown godmode:systems_administrator /home/special
```

```
chmod 720 /home/special
```

Owner (**godmode**) has full permissions, **systems_administrator** group only write.

```
root@task1-simonas:~# chmod 720 /home/special
```

```
ls -ld /home/special
```

ls lists files and directories, **-l** displays detailed (long) information, including permissions, number of links, owner, group, size, modification date, and name, **-d** lists the directory itself rather than its contents. Without **-d**, **ls -l <directory>** would list all files within that directory.

```
root@task1-simonas:~# ls -ld /home/special
drwx-w---- 2 godmode systems_administrator 4096
```

getfacl /home/special

getfacl command retrieves and displays ACL information, showing both the basic permissions (like those previously from from ls -l) and any additional permissions that have been set with ACLs.

```
root@task1-simonas:~# getfacl /home/special
getfacl: Removing leading '/' from absolute path names
# file: home/special
# owner: godmode
# group: systems_administrator
user::rwx
group::-w-
other::---
```

```
setfacl -m u:boss:rwx /home/special
```

setfacl command sets or modifies ACLs, **-m** option stands for “modify”, it allows to add or modify ACL entries, **u** means that permissions will be granted to user, then user name, and permissions (**r** – read, **w** – write, **x** – execute), and path where those permissions should be.

```
root@task1-simonas:~# setfacl -m u:boss:rwx /home/special
```

```
root@task1-simonas:~# getfacl /home/special
getfacl: Removing leading '/' from absolute path names
# file: home/special
# owner: godmode
# group: systems_administrator
user::rwx
user:boss:rwx
group::-w-
mask::rwx
other::---
```

```
root@task1-simonas:~# ls -ld /home/special
drwxrwx---+ 2 godmode systems_administrator 4096
```

Managers directory

```
root@task1-simonas:~# mkdir -p /home/managers
```

If no owner is provided than the owner is root

```
root@task1-simonas:~# chown :managers /home/managers
```

chmod 070 provides that the only managers group can access this folder and have full control there.

```
root@task1-simonas:~# chmod 070 /home/managers
```

If **`setfacl u::`** user is not provided that means that it is for owner user, **`g::`** for owner group, **`o::`** for others. **`-d`** stands for default, that means that it works on files. In this case user will have full control over user created files, for others only read.

```
root@task1-simonas:~# setfacl -d -m u::rwx /home/managers
```

```
root@task1-simonas:~# setfacl -d -m g::r-- /home/managers
```

```
root@task1-simonas:~# setfacl -d -m o::--- /home/managers
```

`su` command switch users. It is good for testing in other users. From **`root`** there is no password for switching user.

```
root@task1-simonas:~# su anthony
$ touch /home/managers/anthony_file_1.txt
$ cat /home/managers/anthony_file_1.txt
$ exit
```

```
root@task1-simonas:~# su elisa
You are required to change your password immediately (administrator enforced)
Changing password for elisa.
Current password:
New password:
Retype new password:
$ cat /home/managers/anthony_file_1.txt
$ exit
```

```
root@task1-simonas:~# su elisa
$ echo "Test" >> /home/managers/anthony_file.txt
sh: 1: cannot create /home/managers/anthony_file.txt: Permission denied
$ exit
```

```
root@task1-simonas:~# getfacl /home/managers
getfacl: Removing leading '/' from absolute path names
# file: home/managers
# owner: root
# group: managers
user::---
group::rwx
other::---
default:user::rwx
default:group::r--
default:other::---
```

```
root@task1-simonas:~# getfacl /home/managers/anthony_file_1.txt
getfacl: Removing leading '/' from absolute path names
# file: home/managers/anthony_file_1.txt
# owner: anthony
# group: managers
user::rw-
group::r--
other::---
```

```
godmode@task1-simonas:~$ cat /home/managers/anthony_file.txt
cat: /home/managers/anthony_file.txt: Permission denied
godmode@task1-simonas:~$ ls /home/managers
ls: cannot open directory '/home/managers': Permission denied
```

```
root@task1-simonas:~# su elisa
$ ls /home/managers
anthony_file_1.txt  anthony_file.txt
$ getfacl /home/managers/anthony_file_1.txt
getfacl: Removing leading '/' from absolute path names
# file: home/managers/anthony_file_1.txt
# owner: anthony
# group: managers
user::rw-
group::r--
other::---

$ getfacl /home/managers
getfacl: Removing leading '/' from absolute path names
# file: home/managers
# owner: root
# group: managers
user::---
group::rwx
other::---
default:user::rwx
default:group::r--
default:other::---

$ rm /home/managers/anthony_file_1.txt
rm: remove write-protected regular empty file '/home/managers/anthony_file_1.txt'? y
$ ls /home/managers
anthony_file.txt
$
```

`+t` flag adds sticky bit. When the sticky bit is set on a directory, only the directory's owner, the file's owner, or the superuser (root) can delete or rename files within that directory, regardless of write permissions for other users.

```
root@task1-simonas:~# chmod o+t /home/managers
```

```
root@task1-simonas:~# ls -ld /home/managers
```

```
d---rwx--T+ 2 root managers 4096
```

```
root@task1-simonas:~# su anthony
$ touch /home/managers/anthony_file_new.txt
$ exit
```

```
$ rm /home/managers/anthony_file_new.txt
rm: remove write-protected regular empty file '/home/managers/anthony_file_new.txt'? y
rm: cannot remove '/home/managers/anthony_file_new.txt': Operation not permitted
$ exit
root@task1-simonas:~# su anthony
$ rm /home/managers/anthony_file_new.txt
$ exit
```

`-R` option stands for "recursive" and it applies the specified ACL settings not only to the directory itself but also to all files and subdirectories within it.

`-d` option means that all files and subdirectories created inside will inherit these permissions by default, but it won't change permissions on existing files or directories within it.

Combined sets that every item created within the folder and its subdirectories inherit specified permissions.

```
root@task1-simonas:~# setfacl -m g:ceo:rwx /home/managers
root@task1-simonas:~# setfacl -R -m g:ceo:rwx /home/managers
root@task1-simonas:~# setfacl -d -m g:ceo:rwx /home/managers
```

There were some issues when trying to create and access CEO created file, setting mask and to others read access seems to solve the issue.

```
root@task1-simonas:~# setfacl -d -m o::r-- /home/managers
```

```
root@task1-simonas:~# setfacl -d -m m::rwx /home/managers
```

```
root@task1-simonas:~# getfacl /home/managers
getfacl: Removing leading '/' from absolute path names
# file: home/managers
# owner: root
# group: managers
# flags: --t
user::---
group::rwx
group:ceo:rwx
mask::rwx
other::---
default:user::rwx
default:group::r--
default:group:ceo:rwx
default:mask::rwx
default:other::r--
```

```
root@task1-simonas:~# su boss
$ ls /home/managers
anthony_file.txt
$ touch /home/managers/ceo_file_for_managers.txt
$ echo "Test" >> /home/managers/ceo_file_for_managers.txt
$ cat /home/managers/ceo_file_for_managers.txt
Test
```

```
root@task1-simonas:~# su boss
$ touch /home/managers/ceo_file.txt
$ exit
root@task1-simonas:~# su anthony
$ cat /home/managers/ceo_file.txt
$ exit
root@task1-simonas:~# su anthony
$ echo "Testas" >> /home/managers/ceo_file.txt
sh: 1: cannot create /home/managers/ceo_file.txt: Permission denied
$ exit
```

```
root@task1-simonas:~# setfacl -m g:administration:rx /home/managers
root@task1-simonas:~# setfacl -R -m g:administration:rx /home/managers
root@task1-simonas:~# setfacl -d -m g:administration:rx /home/managers
root@task1-simonas:~# setfacl -m g:systems_administrator:rx /home/managers
root@task1-simonas:~# setfacl -R -m g:systems_administrator:rx /home/managers
root@task1-simonas:~# setfacl -d -m g:systems_administrator:rx /home/managers
root@task1-simonas:~#
```

Administration folder

```
root@task1-simonas:~# mkdir -p /home/administration
root@task1-simonas:~# chown :administration /home/administration
root@task1-simonas:~# chmod 070 /home/administration
root@task1-simonas:~# setfacl -d -m u::rwx /home/administration
root@task1-simonas:~# setfacl -d -m g::r-- /home/administration
root@task1-simonas:~# setfacl -d -m o::r-- /home/administration
root@task1-simonas:~# setfacl -d -m m::rwx /home/administration
root@task1-simonas:~# chmod o+t /home/administration
root@task1-simonas:~# setfacl -m g:ceo:rwx /home/administration
root@task1-simonas:~# setfacl -R -m g:ceo:rwx /home/administration
root@task1-simonas:~# setfacl -d -m g:ceo:rwx /home/administration
root@task1-simonas:~# setfacl -m g:systems_administrator:rx /home/administration
root@task1-simonas:~# setfacl -R -m g:systems_administrator:rx /home/administration
root@task1-simonas:~# setfacl -d -m g:systems_administrator:rx /home/administration
```

```
root@task1-simonas:~# getfacl /home/administration
getfacl: Removing leading '/' from absolute path names
# file: home/administration
# owner: root
# group: administration
# flags: --t
user::---
group::rwx
group:systems_administrator:r-x
group:ceo:rwx
mask::rwx
other::---
default:user::rwx
default:group::r--
default:group:systems_administrator:r-x
default:group:ceo:rwx
default:mask::rwx
default:other::r--
```

CEO folder

```
root@task1-simonas:~# mkdir -p /home/ceo
root@task1-simonas:~# chown boss:ceo /home/ceo
root@task1-simonas:~# chmod 770 /home/ceo
```

```
root@task1-simonas:~# setfacl -R -m g:systems_administrator:rx /home/ceo
root@task1-simonas:~# setfacl -d -m g:systems_administrator:rx /home/ceo
```

```
root@task1-simonas:~# setfacl -d -m u:alice:wx /home/ceo
root@task1-simonas:~# setfacl -R -m u:alice:wx /home/ceo
```

```
root@task1-simonas:~# setfacl -R -m g:ceo:rwx /home/ceo
root@task1-simonas:~# setfacl -d -m g:ceo:rwx /home/ceo
```

```
root@task1-simonas:~# su boss
$ touch /home/ceo/very_sensitive_data.txt
$ echo "Sensitive data by CEO" > /home/ceo/very_sensitive_data.txt
$ cat /home/ceo/very_sensitive_data.txt
Sensitive data by CEO
$ exit
root@task1-simonas:~# su alice
$ ls /home/ceo
ls: cannot open directory '/home/ceo': Permission denied
$ cat /home/ceo/very_sensitive_data.txt
cat: /home/ceo/very_sensitive_data.txt: Permission denied
$ touch /home/ceo/report_for_ceo.txt
$ echo "This is a report for CEO by Alice" > /home/ceo/report_for_ceo.txt
$ cat /home/ceo/report_for_ceo.txt
This is a report for CEO by Alice
$
```

```
root@task1-simonas:~# su boss
$ cat /home/ceo/report_for_ceo.txt
This is a report for CEO by Alice
$
```

```
root@task1-simonas:~# chmod +t /home/ceo
```

```
root@task1-simonas:~# su alice
$ rm /home/ceo/very_sensitive_data.txt
rm: cannot remove '/home/ceo/very_sensitive_data.txt': Operation not permitted
```

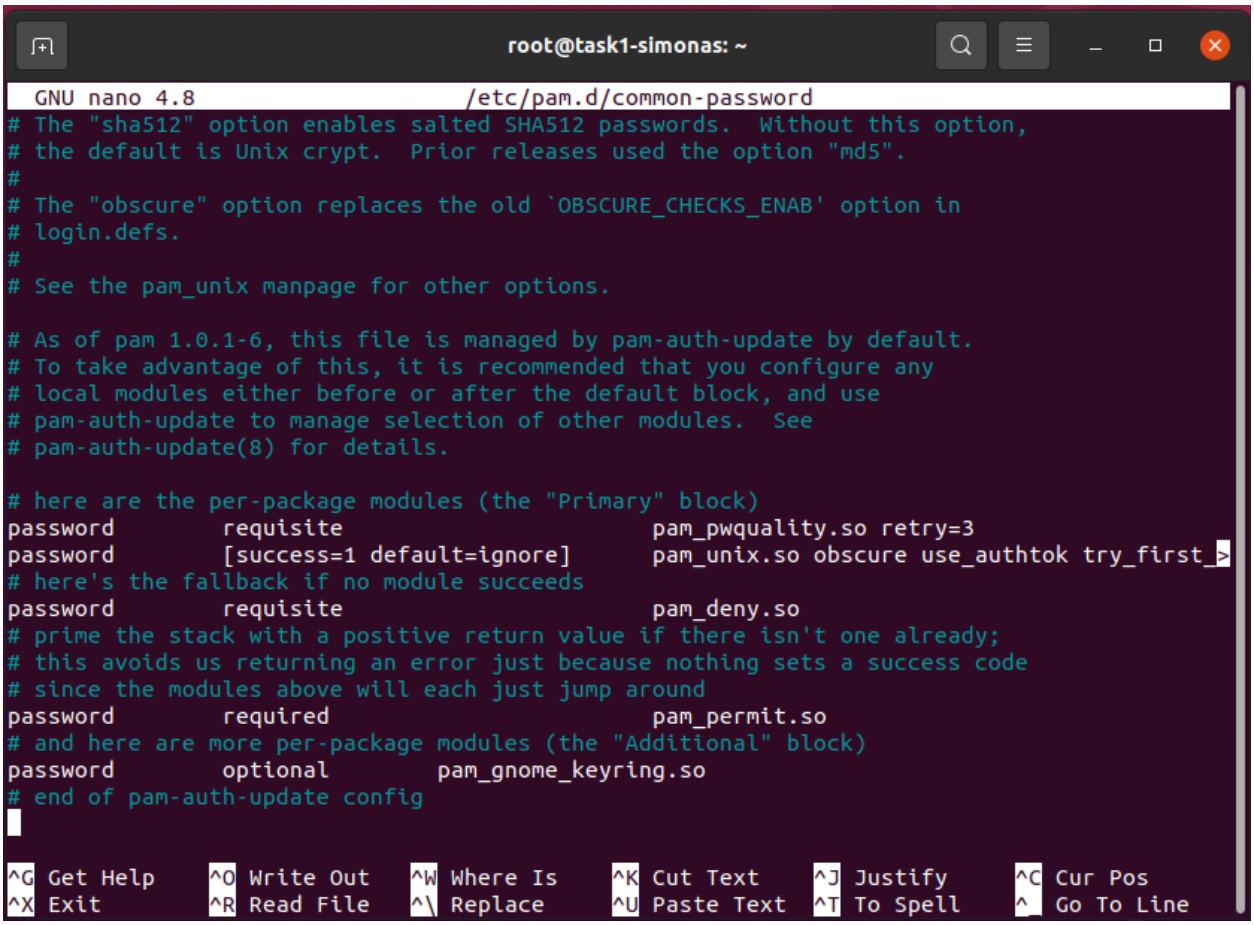
```
root@task1-simonas:~# su boss
$ ls /home/ceo
report_for_ceo.txt  very_sensitive_data.txt
$ rm /home/ceo/report_for_ceo.txt
```

```
root@task1-simonas:~# su boss
$ ls /home/ceo
report_for_ceo.txt  very_sensitive_data.txt
$ rm /home/ceo/report_for_ceo.txt
$ ls /home/ceo
very_sensitive_data.txt
```

Passwords

First, I `apt-get install libpam-pwquality` and edited the `/etc/pam.d/common-password` file to set password strength requirements:

```
nano /etc/pam.d/common-password
```



In this file, I located the line containing `pam_pwquality.so` and modified it to include the desired password complexity options so command looks like this:

```
password requisite pam_pwquality.so retry=3 minlen=12 dcredit=-1 ucredit=-1
ocredit=-1 lcredit=-1
```

`retry=3` allows the user three attempts to enter a valid password before failing.

`minlen=12` sets the minimum password length to 12 characters.

`dcredit=-1` requires at least one digit.

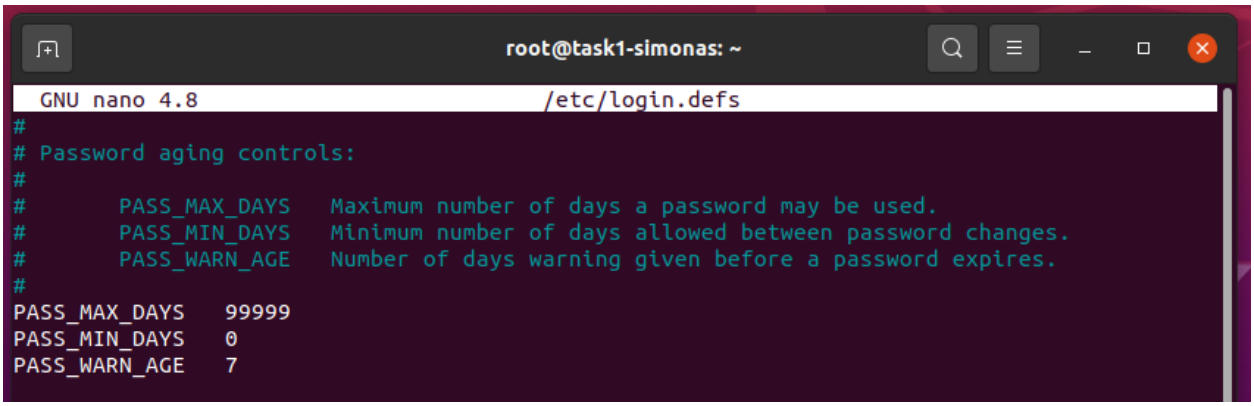
`ucredit=-1` requires at least one uppercase letter.

`lcredit=-1` requires at least one lowercase letter.

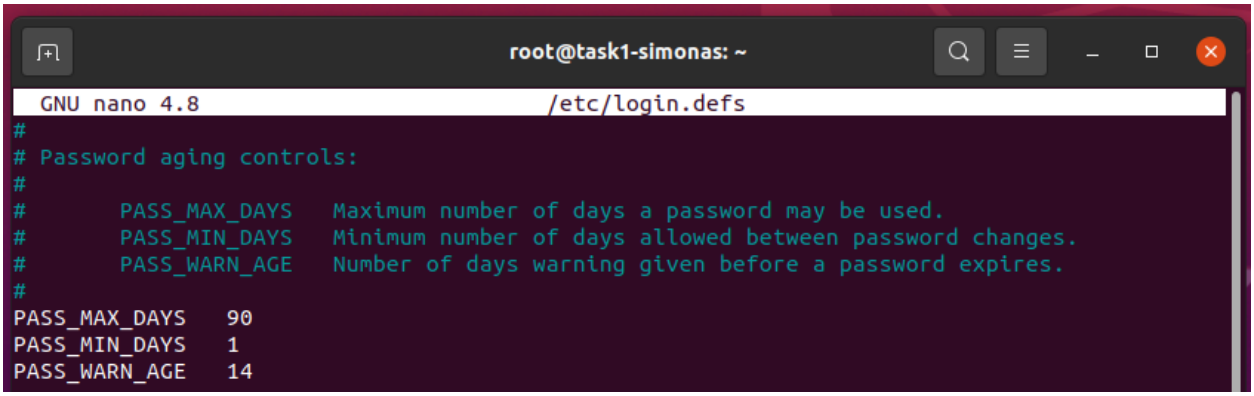
`ocredit=-1` requires at least one special character.

After making these changes, I saved the file by pressing `Ctrl + S`, then exited with `Ctrl + X`.

Next, I edited the `/etc/login.defs` file to set the password aging policies:



I adjusted the following parameters:



`PASS_MAX_DAYS 90` sets the maximum number of days a password remains valid (90 days).

`PASS_MIN_DAYS 1` requires at least one day before a user can change their password again.

`PASS_WARN_AGE 14` sends a warning to the user 14 days before the password expires.

To apply these aging policies to existing user accounts (because **`/etc/login.defs`** works on newly created users), I used the chage command for all the users:

```
chage -M 90 -m 1 -W 14 godmode
chage -M 90 -m 1 -W 14 boss
chage -M 90 -m 1 -W 14 alice
chage -M 90 -m 1 -W 14 gabi
chage -M 90 -m 1 -W 14 anthony
chage -M 90 -m 1 -W 14 elisa
chage -M 90 -m 1 -W 14 jolie
chage -M 90 -m 1 -W 14 tom
chage -M 90 -m 1 -W 14 supreme
```

`-M 90` sets the maximum password age to 90 days.
`-m 1` sets the minimum password age to 1 day.
`-W 14` sets the warning period to 14 days before expiration.

```
root@task1-simonas:~# chage -M 90 -m 1 -W 14 godmode
root@task1-simonas:~# chage -M 90 -m 1 -W 14 boss
root@task1-simonas:~# chage -M 90 -m 1 -W 14 alice
root@task1-simonas:~# chage -M 90 -m 1 -W 14 gabi
root@task1-simonas:~# chage -M 90 -m 1 -W 14 anthony
root@task1-simonas:~# chage -M 90 -m 1 -W 14 elisa
root@task1-simonas:~# chage -M 90 -m 1 -W 14 jolie
root@task1-simonas:~# chage -M 90 -m 1 -W 14 tom
root@task1-simonas:~# chage -M 90 -m 1 -W 14 supreme
```

To check if everything changed we can use **`chage -l <username>.`**

To test the new password policies, I attempted to change a user's password to those that doesn't meet the new complexity requirements:

`passwd`

```
godmode@task1-simonas:~$ su tom
Password:
$ passwd
Changing password for tom.
Current password:
New password:
BAD PASSWORD: The password contains less than 1 digits
New password:
BAD PASSWORD: The password contains less than 1 uppercase letters
New password:
BAD PASSWORD: The password contains less than 1 non-alphanumeric characters
passwd: Have exhausted maximum number of retries for service
passwd: password unchanged
$ passwd
Changing password for tom.
Current password:
New password:
BAD PASSWORD: The password is shorter than 12 characters
New password:
Retype new password:
passwd: password updated successfully
```

I also checked the password aging policy:

```
godmode@task1-simonas:~$ su tom
Password:
$ passwd
Changing password for tom.
Current password:
You must wait longer to change your password
passwd: Authentication token manipulation error
passwd: password unchanged
$
```

I used the chage command to simulate the password nearing its expiration:

```
root@task1-simonas:~# chage -d $(date -d "-85 days" +"%Y-%m-%d") tom
root@task1-simonas:~# exit
logout
godmode@task1-simonas:~$ su tom
Password:
Warning: your password will expire in 5 days
$
```

`$(date -d "-85 days" +"%Y-%m-%d")` calculates the date 85 days ago in the format `YYYY-MM-DD`. `date -d "-85 days"` command finds the date 85 days before today. `+ "%Y-%m-%d"` formats the output as `YYYY-MM-DD`, which `chage` requires.

Prohibit employees from accessing terminal (CLI)

First, I searched what is location of `gnome-terminal` with the following command:

```
which gnome-terminal
```

```
root@task1-simonas:~# which gnome-terminal
/usr/bin/gnome-terminal
```

I checked permissions for `/usr/bin/gnome-terminal`:

```
getfacl /usr/bin/gnome-terminal
```

```
root@task1-simonas:~# getfacl /usr/bin/gnome-terminal
getfacl: Removing leading '/' from absolute path names
# file: usr/bin/gnome-terminal
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
```

I changed the group ownership of the GNOME Terminal application to `systems_administrator`:

```
chgrp systems_administrator /usr/bin/gnome-terminal
```

`chgrp` changes owner group to specified path.

```
root@task1-simonas:~# chgrp systems_administrator /usr/bin/gnome-terminal
root@task1-simonas:~# getfacl /usr/bin/gnome-terminal
getfacl: Removing leading '/' from absolute path names
# file: usr/bin/gnome-terminal
# owner: root
# group: systems_administrator
user::rwx
group::r-x
other::r-x
```

I modified the permissions of the GNOME Terminal so that only the owner and group can execute it:

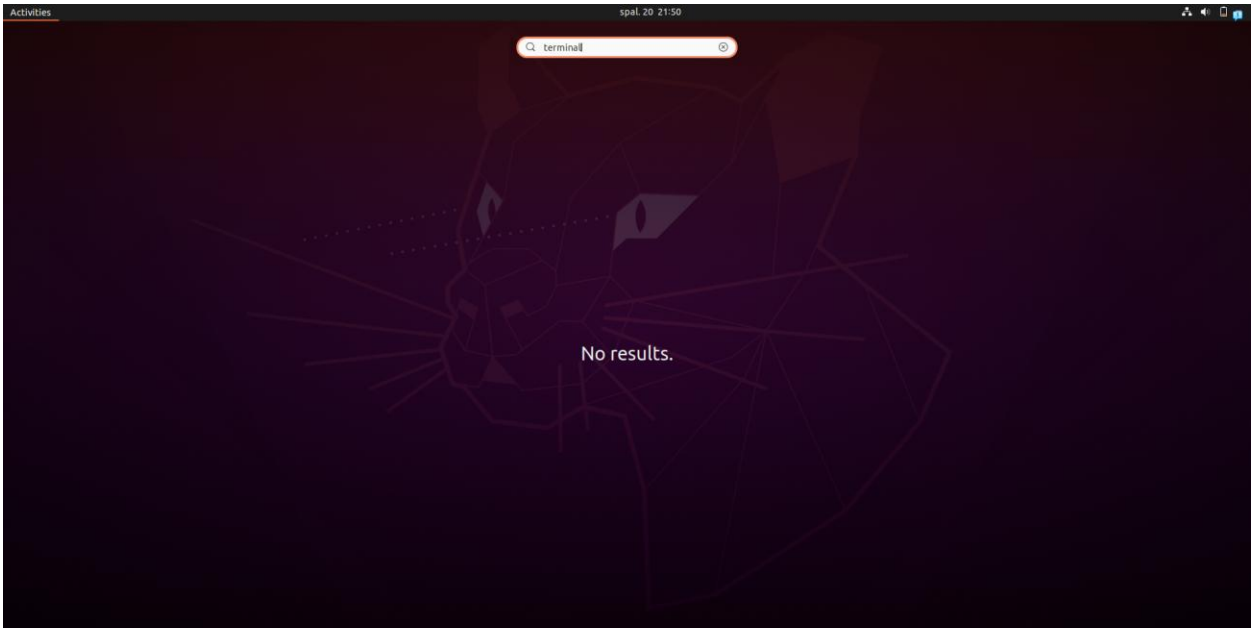
```
chmod 750 /usr/bin/gnome-terminal
```

```
root@task1-simonas:~# chmod 750 /usr/bin/gnome-terminal
root@task1-simonas:~# getfacl /usr/bin/gnome-terminal
getfacl: Removing leading '/' from absolute path names
# file: usr/bin/gnome-terminal
# owner: root
# group: systems_administrator
user::rwx
group::r-x
other::---
```

I verified that the restricted users could no longer open terminal applications, while allowed users retained access. Since I was learning at the beginning of this task, I didn't make the flag `-m` while creating new users with **`useradd`** and it didn't create any users' home locations. I created it myself now for user Tom (to test and switch user via GUI):

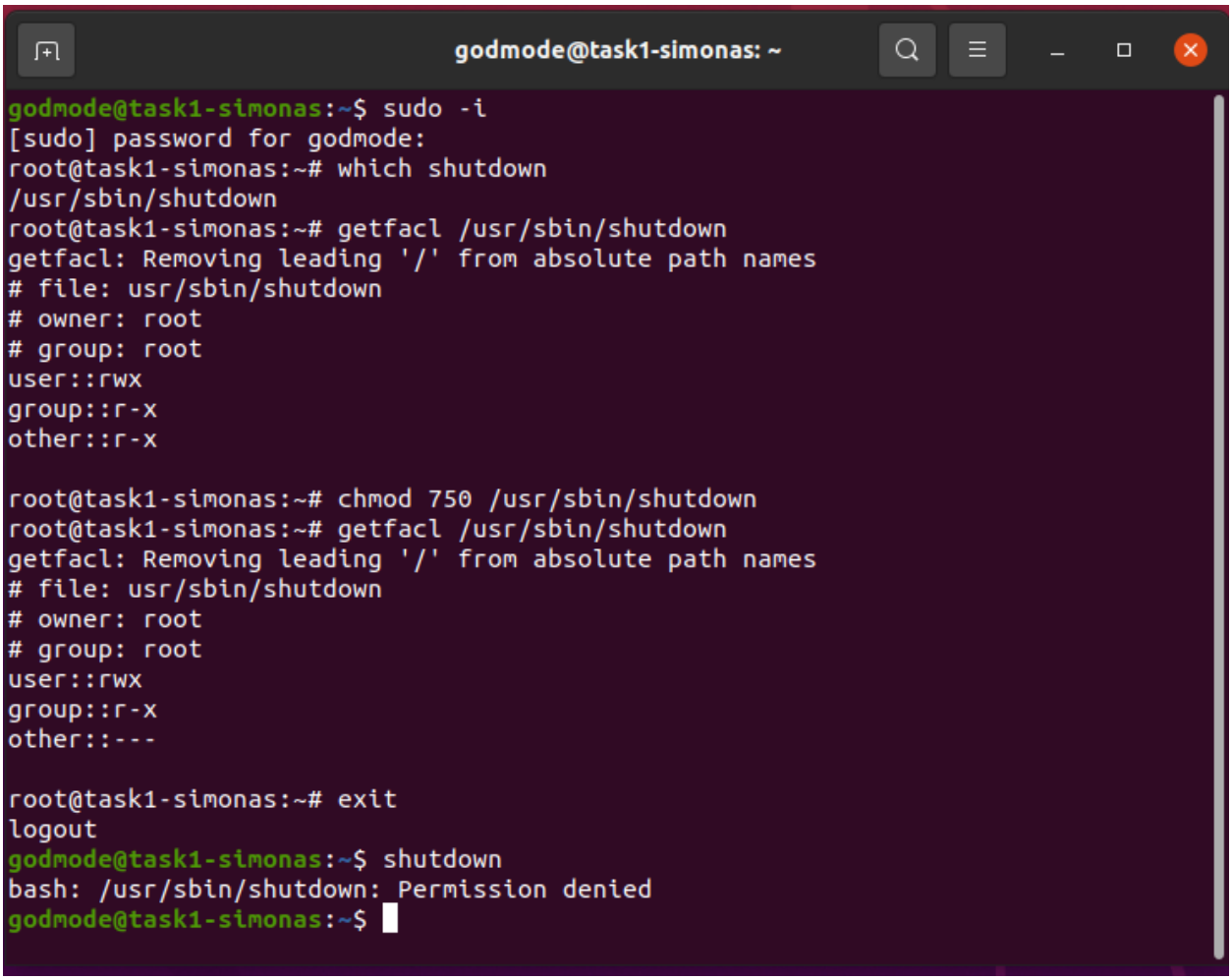
```
root@task1-simonas:~# mkdir /home/tom
root@task1-simonas:~# chown tom /home/tom
root@task1-simonas:~# chmod 755 /home/tom
root@task1-simonas:~# getfacl /home/tom
getfacl: Removing leading '/' from absolute path names
# file: home/tom
# owner: tom
# group: root
user::rwx
group::r-x
other::r-x
```

I logged in as a Tom and there was no terminal for Tom:



To revert **`chmod 755 /usr/bin/gnome-terminal command`** could be used.

Prohibit using shutdown command



Prohibit employees from configuring network adapter:

Files in /etc/netplan contains YAML files that define network settings, those files contais configuration details for network interfaces.

```
root@task1-simonas:~# ls /etc/netplan
01-network-manager-all.yaml
root@task1-simonas:~# getfacl /etc/netplan
getfacl: Removing leading '/' from absolute path names
# file: etc/netplan
# owner: root
# group: root
user::rwx
group::r-x
other::r-x

root@task1-simonas:~# chmod 750 /etc/netplan
root@task1-simonas:~# getfacl /etc/netplan
getfacl: Removing leading '/' from absolute path names
# file: etc/netplan
# owner: root
# group: root
user::rwx
group::r-x
other::---
```

Activate event logging with rsyslog

I made sure that the **rsyslog** service is enabled to start at boot and then started it:

```
systemctl enable rsyslog
systemctl start rsyslog
```

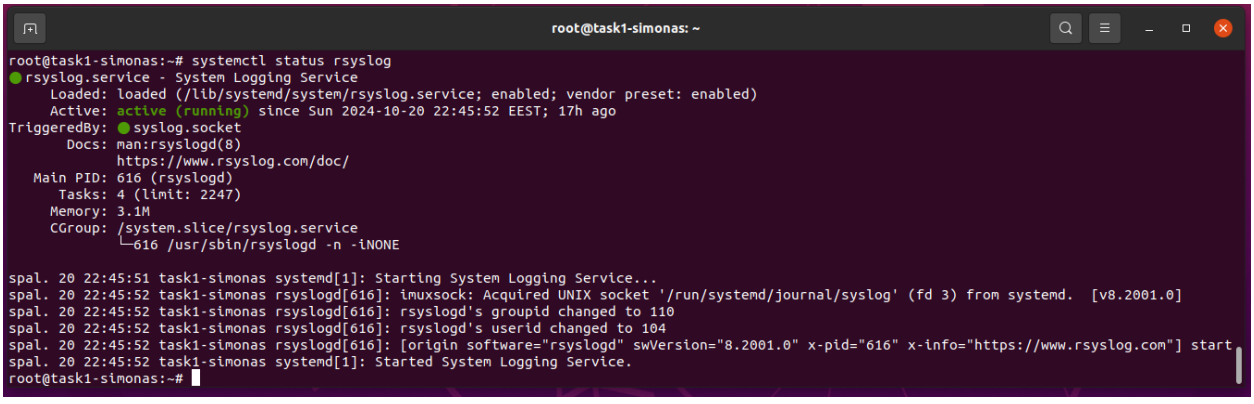
systemctl enable ensures that the service starts automatically after reboot. **systemctl start** initiates the service right away.

```
root@task1-simonas:~# systemctl enable rsyslog
Synchronizing state of rsyslog.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable rsyslog
root@task1-simonas:~# systemctl start rsyslog
```

Then, I checked whether the service is running correctly by using:

```
systemctl status rsyslog
```

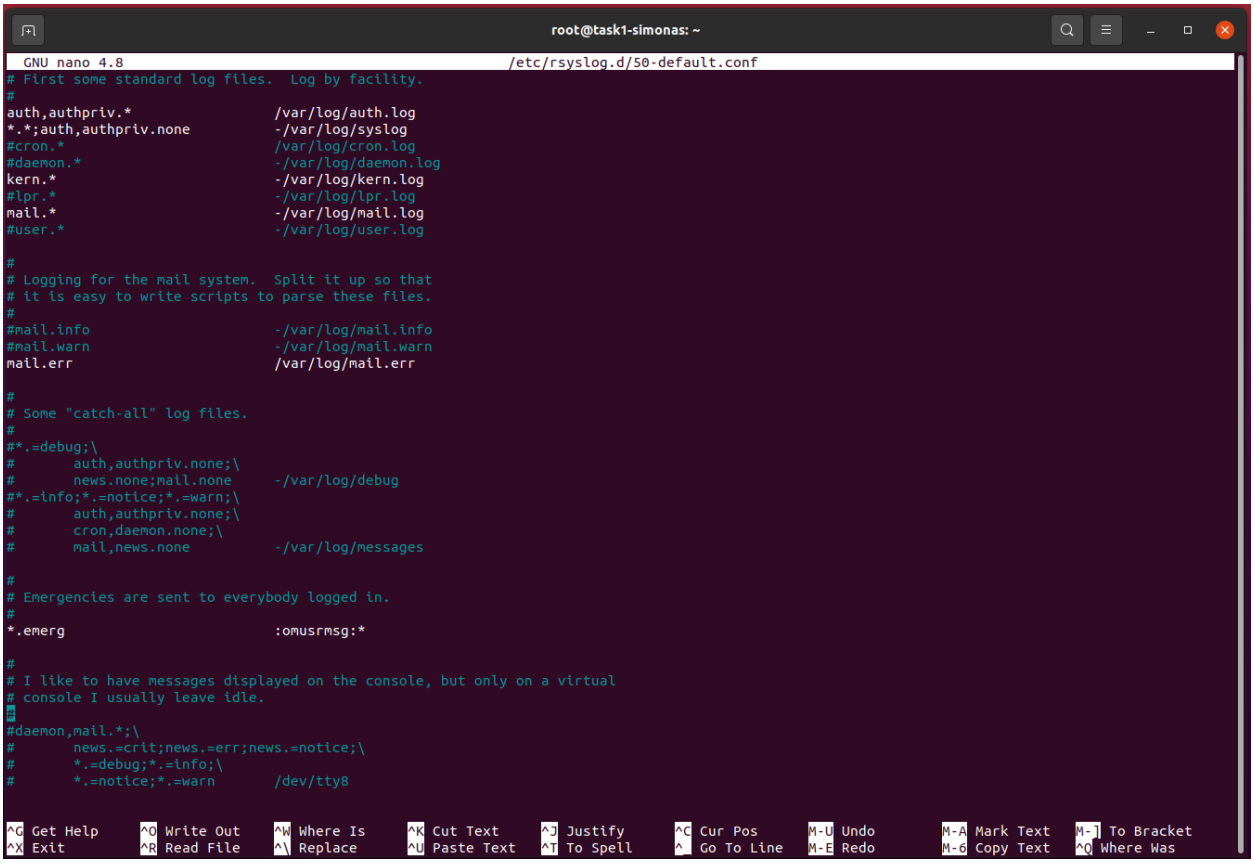
This command shows the current state of **rsyslog**, confirming it is active and running.



Then, for configuring rsyslog I opened the rsyslog configuration file for editing:

```
nano /etc/rsyslog.d/50-default.conf
```

```
root@task1-simonas:~# nano /etc/rsyslog.d/50-default.conf
```



Inside this file, I left authentication logs (all authentication events (success and failure), for security monitoring), system-wide logs (all system events except authentication), kernel logs (kernel events, for tracking hardware and system issues).

Then, I restarted **rsyslog** service to apply changes:

```
systemctl restart rsyslog
```

```
root@task1-simonas:~# systemctl restart rsyslog
```

Then, I checked the logs to verify if rsyslog is properly logging events by viewing the syslog:

```
tail -f /var/log/syslog
```

```
root@task1-simonas:~# tail -f /var/log/syslog
Oct 21 16:25:16 task1-simonas systemd[1]: Stopping System Logging Service...
Oct 21 16:25:16 task1-simonas systemd[1]: rsyslog.service: Succeeded.
Oct 21 16:25:16 task1-simonas rsyslogd: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="616" x-info="https://www.rsyslog.com"] exiting on signal 15.
Oct 21 16:25:16 task1-simonas systemd[1]: Stopped System Logging Service.
Oct 21 16:25:16 task1-simonas systemd[1]: Starting System Logging Service...
Oct 21 16:25:16 task1-simonas rsyslogd: imuxsock: Acquired UNIX socket '/run/systemd/journal/syslog' (fd 3) from systemd. [v8.2001.0]
Oct 21 16:25:16 task1-simonas rsyslogd: rsyslogd's groupid changed to 110
Oct 21 16:25:16 task1-simonas systemd[1]: Started System Logging Service.
Oct 21 16:25:16 task1-simonas rsyslogd: rsyslogd's userid changed to 104
Oct 21 16:25:16 task1-simonas rsyslogd: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="3871" x-info="https://www.rsyslog.com"] start
```

I also tested logging by generating a test log entry using the logger command:

```
logger "Test log entry"
```

And checked **/var/log/syslog** to see if the entry was recorded with **tail -f /var/log/syslog**

```
root@task1-simonas:~# logger "Test log entry"
root@task1-simonas:~# tail -f /var/log/syslog
Oct 21 16:25:16 task1-simonas systemd[1]: rsyslog.service: Succeeded.
Oct 21 16:25:16 task1-simonas rsyslogd: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="616" x-info="https://www.rsyslog.com"] exiting on signal 15.
Oct 21 16:25:16 task1-simonas systemd[1]: Stopped System Logging Service.
Oct 21 16:25:16 task1-simonas systemd[1]: Starting System Logging Service...
Oct 21 16:25:16 task1-simonas rsyslogd: imuxsock: Acquired UNIX socket '/run/systemd/journal/syslog' (fd 3) from systemd.  [v8.2001.0]
Oct 21 16:25:16 task1-simonas rsyslogd: rsyslogd's groupid changed to 110
Oct 21 16:25:16 task1-simonas systemd[1]: Started System Logging Service.
Oct 21 16:25:16 task1-simonas rsyslogd: rsyslogd's userid changed to 104
Oct 21 16:25:16 task1-simonas rsyslogd: [origin software="rsyslogd" swVersion="8.2001.0" x-pid="3871" x-info="https://www.rsyslog.com"] start
Oct 21 16:26:58 task1-simonas godmode: Test log entry
```

Then I checked authentication log with command:

`tail -f /var/log/auth.log`

```
root@task1-simonas:~# tail -f /var/log/auth.log
Oct 21 16:31:14 task1-simonas systemd-logind[625]: New session c3 of user gdm.
Oct 21 16:31:14 task1-simonas systemd: pam_unix(systemd-user:session): session opened for user gdm by (uid=0)
Oct 21 16:31:16 task1-simonas polkitd(authority=local): Registered Authentication Agent for unix-session:c3 (system bus name :1.206 [/usr/bin/gnome-shell], object
path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Oct 21 16:31:20 task1-simonas systemd-logind[625]: Removed session 9.
Oct 21 16:31:24 task1-simonas gdm-password]: gkr-pam: unlocked login keyring
Oct 21 16:31:34 task1-simonas gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session closed for user gdm
Oct 21 16:31:34 task1-simonas systemd-logind[625]: Session c3 logged out. Waiting for processes to exit.
Oct 21 16:31:34 task1-simonas systemd-logind[625]: Removed session c3.
Oct 21 16:31:34 task1-simonas polkitd(authority=local): Unregistered Authentication Agent for unix-session:c3 (system bus name :1.206, object path /org/freedesktop
/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8) (disconnected from bus)
Oct 21 16:31:39 task1-simonas dbus-daemon[592]: [system] Failed to activate service 'org.bluez': timed out (service_start_timeout=25000ms)
```

Then, I decided to login as “tom” at it got reflected:

```
root@task1-simonas: ~
Oct 21 16:31:14 task1-simonas systemd: pam_unix(systemd-user:session): session opened for user gdm by (uid=0)
Oct 21 16:31:16 task1-simonas polkitd(authority=local): Registered Authentication Agent for unix-session:c3 (system bus name :1.206 [/usr/bin/gnome-shell], object
path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Oct 21 16:31:20 task1-simonas systemd-logind[625]: Removed session 9.
Oct 21 16:31:24 task1-simonas gdm-password]: gkr-pam: unlocked login keyring
Oct 21 16:31:34 task1-simonas gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session closed for user gdm
Oct 21 16:31:34 task1-simonas systemd-logind[625]: Session c3 logged out. Waiting for processes to exit.
Oct 21 16:31:34 task1-simonas systemd-logind[625]: Removed session c3.
Oct 21 16:31:34 task1-simonas polkitd(authority=local): Unregistered Authentication Agent for unix-session:c3 (system bus name :1.206, object path /org/freedesktop
/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8) (disconnected from bus)
Oct 21 16:31:39 task1-simonas dbus-daemon[592]: [system] Failed to activate service 'org.bluez': timed out (service_start_timeout=25000ms)
Oct 21 16:36:32 task1-simonas gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session opened for user gdm by (uid=0)
Oct 21 16:36:32 task1-simonas systemd-logind[625]: New session c4 of user gdm.
Oct 21 16:36:32 task1-simonas systemd: pam_unix(systemd-user:session): session opened for user gdm by (uid=0)
Oct 21 16:36:34 task1-simonas polkitd(authority=local): Registered Authentication Agent for unix-session:c4 (system bus name :1.237 [/usr/bin/gnome-shell], object
path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Oct 21 16:36:46 task1-simonas gdm-password]: gkr-pam: unable to locate daemon control file
Oct 21 16:36:46 task1-simonas gdm-password]: gkr-pam: stashed password to try later in open session
Oct 21 16:36:46 task1-simonas gdm-password]: pam_unix(gdm-password:session): session opened for user tom by (uid=0)
Oct 21 16:36:46 task1-simonas systemd-logind[625]: New session 13 of user tom.
Oct 21 16:36:46 task1-simonas systemd: pam_unix(systemd-user:session): session opened for user tom by (uid=0)
Oct 21 16:36:46 task1-simonas gdm-password]: gkr-pam: gnome-keyring-daemon started properly and unlocked keyring
Oct 21 16:36:48 task1-simonas gnome-keyring-daemon[5723]: The PKCS#11 component was already initialized
Oct 21 16:36:48 task1-simonas gnome-keyring-daemon[5723]: The Secret Service was already initialized
Oct 21 16:36:49 task1-simonas polkitd(authority=local): Registered Authentication Agent for unix-session:13 (system bus name :1.273 [/usr/bin/gnome-shell], object
path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Oct 21 16:36:53 task1-simonas gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session closed for user gdm
Oct 21 16:36:53 task1-simonas systemd-logind[625]: Session c4 logged out. Waiting for processes to exit.
Oct 21 16:36:53 task1-simonas systemd-logind[625]: Removed session c4.
Oct 21 16:36:54 task1-simonas polkitd(authority=local): Unregistered Authentication Agent for unix-session:c4 (system bus name :1.237, object path /org/freedesktop
/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8) (disconnected from bus)
Oct 21 16:36:57 task1-simonas dbus-daemon[592]: [system] Failed to activate service 'org.bluez': timed out (service_start_timeout=25000ms)
Oct 21 16:37:14 task1-simonas gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session opened for user gdm by (uid=0)
Oct 21 16:37:14 task1-simonas systemd-logind[625]: New session c5 of user gdm.
Oct 21 16:37:14 task1-simonas systemd: pam_unix(systemd-user:session): session opened for user gdm by (uid=0)
Oct 21 16:37:17 task1-simonas polkitd(authority=local): Registered Authentication Agent for unix-session:c5 (system bus name :1.299 [/usr/bin/gnome-shell], object
path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Oct 21 16:37:25 task1-simonas gdm-password]: gkr-pam: unlocked login keyring
Oct 21 16:37:35 task1-simonas gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session closed for user gdm
Oct 21 16:37:35 task1-simonas systemd-logind[625]: Session c5 logged out. Waiting for processes to exit.
Oct 21 16:37:35 task1-simonas systemd-logind[625]: Removed session c5.
Oct 21 16:37:35 task1-simonas polkitd(authority=local): Unregistered Authentication Agent for unix-session:c5 (system bus name :1.299, object path /org/freedesktop
/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8) (disconnected from bus)
Oct 21 16:37:41 task1-simonas dbus-daemon[592]: [system] Failed to activate service 'org.bluez': timed out (service_start_timeout=25000ms)
```

I also checked and verified that kernel messages are being logged:

`tail -f /var/log/kern.log`

```
root@task1-simonas:~# tail -f /var/log/kern.log
Oct 21 16:36:55 task1-simonas kernel: [ 6694.859253] 13:36:55.966153 main      cannot aquire pidfile /home/tom/.vboxclient-draganddrop-tty3-control.pid, exitting
Oct 21 16:36:56 task1-simonas kernel: [ 6694.943399] 13:36:56.050428 main      cannot aquire pidfile /home/tom/.vboxclient-hostversion-tty3-control.pid, exitting
Oct 21 16:36:56 task1-simonas kernel: [ 6694.984717] 13:36:56.091787 main      cannot aquire pidfile /home/tom/.vboxclient-vmsvga-session-tty3-control.pid, exittin
g
Oct 21 16:37:15 task1-simonas kernel: [ 6713.945415] rfkill: input handler enabled
Oct 21 16:37:18 task1-simonas kernel: [ 6717.360564] rfkill: input handler disabled
Oct 21 16:37:19 task1-simonas kernel: [ 6718.612687] 13:37:19.719578 SHCLX11   Shared Clipboard: Converting X11 format index 0x3 to VBox format 0x1 failed, rc=VERR
_SHCLPB_NO_DATA
Oct 21 16:37:19 task1-simonas kernel: [ 6718.617035] 13:37:19.723862 SHCLX11   Shared Clipboard: Reading clipboard data in format 0x1 from host failed with VERR_SH
CLPB_NO_DATA
Oct 21 16:37:19 task1-simonas kernel: [ 6718.618002] 13:37:19.725208 SHCLX11   Shared Clipboard: Requesting data in format 0x1 from host failed with VERR_SHCLPB_NO
_DATA
Oct 21 16:37:25 task1-simonas kernel: [ 6724.593284] rfkill: input handler enabled
Oct 21 16:37:25 task1-simonas kernel: [ 6724.595659] rfkill: input handler disabled
```

Take ownership of the selected files, which were created by other employees

`ls -l` command lists out files with owners provided and rights.

```
godmode@task1-simonas:~$ ls -l /home/managers
total 8
-rw-rwxr--+ 1 anthony managers 5 spal.  19 15:15 anthony_file.txt
-rw-rwxr--+ 1 boss    ceo      5 spal.  19 16:37 ceo_file_for_managers.txt
-rw-rwxr--+ 1 boss    ceo      0 spal.  19 16:42 ceo_file.txt
```

`chown` command has a syntax `chown <user>:<group> <file or directory path to be owned>.`

```
godmode@task1-simonas:~$ sudo chown godmode: /home/managers/anthony_file.txt
godmode@task1-simonas:~$ ls -l /home/managers
total 8
-rw-rwxr--+ 1 godmode godmode 5 spal.  19 15:15 anthony_file.txt
-rw-rwxr--+ 1 boss    ceo     5 spal.  19 16:37 ceo_file_for_managers.txt
-rw-rwxr--+ 1 boss    ceo     0 spal.  19 16:42 ceo_file.txt
godmode@task1-simonas:~$
```

Implement Logging and Sessions recording for Privileged users.

First, I installed `auditd` :

```
root@task1-simonas:~# apt install auditd
```

Then, I ensured that `auditd` is running and enabled on boot:

```
systemctl start auditd
systemctl enable auditd
```

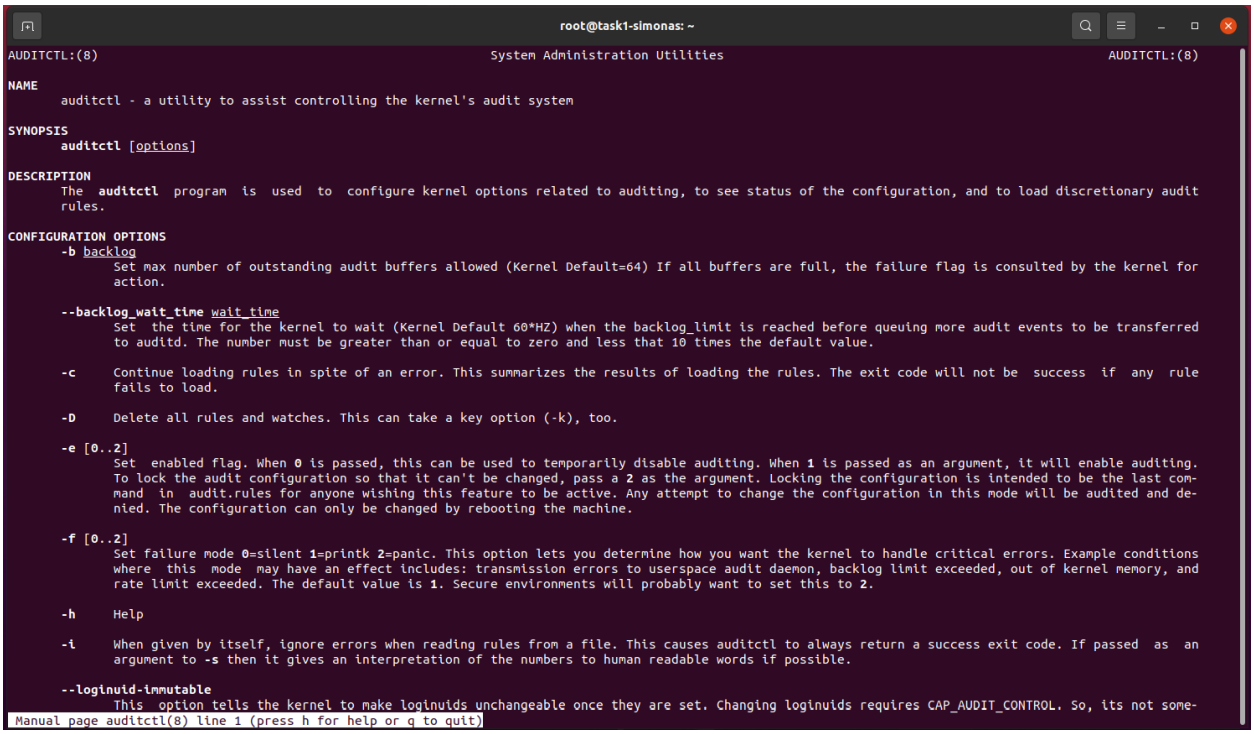
```
root@task1-simonas:~# systemctl start auditd
root@task1-simonas:~# systemctl enable auditd
Synchronizing state of auditd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable auditd
```

I checked the status to ensure auditd is active and running without issues:

```
systemctl status auditd
```

```
root@task1-simonas:~# systemctl status auditd
● auditd.service - Security Auditing Service
     Loaded: loaded (/lib/systemd/system/auditd.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-10-21 23:17:22 EEST; 21min ago
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
   Main PID: 7712 (auditd)
      Tasks: 2 (limit: 2247)
     Memory: 384.0K
     CGroup: /system.slice/auditd.service
             └─7712 /sbin/auditd

spal. 21 23:17:22 task1-simonas augenrules[7730]: backlog_wait_time 15000
spal. 21 23:17:22 task1-simonas augenrules[7730]: enabled 1
spal. 21 23:17:22 task1-simonas augenrules[7730]: failure 1
spal. 21 23:17:22 task1-simonas augenrules[7730]: pid 7712
spal. 21 23:17:22 task1-simonas augenrules[7730]: rate_limit 0
spal. 21 23:17:22 task1-simonas augenrules[7730]: backlog_limit 8192
spal. 21 23:17:22 task1-simonas augenrules[7730]: lost 0
spal. 21 23:17:22 task1-simonas augenrules[7730]: backlog 4
spal. 21 23:17:22 task1-simonas augenrules[7730]: backlog_wait_time 0
spal. 21 23:17:22 task1-simonas systemd[1]: Started Security Auditing Service.
root@task1-simonas:~#
```

Then, I entered `man auditctl` command to check possible parameters in auditd rules



For rules I used this and this tutorial and wrote

```
-a always,exit -F arch=b64 -S execve -F euid=0 -F auid>=1000 -F auid!=-1 -F key=subcom
```

`-a always,exit` specifies that we want to log whenever a system call (execve in this case) is made and exits (when the command is completed).

`-F arch=b64` filters the logs to only capture 64-bit architecture system calls.

`-S execve` indicates we are monitoring the execve system call, which is responsible for executing commands.

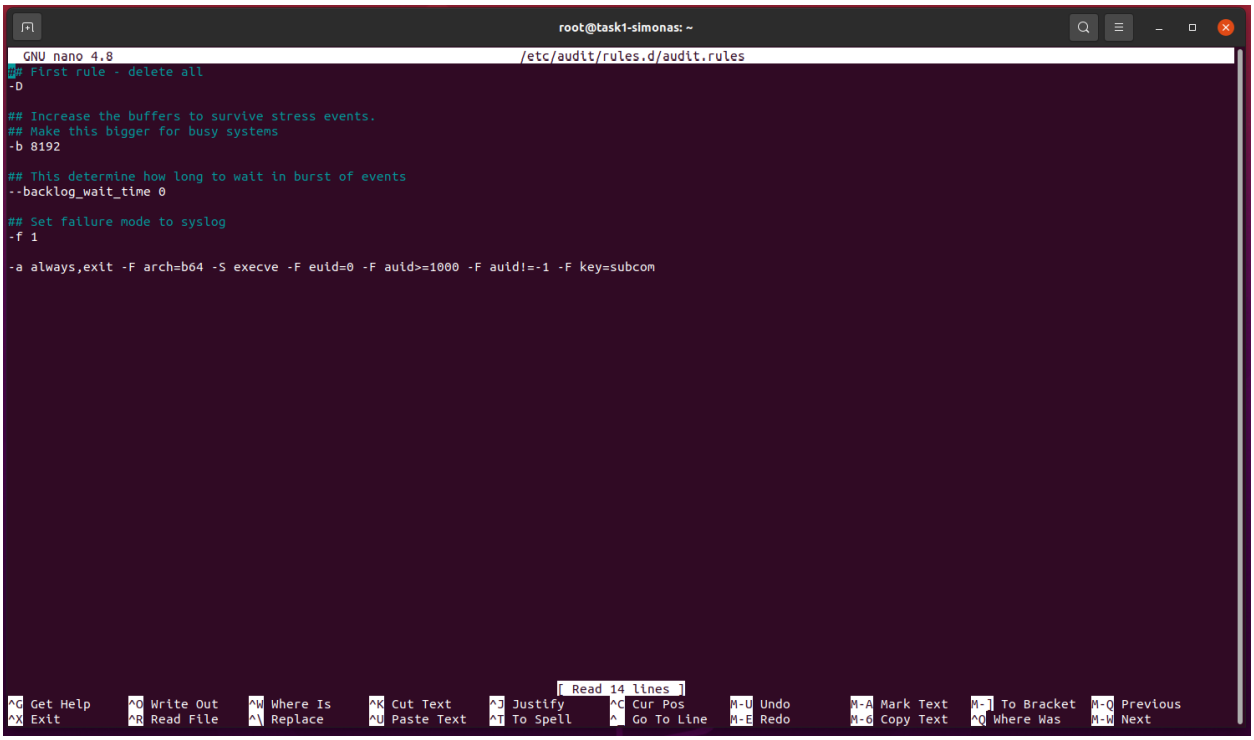
`-F euid=0` filters to log only commands where the effective user ID (euid) is 0, which means root or sudo-level privileges are being used.

`-F auid>=1000` ensures the rule only applies to regular users (non-system users), since system accounts usually have a UID below 1000. This filters out actions by system processes.

`-F auid!=-1` ensures that logging commands run by actual logged-in users. AUID (Audit User ID) of -1 indicates actions not associated with a user (such as daemons or kernel tasks).

`-F key=subcom` assigns a key (subcom) to the logs, making it easier to search for these events later.

I open the audit rules file with `nano /etc/audit/rules.d/audit.rules` and wrote the rules there:



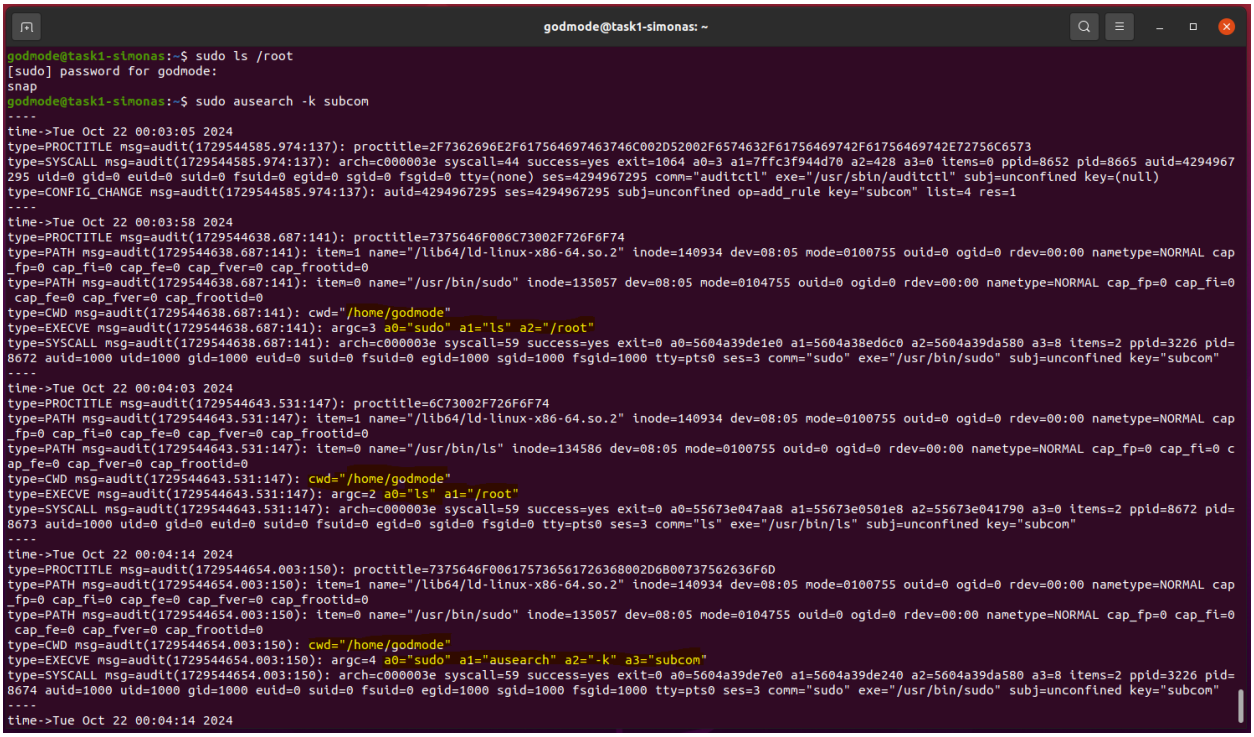
Then, I restarted auditd service:

```
systemctl restart auditd
```

```
root@task1-simonas:~# systemctl restart auditd
```

Then, I tested out by writing **`sudo ls /root`** command and I checked the the audit logs for entries matching the key subcom:

```
sudo ausearch -k subcom
```

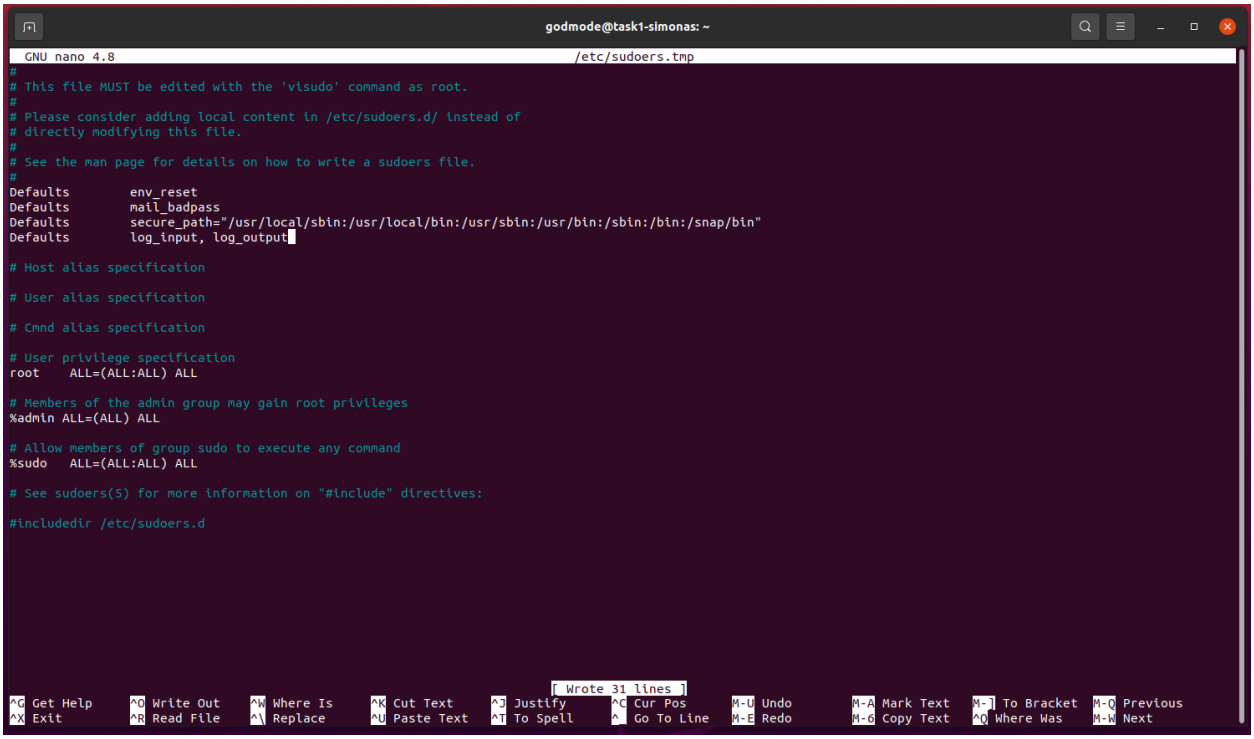


Sessions recording

`sudo visudo` command is used to safely edit the sudoers file, which controls permissions for users who can run commands with sudo privileges.

```
godmode@task1-simonas:~$ sudo visudo
```

`Defaults log_input, log_output` in the sudoers file enables logging of both input commands and output for any command run with sudo.



I tested it out writing `sudo ls /root` command

```
godmode@task1-simonas:~$ sudo ls /root
snap
```

Logs are saved in `/var/log/sudo-io` folder:

```
godmode@task1-simonas:~$ sudo ls /var/log/sudo-io
00  seq
godmode@task1-simonas:~$ sudo ls /var/log/sudo-io/00
00
godmode@task1-simonas:~$ sudo ls /var/log/sudo-io/00/00
01  02  03  04  05
godmode@task1-simonas:~$ sudo ls /var/log/sudo-io/00/00/01
log  stderr  stdin  stdout  timing  ttyin  ttyout
godmode@task1-simonas:~$ sudo cat /var/log/sudo-io/00/00/01/log
1729545148:godmode:root::/dev/pts/0:45:163
/home/godmode
/usr/bin/ls /root
godmode@task1-simonas:~$
```

Use/Implement specified commands/tools/methods

As I've implemented **chmod**, **setfacl**, **getfacl**, **default ACL**, **chown**, **passwd** commands in the previous tasks, I've left with **SUID**, **GUID** and **chattr**.

SUID

First, I found what location is for "su" (switch user) command

```
which su
```

```
godmode@task1-simonas:~$ which su
/usr/bin/su
```

Then, I verified that the su command has the SUID bit set by checking its permissions:

```
ls -l /bin/su
```

The s in the owner's execute position confirms that the SUID bit is set, allowing regular users to execute su with root privileges.

```
godmode@task1-simonas:~$ ls -l /usr/bin/su
-rwsr-xr-x 1 root root 67816 vas.    7  2022 /usr/bin/su
```

To remove the SUID bit from su, I used the following command as a user with sudo privileges:

```
sudo chmod u-s /bin/su
```

```
godmode@task1-simonas:~$ sudo chmod u-s /usr/bin/su
[sudo] password for godmode:
```

I checked the permissions of the su command again to confirm the SUID bit was removed:

```
ls -l /bin/su
```

```
godmode@task1-simonas:~$ ls -l /usr/bin/su
-rwxr-xr-x 1 root root 67816 vas.    7  2022 /usr/bin/su
```

Lastly, I tried to switch user to tom to test the su command without sudo:

```
su tom
```

```
godmode@task1-simonas:~$ su tom
Password:
su: Authentication failure
```

To set SUID again this command is used:

```
sudo chmod u+s /usr/bin/su
```

```
godmode@task1-simonas:~$ sudo chmod u+s /usr/bin/su
godmode@task1-simonas:~$ ls -l /usr/bin/su
-rwsr-xr-x 1 root root 67816 vas.    7  2022 /usr/bin/su
godmode@task1-simonas:~$ su tom
Password:
$ exit
godmode@task1-simonas:~$
```

GUID

I used GUID to ensure that any file created in a directory inherits the group ownership of the directory, not the user's primary group. For example, when user boss creates new file in `/home/managers` folder it belongs to ceo group:

```
godmode@task1-simonas:~$ su boss
Password:
$ touch /home/managers/new_boss_file.txt
$ getfacl /home/managers/new_boss_file.txt
getfacl: Removing leading '/' from absolute path names
# file: home/managers/new_boss_file.txt
# owner: boss
# group: ceo
user::rw-
group::r--
other::r--

$ rm /home/managers/new_boss_file.txt
$ exit
```

I used this command:

```
chmod g+s /home/managers
```

This sets the GUID bit on the `/home/managers` directory. Now, any files created within the `/home/managers` folder will automatically belong to the managers group, regardless of who creates the file.

```
root@task1-simonas:~# chmod g+s /home/managers
```

To test it out I created file as user `boss` from group `ceo` and checked its rights:

```
godmode@task1-simonas:~$ su boss
Password:
$ touch /home/managers/new_boss_file.txt
$ getfacl /home/managers/new_boss_file.txt
getfacl: Removing leading '/' from absolute path names
# file: home/managers/new_boss_file.txt
# owner: boss
# group: managers
user::rw-
group::r--
other::r--
```

Chattr

I used the `chattr` command to make a file immutable to prevent unauthorized users or even root from modifying or deleting important files. I created a file as `godmode` to my secure folder:

```
godmode@task1-simonas:~$ touch /home/godmode/secure/very_important_file_for_system
```

Then I used `chattr` command `+i` for file to make file immutable:

```
chattr +i <filename>
```

```
godmode@task1-simonas:~$ sudo chattr +i /home/godmode/secure/very_important_file_for_system
```

I tried to delete it but operation was not permitted:

```
godmode@task1-simonas:~$ rm /home/godmode/secure/very_important_file_for_system
rm: cannot remove '/home/godmode/secure/very_important_file_for_system': Operation not permitted
```

Also, if used on folder with **+i** flag – no files can be created, deleted, or renamed within the folder, if used with **+a** – restricts deletion or modification of files, but allows creating and modifying them (for logs, for example).