htl donaustadt
Donaustadtstraße 45
1220 Wien

Abteilung: Informationstechnologie
Schwerpunkt: Netzwerktechnik

Secure data storage on Windows

Laboratory protocol Exercise 8: Secure data storage on Windows

Figure 1: Grouplogo

Subject: ITSI
Class: 3AHITN
Name: Stefan Fürst, Justin Tremurici
Gruppenname/Nummer: todo/12
Supervisor: SPAC, ZIVK
Exercise dates: 14.02.2025 | 21.02.2025 | 28.02.2025 | 7.02.2025
Submission date: 14.3.2025

Contents

1 Task definition

1.1 Task Overview

The goal of this exercise is to set up a secure and structured data storage system on a Windows Server, ensuring proper access control and encryption. The tasks include installing the operating system, configuring users and groups, setting up a folder structure, and securing access with permissions.
First, `Windows Server 2019` or newer must be installed in a virtualized environment, with the `hostname` and user accounts configured according to a naming convention. A structured folder system should be created based on an assigned fictional company, categorizing data logically and including sample files.
User management involves defining necessary accounts, organizing them into `organizational groups`, and enforcing a consistent naming scheme. Permissions must be applied using `NTFS` security settings, restricting access appropriately. `Network shares` need to be configured to allow remote access while maintaining security through controlled sharing settings.
The storage system must be optimized by creating a new `partition`, migrating data, and applying `BitLocker` encryption. All configurations should be verified, and documentation is required throughout the process. `PowerShell` automation is encouraged where applicable.[1]

2 Summary

This task was fully automated using two `PowerShell` scripts. The first script set up the environment by configuring the necessary system settings and creating a `Scheduled Task` using `New-ScheduledTask` and `Register-ScheduledTask` to execute the second script at predefined intervals. The second script performed all required operations, including creating and managing `users`, `directories`, `files`, and their `permissions`.
To manage users, the script utilized commands such as `New-LocalUser` to create users, `Add-LocalGroupMember` to assign them to groups, and `Set-LocalUser` to modify user settings. The script dynamically generated random user accounts and assigned them to groups based on predefined logic.
Organizational groups and security groups were structured to align with the fictional company scenario, `BuildSmart BIM`, a digital building company specializing in `architecture`, `material lists`, and `time plans`.
For managing directories and files, commands like `New-Item` were used to create directory structures, and `icacls` was used to configure `NTFS permissions` for securing access.
The partitioning of the drive was accomplished using `Resize-Partition` to shrink the primary partition and `New-Partition` to create a new `B:` partition, followed by `Format-Volume` to prepare it for use. The existing directory structure was then migrated using `Move-Item`.
To enable file sharing, `New-SmbShare` was employed to create network shares. Additionally, the script encrpyed the partition using `Enable-BitLocker`.
Through this approach, the entire process was streamlined and executed automatically, significantly reducing manual effort while maintaining a structured and secure environment.

[1] This task definition was generated using ChatGPT.

3 Exercise Execution

3.1 Introduction

This entire exercise was automated with a script that can be invoked with a single command. It is designed to be used after setting up a fresh install of a new version of Windows Server. This script will complete the entire exercise automatically.
Note: For the script to work, the language of the Windows Server installation must be set to English, since user and group names are localized with no available aliases [1].
I will go through the execution of the script, explaining everything it does and how it works.
To run it, simply log into the Administrator account, press **Windows + R** to open the Run dialog, and paste the following command:

```
powershell.exe iwr https://tinyurl.com/mr234bdr | iex
```

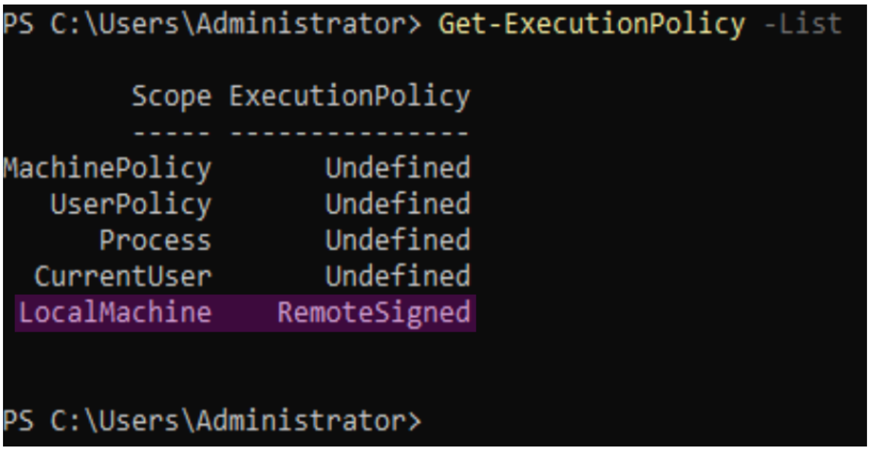
This runs `powershell.exe` and uses the abbreviation `iwr`, which stands for `Invoke-WebRequest`. This command makes a web request to the given link, which in this case is `https://tinyurl.com/mr234bdr`. This URL redirects to the raw file from my GitHub repository, where the first of two PowerShell scripts is downloaded [2]. **Important:** Whenever you run a command like this, always open the URL in your web browser first to verify its contents. Check whether the script has any red flags, such as obfuscation or suspicious behavior that it is not intended to perform. The downloaded script is then piped into `iex`, which is short for `Invoke-Expression`. This executes the output from `stdin`, which, in this case, is the contents of the script [3, 4].

3.2 Explaining the first script

3.2.1 Changing the execution policy

The first line of the `setup.ps1` script is the following:
`Set-ExecutionPolicy RemoteSigned -Scope LocalMachine -Force`

This sets the `ExecutionPolicy` parameter to the `RemoteSigned` policy. The `Scope` parameter specifies the default scope value in this command as `LocalMachine`. The `-Force` parameter is used to suppress all confirmation prompts.So that the second script that is downloaded in this script is allowed to run on this device.[5]
This can be verified by running the following command as shown in Figure 2:
`Get-ExecutionPolicy -List`



```
PS C:\Users\Administrator> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine    RemoteSigned


PS C:\Users\Administrator>
```

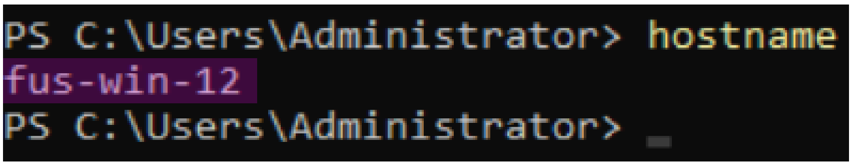
Figure 2: Listing the execution policies of the local machine

Installing BitLocker

`Install-WindowsFeature BitLocker` installs the Windows feature `BitLocker`, which is used for drive encryption and will be needed later in the exercise. It is included in the setup because its installation requires a reboot [6, 7].

Changing the Hostname

`Rename-Computer -NewName "fus-win-12" -Force` changes the hostname of the computer to `fus-win-12`. The `-Force` parameter ensures the command runs non-interactively. This requires a reboot as well and is therefore included in the setup script [8].
The change of the hostname can be veryfied by using the hostname command as displayed in Fiugre 3.



```
PS C:\Users\Administrator> hostname
fus-win-12
PS C:\Users\Administrator> _
```

Figure 3: Verifying the hostname change

Downloading the second script

To download the second script, `Invoke-WebRequest` is used again with a `url` and `dest` variable to store the desired URL and destination file. The last line executes the `Invoke-WebRequest` command again.

```
#setting the desired url
$url = "https://raw.githubusercontent.com/Stefanistkuhl/obsidianschule
        /refs/heads/main/3.Klasse/itsi/aufgaben/windoof/script.ps1"
#setting the destination file
$dest = "C:\Users\Administrator\script.ps1"
#invoking the webrequest
Invoke-WebRequest -Uri $url -OutFile $dest
```

3.2.2 Enabling Remote Desktop

For easier management and testing of users, we choose to enable Remote Desktop.
This is done by changing a registry key, `fDenyTSConnections`, and setting its value to 0, allowing remote connections.
Furthermore, a firewall rule needs to be enabled to allow Remote Desktop connections to be established with this server [9].

```
#changing the registry key
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control
        \Terminal Server' -name "fDenyTSConnections" -value 0
#allow rdp connections in the firewall
Enable-NetFirewallRule -DisplayGroup "Remote Desktop"
```

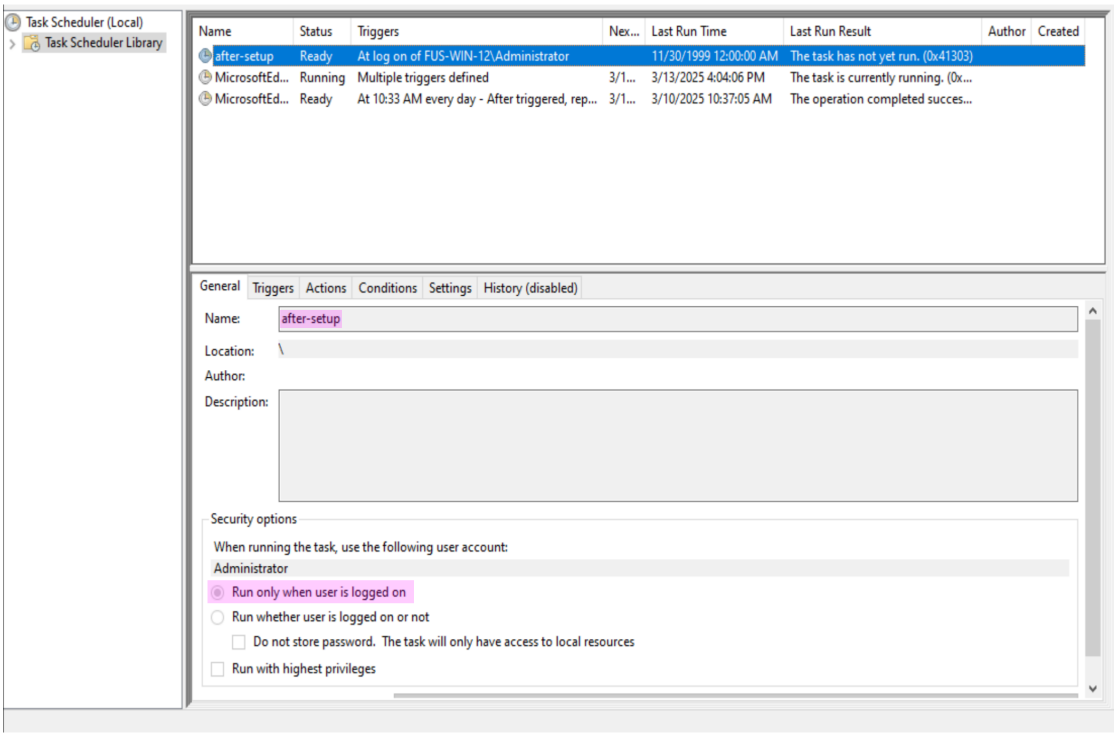
3.2.3 Creating a Scheduled Task

Since a restart is needed for the first part, some form of persistence is required for the second script to execute. For this, `Windows Scheduled Tasks` are used to execute the second script after a reboot with a set trigger.

The first line defines an action to be run, which is stored in the `Action` variable. The defined action executes PowerShell with the desired script file path as the argument.
The second line defines the trigger of the task, stored in the `Trigger` variable. The set trigger is configured to run once the user ”Administrator” logs in.
The third line defines the settings of the task, stored in the `Settings` variable. This is boilerplate code to create a settings set, which is required for the command.
The last line creates the task itself, sets the task name to `after-setup`, and assigns the previously created trigger, action, and settings variables [10].

```
#setting the desired action
$Action = New-ScheduledTaskAction -Execute "powershell.exe" -Argument
          "-file C:\Users\Administrator\script.ps1"
#setting the trigger
$Trigger = New-ScheduledTaskTrigger -AtLogon -User "Administrator"
#creating the setting set
$Settings = New-ScheduledTaskSettingsSet
#creating the task
Register-ScheduledTask -TaskName "after-setup" -Action $Action
          -Trigger $Trigger -Settings $Settings
#reboot
Restart-Computer
```

Lastly, the server will be rebooted since the setup script is complete, and the second script will be invoked at login.
The created task can be viewed in Windows Task Scheduler, as demonstrated in Figure 4.



Name	Status	Triggers	Nex...	Last Run Time	Last Run Result	Author	Created
after-setup	Ready	At log on of FUS-WIN-12\Administrator		11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)		
MicrosoftEd...	Running	Multiple triggers defined	3/1...	3/13/2025 4:04:06 PM	The task is currently running. (0x...		
MicrosoftEd...	Ready	At 10:33 AM every day - After triggered, rep...	3/1...	3/10/2025 10:37:05 AM	The operation completed succes...		

Name: after-setup
Location: \
Author:
Description:

Security options
When running the task, use the following user account:
Administrator
● Run only when user is logged on
○ Run whether user is logged on or not
□ Do not store password. The task will only have access to local resources
□ Run with highest privileges

Figure 4: Inspecting the created task in task scheduler

3.2.4 The secound sript

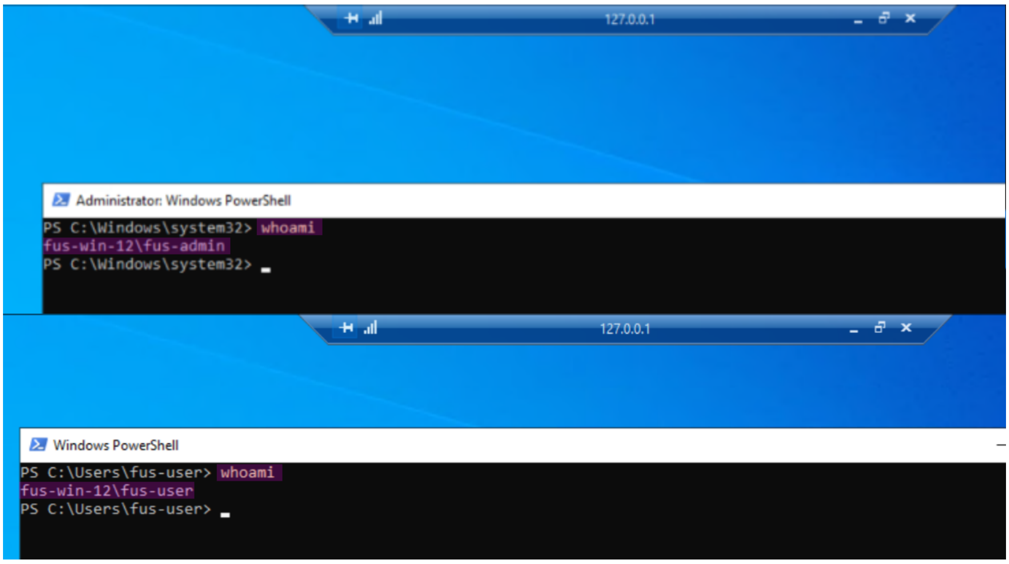
Now that the setup script has finished running, after logging in as the administrator again, the second script will be launched. Let's break it down as well from top to bottom.[2]

3.2.5 Creating Users and Adding Them to Groups

In this step, two new users will be created: `fus-admin` and `fus-user`. These serve as the administrator's low-privileged account and privileged account, respectively, in this scenario.
The first line of the snippet below generates a securely encrypted string from a set input string using the `-AsPlainText` parameter. This helps protect confidential plain text. The variable will be used for the passwords of all users. Obviously, hardcoding the password in the script is not secure, but it is fine for this example. This secure string will be used for every command where a password is required [11]. Using the `New-LocalUser` command, the two mentioned users are created. The `-Name` parameter is used to set the user's name, and the `-Password` parameter assigns the password to the previously generated secure string [12].
After this, the `fus-admin` user is added to the `Administrators` group using the `Add-LocalGroupMember` command to grant them administrator privileges. Meanwhile, the `fus-user` is added to the `Remote Desktop Users` group to allow remote desktop connections. The `-Group` and `-Member` parameters are used to specify the group and member to add [13].

```
$supersurepassword = ConvertTo-SecureString "rafi123_" -AsPlainText
New-LocalUser -Name 'fus-admin' -Password $supersurepassword
New-LocalUser -Name 'fus-user' -Password $supersurepassword
Add-LocalGroupMember -Group "Administrators" -Member fus-admin
Add-LocalGroupMember -Group "Remote Desktop Users" -Member fus-user
```

To test the users' functionality, I logged in as the two new users via Remote Desktop and used the `whoami` command to print the current user, as shown in Figure 5.



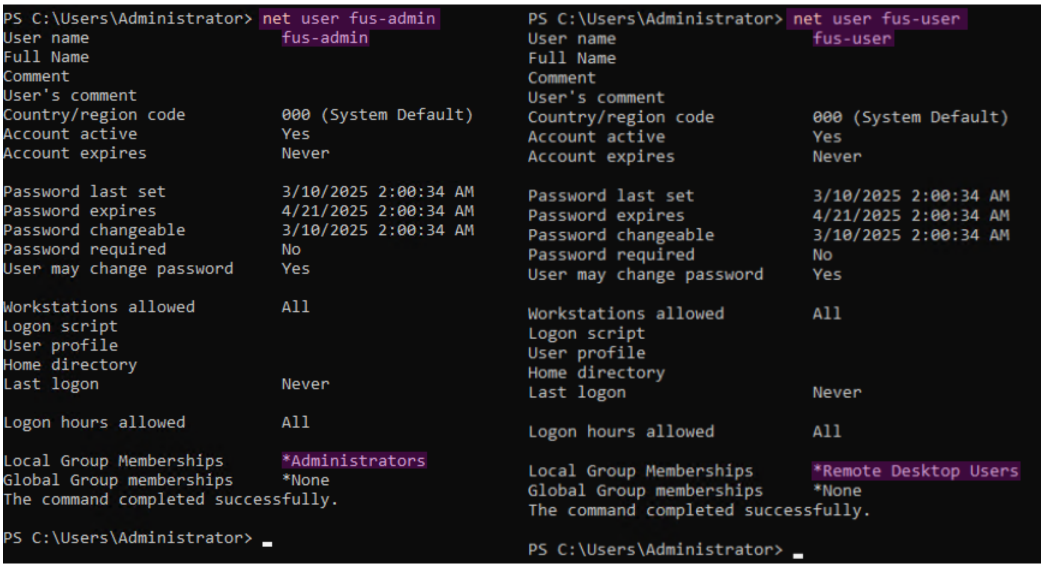
```
PS C:\Windows\system32> whoami
fus-win-12\fus-admin
PS C:\Windows\system32> _
```

```
PS C:\Users\fus-user> whoami
fus-win-12\fus-user
PS C:\Users\fus-user> _
```

Figure 5: Verifying the functionality of the two users

[2] Note that the sequence in the script does not match the order of tasks in the task definitions, but it makes more sense to break the script down and mention the tasks during it.

Lastly, to see the groups the users are part of, the `net user <user>` command can be used to print out information about a given user, as shown in Figure 6.



```
PS C:\Users\Administrator> net user fus-admin
User name                    fus-admin
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            3/10/2025 2:00:34 AM
Password expires             4/21/2025 2:00:34 AM
Password changeable          3/10/2025 2:00:34 AM
Password required            No
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *None
The command completed successfully.

PS C:\Users\Administrator> _
```

```
PS C:\Users\Administrator> net user fus-user
User name                    fus-user
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            3/10/2025 2:00:34 AM
Password expires             4/21/2025 2:00:34 AM
Password changeable          3/10/2025 2:00:34 AM
Password required            No
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships      *Remote Desktop Users
Global Group memberships     *None
The command completed successfully.

PS C:\Users\Administrator> _
```

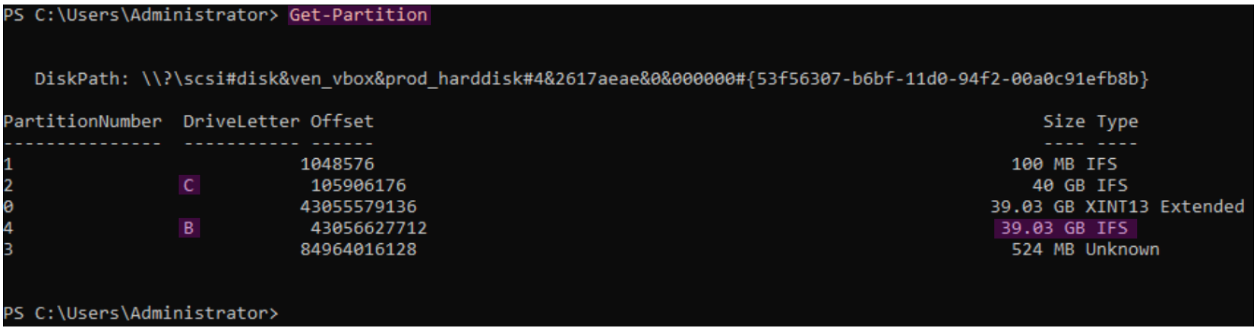
Figure 6: Printing information about the users

3.2.6 Resizing the Disk and Creating a New Partition

To resize the main disk, the `Resize-Partition` command is used. The disk to resize is specified using the `-DiskNumber` parameter, which in this case is disk 0 (the C: drive). The partition to resize is selected with the `-PartitionNumber` parameter, which in this case is partition 2, since partition 1 is the reserved Windows boot partition. Finally, the `-Size` parameter is used to set the new partition size, which is 40GB in this case [14]. Next, a new partition[3] has to be created using the `New-Partition` command. The `-DiskNumber` parameter specifies the same disk (disk 0), and the `-UseMaximumSize` parameter is used to allocate all the remaining free space for this partition. The `-DriveLetter` parameter assigns a drive letter (B:) to the new partition. The drive letter is what appears in File Explorer under "This PC" and is comparable to how partitions and drive names are handled in Linux, such as `/dev/sda`, `/dev/sda1`, `/dev/nvme0n1`, or `/dev/nvme0n1p1`.[16, 15] Once the volume is created, it must be formatted. The `Format-Volume` command is used for this purpose. The `-DriveLetter` parameter specifies the volume to format, and the `-FileSystem` parameter sets the desired file system, which in this case is `NTFS`, the default Windows file system. The `-AllocationUnitSize` parameter sets the allocation (or cluster) size. This defines the minimum space allocated for each file. While one allocation unit can store one file, a file may span multiple allocation units. This means that files smaller than the allocation size waste space. For `NTFS`, it is generally recommended to use a cluster size of `4096` bytes, as this allows storing files up to `16TB` while minimizing wasted space when storing small files [17, 18, 19].

```
Resize-Partition -DiskNumber 0 -PartitionNumber 2 -Size (40GB)
New-Partition -DiskNumber 0 -UseMaximumSize -DriveLetter B
Format-Volume -DriveLetter B -FileSystem NTFS -AllocationUnitSize 4096
```

Using the `Get-Partition` command, the partition table is printed, allowing us to observe the changes, as shown in Figure 7.



```
PS C:\Users\Administrator> Get-Partition


   DiskPath: \\?\scsi#disk&ven_vbox&prod_harddisk#4&2617aeae&0&000000#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}

PartitionNumber  DriveLetter Offset                                                                   Size Type
---------------  ----------- ------                                                                   ---- ----
1                           1048576                                                                100 MB IFS
2                C           105906176                                                              40 GB IFS
0                           43055579136                                                         39.03 GB XINT13 Extended
4                B           43056627712                                                        39.03 GB IFS
3                           84964016128                                                          524 MB Unknown


PS C:\Users\Administrator>
```

Figure 7: Printing the partition table

[3] Both the terms volume and partition are used in Windows. The difference is that a partition is a logical section of the disk, while a volume is created on a partition and formatted with a file system. There are multiple volume types depending on the use case. This volume will be formatted later with the desired file system [15].

3.2.7 Creating Directories

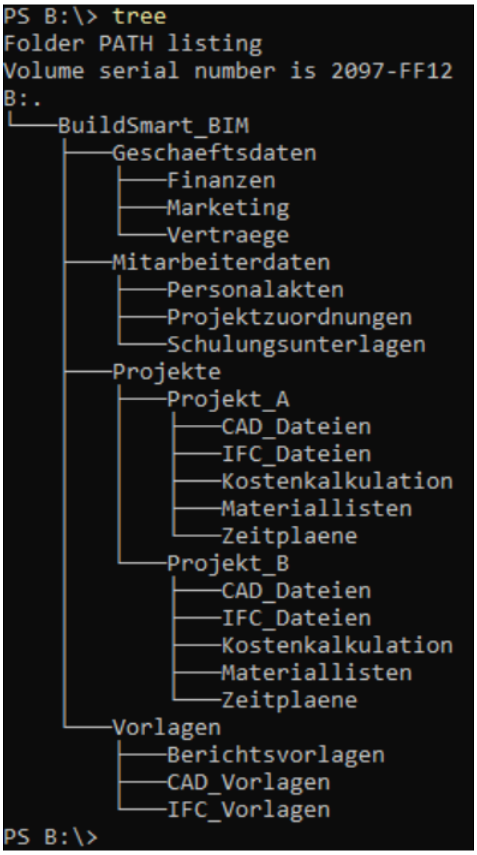
For this scenario, we are supposed to create a directory structure for the imaginary company's server.
As shown in Figure 8, the structure is visualized using the tree command. The structure consists of a main directory named BuildSmart_BIM, which represents the company's name. This directory contains four subdirectories:
- **Business**: Stores business-related data such as contracts, finances, etc.
- **Employees**: Contains employee-related data.
- **Projects**: Includes two example projects. Each project has subdirectories for CAD files and cost calculations.
- **Templates**: Holds templates for reports and CAD files.

Refer to Figure 8 for the complete directory structure.
The directories are created using the snippet below, where an array of directories to create is declared and then looped over to create the directories. Additionally, the root variable allows for easy modification of the location where the structure is created.
Besides the added comments, no further explanations are required for this section.[4]

```
#setting the root variable
$root = "B:\BuildSmart_BIM"

#checking if the root dir already exists
if (-not (Test-Path -Path $root)){
    New-Item -ItemType Directory -Path $root | Out-Null
    Write-Host "'$root' created"
} else{
    Write-Host "root '$root' already exists"
}

#creating the subdirs array
$subdirs = @(
    "$root\Projekte\Projekt_A\CAD_Dateien",
    ...
)

#looping over the array and creating the dirs
foreach ($dir in $subdirs){
    if (-not (Test-Path -Path $dir)){
        New-Item -ItemType Directory -Path $dir | Out-Null
        Write-Host "'$dir' created"
    } else{
        Write-Host "'$dir' already exists"
    }
}
```

[4] not the entire dir array is shown since it can be viewed on github in section 6 where all the used scripts are linked.

The directory structure that was just created can be viewed using the `tree` command, whose output is shown in Figure 8.



```
PS B:\> tree
Folder PATH listing
Volume serial number is 2097-FF12
B:.
└───BuildSmart_BIM
    ├───Geschaeftsdaten
    │   ├───Finanzen
    │   ├───Marketing
    │   └───Vertraege
    ├───Mitarbeiterdaten
    │   ├───Personalakten
    │   ├───Projektzuordnungen
    │   └───Schulungsunterlagen
    ├───Projekte
    │   ├───Projekt_A
    │   │   ├───CAD_Dateien
    │   │   ├───IFC_Dateien
    │   │   ├───Kostenkalkulation
    │   │   ├───Materiallisten
    │   │   └───Zeitplaene
    │   └───Projekt_B
    │       ├───CAD_Dateien
    │       ├───IFC_Dateien
    │       ├───Kostenkalkulation
    │       ├───Materiallisten
    │       └───Zeitplaene
    └───Vorlagen
        ├───Berichtsvorlagen
        ├───CAD_Vorlagen
        └───IFC_Vorlagen
PS B:\>
```

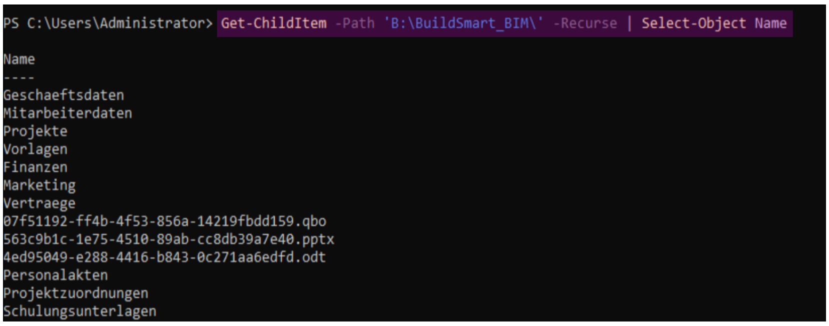
Figure 8: Printing the directory structure

3.2.8 Populating the Directories

To populate the directories, an array containing the subdirectories was created along with a file type map. The map's key corresponds to the subdirectory name, and the value is an array of possible file type extensions. These file extensions were generated using AI and represent file formats that would typically be used in those directories.
The script iterates over each directory, creates a random file with an appropriate file type, and names it using a randomly generated GUID. Additionally, the file's contents are set to a random GUID.

```
#creating the subdirectories array
$subdirs = @(
    "$root\Projekte\Projekt_A\CAD_Dateien",
    ...
)

#creating the file type map
$fileTypeMap = @{
    "CAD_Dateien" = @(".dwg", ".CATPart", ".CATProduct", ...)
}

#iterating over the subdirectories
foreach ($dir in $subdirs){
    # Getting the last part of the current directory
    $currentDirName = Split-Path -Path $dir -Leaf
    # Checking if the directory matches a key in the map
    if ($fileTypeMap.ContainsKey($currentDirName)){
        # Generating a random string for the filename
        $randomString = [System.Guid]::NewGuid().ToString()
        # Getting the matching file types for that directory
        $matchingFileTypes = $fileTypeMap[$currentDirName]
        # Selecting a random file type from the array
        $randomValue = Get-Random -InputObject $matchingFileTypes
        # Constructing the file path for the new file
        $filePath = Join-Path -Path $dir -ChildPath "$randomString$randomValue"
        # Writing the random string to the file
        $randomString | Out-File -FilePath $filePath -Encoding UTF8
    }
}
```

To recursively list all the files that we created in the directory structure, the `Get-ChildItem` command is used with the `-Path` parameter set to the desired path and the `-Recurse` parameter enabled. The output is then piped into `Select-Object Name` to retrieve only the names of directories and files. A truncated Version of the output of this is shown in Figure 9.



```
PS C:\Users\Administrator> Get-ChildItem -Path 'B:\BuildSmart_BIM\' -Recurse | Select-Object Name

Name
----
Geschaeftsdaten
Mitarbeiterdaten
Projekte
Vorlagen
Finanzen
Marketing
Vertraege
07f51192-ff4b-4f53-856a-14219fbdd159.qbo
563c9b1c-1e75-4510-89ab-cc8db39a7e40.pptx
4ed95049-e288-4416-b843-0c271aa6edfd.odt
Personalakten
Projektzuordnungen
Schulungsunterlagen
```

Figure 9: Recursively listing the files

3.2.9 Creating Users and Groups

To manage users in this fictitious company, both organizational and security groups will be created. Organizational groups are used to categorize users based on their department, while security groups are used to control access permissions. For example, users can be added to a security group related to a specific project to grant them read or write access to certain directories.
When managing a large number of users, it is important to have a consistent naming scheme for both users and groups to ensure easier administration.

Naming Conventions
The naming conventions for groups and users are as follows:

- **Organizational groups**: Prefixed with `BIM-`, e.g., `BIM-Engineer`.
- **Security groups**: Prefixed with `SG_BIM-`, followed by the specific permission, e.g., `SG_BIM-Project_A_Read`.
- **Users**: Follow the format `BIM-<id>-<Surname>`, where:
  - The ID consists of 6 digits.
  - The first 2 digits correspond to the organizational group.
  - The last 4 digits uniquely identify the user.
  - Each new user receives an incremented number.

This naming convention limits the number of employees to 1,000, which is sufficient for a small business. More digits can be added in the future if necessary. [5] The mapping of organizational groups to their corresponding ID prefixes is as follows:

```
"Projektleiter"  = "01"
"Architekten"    = "02"
"Ingenieure"     = "03"
"Verwaltung"     = "04"
"Gaeste"         = "05"
```

[5] Since the maximum length of a Windows username is 20 characters, if a surname is too long, the username will be truncated accordingly.

Creating Groups

The relationship between security and organizational groups is illustrated in Figure 10 below.

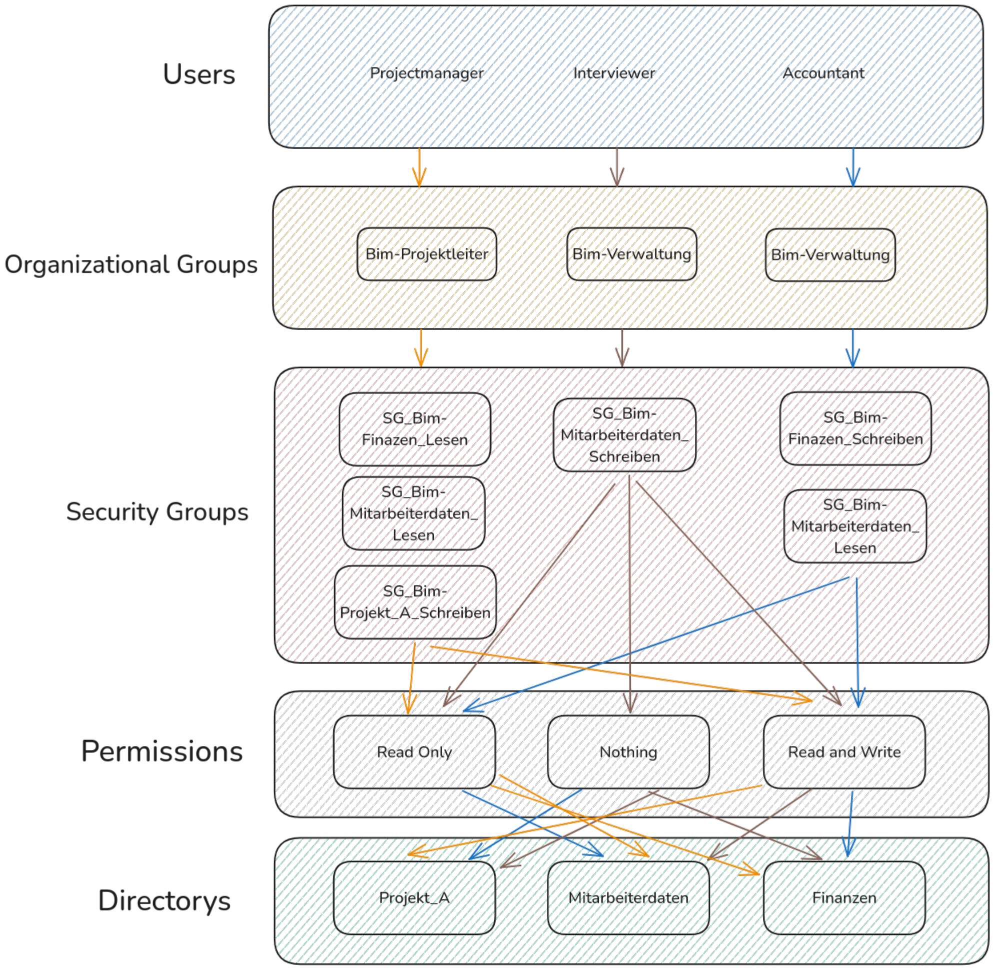


Figure 10: Diagramm of the users, groups and their permissions

To create the groups, a list of randomly generated surnames is downloaded and stored in the `name` variable using the `Get-Content` command. Additionally, two prefix variables, `prefix` and `secPrefix`, are declared for organizational and security groups, respectively.

Next, arrays containing the organizational and security group names are iterated over to create the groups using the `New-LocalGroup` command, with names constructed as `prefix+groupname` or `secPrefix+secgroupname`.

Creating Users

Creating users based on the defined naming scheme requires slightly more complex logic than creating groups. First, a mapping is created where the keys are organizational group names and the values are their corresponding numeric prefixes.

A `for-each` loop iterates over the `name` variable. At the start of each iteration:

1. A random element from the `groupnames` array is selected using `Get-Random`.
2. The `groupName` is then constructed using `prefix + randomElement`.
3. The `id_department` variable is set using the `idMap` dictionary with `randomElement` as the key.
4. The `id_idx` variable is generated as a formatted string using `"0:0000"` and the `-f` parameter, ensuring a four-digit format (e.g., `69` becomes `0069`).
5. The `fullID` is constructed by concatenating `id_department` and `id_idx`.
6. The full username is created as `prefix + fullID + " " + name`.
7. If the username exceeds 20 characters, it is truncated.

The user is then created using the `Add-LocalUser` command with the generated username and the previously declared `supersecurepassword` variable.

Lastly, the user is added to both the `Remote Desktop Users` group and the randomly chosen organizational group (`groupName`).

To assign users to the appropriate security groups, a switch-case statement is used to define access relationships based on the diagram in Figure 10 above. Additionally, a random factor determines whether users receive additional read access to other projects or documents.

Below is a shortened version of the relevant code for creating the groups, excluding boilerplate details:

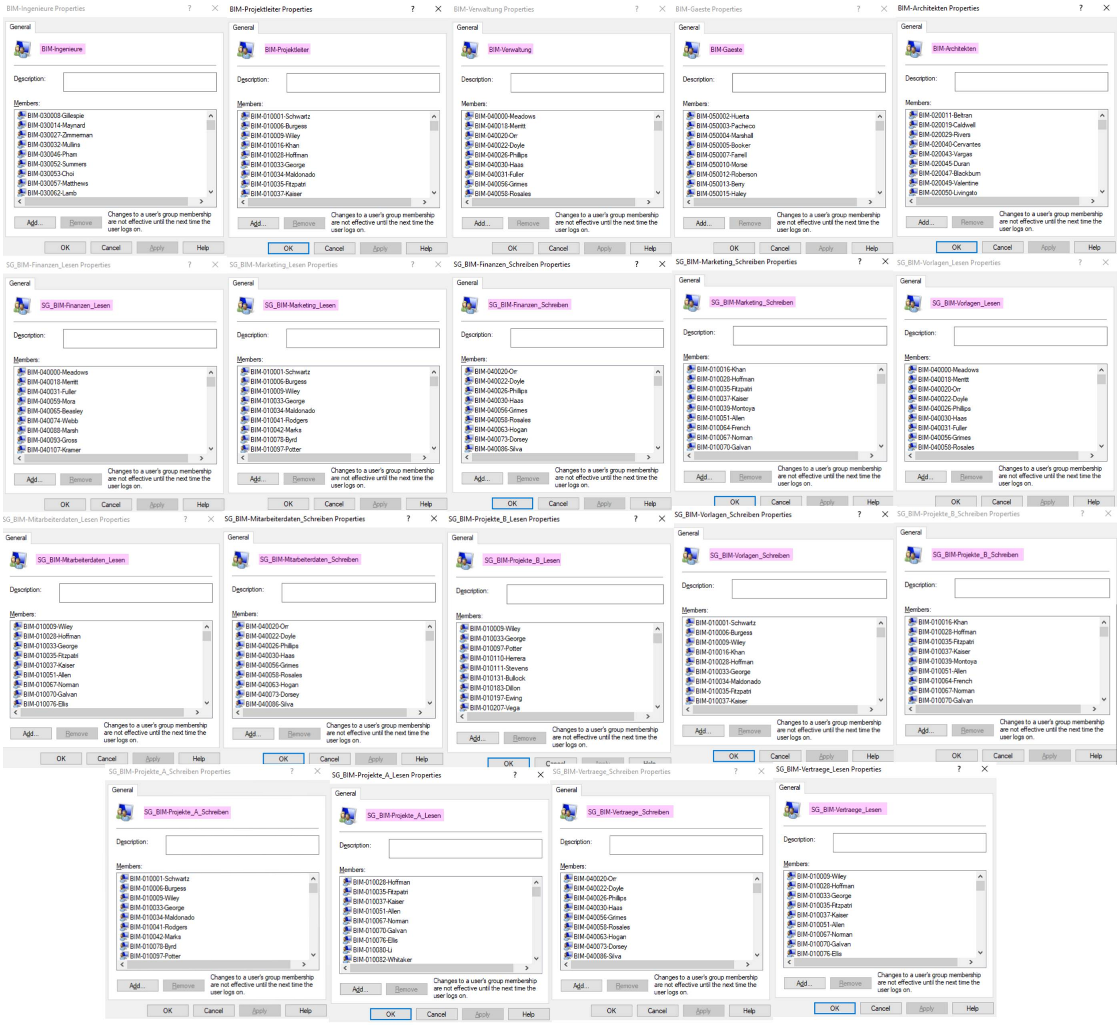
```
#url of the file with sirnames
$url = "https://raw.githubusercontent.com/Stefanistkuhl/obsidianschule/
        refs/heads/main/3.Klasse/itsi/aufgaben/windoof/sirnames.txt"
#destination file
$dest = "C:\Users\Administrator\sirnames.txt"
Invoke-WebRequest -Uri $url -OutFile $dest
#reading the file
$names = Get-Content -Path @($dest)
#setting the prefixes
$prefix = "BIM-"
$secPrefix = "SG_BIM-"

$groupnames = (
    "Projektleiter",
    ...
)

$secGroupNames = (
    "Projekte_A_Lesen",
    ...
)

#creating organizational groups
foreach ($grpname in $groupnames){
    $name = $prefix + $grpname
    New-LocalGroup -Name $name
}

#creating security groups
foreach ($secgrpname in $secGroupNames){
    $name = $secPrefix + $secgrpname
    New-LocalGroup -Name $name
}
```

Below is a shortened version of the relevant code for creating users and assigning them to the correct groups, excluding boilerplate details.

```
#creating the id map to for the id of the groups
$idMap = @{
    "Projektleiter" = "01"
    ...
}
$idx = 0
$fullID = ""

#itorating over the names
foreach($name in $names){
    $username = ""
    #getting a randoom element of the organizational groups
    $randomElement = $groupnames | Get-Random
    #setting the full groupname for the current element
    $groupName = $prefix + $randomElement
    #getting the id for set group
    $id_department = $idMap[$randomElement]
    #formating the id for the index of the user
    $id_idx = "{0:0000}" -f $idx
    #constructing the full id
    $fullID = $id_department + $id_idx
    #constructing the full username
    $userName = $prefix + $fullID + "-" + $name
    #truncating usernames longer than 20 characters
    if ($userName.Length -gt 20){
        $userName = $userName.Substring(0, 20)
    }
    #creating the user
    New-LocalUser -Name $userName -Password $supersurepassword
    #adding the user to the correct organizational group
    Add-LocalGroupMember -Group $groupName -Member $userName
    #adding remote desktop access for the user
    Add-LocalGroupMember -Group "Remote Desktop Users" -Member $username

    switch ($randomElement){
        "Projektleiter"{
            if ((Get-Random -Maximum 2) -eq 0){
                #add to the predefined security groups
                if ((Get-Random -Maximum 2) -eq 0){
                        #add to addtional read security groups
                }
            } else{
                #add to the opposite predefined security groups
                if ((Get-Random -Maximum 2) -eq 0){
                        #add to opposite addtional read security groups
                }
        #rest of the cases for the other groups
    }
    $idx++
}
```

Verifying the Creation of users and groups

To show all the groups and users, I opened Computer Management, navigated to Local Users and Groups → Groups, and then opened the preview for every single group, which are all gathered together in Figure 11.



BIM-Ingenieure Properties
BIM-Ingenieure
Members:
BIM-030008-Gillespie
BIM-030014-Maynard
BIM-030027-Zimmerman
BIM-030032-Mullins
BIM-030046-Pham
BIM-030052-Summers
BIM-030053-Choi
BIM-030057-Matthews
BIM-030062-Lamb
Changes to a user's group membership are not effective until the next time the user logs on.

BIM-Projektleiter Properties
BIM-Projektleiter
Members:
BIM-010001-Schwartz
BIM-010006-Burgess
BIM-010009-Wiley
BIM-010016-Khan
BIM-010028-Hoffman
BIM-010033-George
BIM-010034-Maldonado
BIM-010035-Fitzpatri
BIM-010037-Kaiser
Changes to a user's group membership are not effective until the next time the user logs on.

BIM-Verwaltung Properties
BIM-Verwaltung
Members:
BIM-040000-Meadows
BIM-040018-Merritt
BIM-040020-Orr
BIM-040022-Doyle
BIM-040026-Phillips
BIM-040030-Haas
BIM-040031-Fuller
BIM-040056-Grimes
BIM-040058-Rosales
Changes to a user's group membership are not effective until the next time the user logs on.

BIM-Gaeste Properties
BIM-Gaeste
Members:
BIM-050002-Huerta
BIM-050003-Pacheco
BIM-050004-Marshall
BIM-050005-Booker
BIM-050007-Farrell
BIM-050010-Morse
BIM-050012-Roberson
BIM-050013-Berry
BIM-050015-Haley
Changes to a user's group membership are not effective until the next time the user logs on.

BIM-Architekten Properties
BIM-Architekten
Members:
BIM-020011-Beltran
BIM-020019-Caldwell
BIM-020029-Rivers
BIM-020040-Cervantes
BIM-020043-Vargas
BIM-020045-Duran
BIM-020047-Blackburn
BIM-020049-Valentine
BIM-020050-Livingsto
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Finanzen_Lesen Properties
SG_BIM-Finanzen_Lesen
Members:
BIM-040000-Meadows
BIM-040018-Merritt
BIM-040031-Fuller
BIM-040059-Mora
BIM-040065-Beasley
BIM-040074-Webb
BIM-040088-Marsh
BIM-040093-Gross
BIM-040107-Kramer
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Marketing_Lesen Properties
SG_BIM-Marketing_Lesen
Members:
BIM-010001-Schwartz
BIM-010006-Burgess
BIM-010009-Wiley
BIM-010033-George
BIM-010034-Maldonado
BIM-010041-Rodgers
BIM-010042-Marks
BIM-010078-Byrd
BIM-010097-Potter
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Finanzen_Schreiben Properties
SG_BIM-Finanzen_Schreiben
Members:
BIM-040020-Orr
BIM-040022-Doyle
BIM-040026-Phillips
BIM-040030-Haas
BIM-040056-Grimes
BIM-040058-Rosales
BIM-040063-Hogan
BIM-040073-Dorsey
BIM-040086-Silva
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Marketing_Schreiben Properties
SG_BIM-Marketing_Schreiben
Members:
BIM-010016-Khan
BIM-010028-Hoffman
BIM-010035-Fitzpatri
BIM-010037-Kaiser
BIM-010039-Montoya
BIM-010051-Allen
BIM-010064-French
BIM-010067-Norman
BIM-010070-Galvan
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Vorlagen_Lesen Properties
SG_BIM-Vorlagen_Lesen
Members:
BIM-040000-Meadows
BIM-040018-Merritt
BIM-040020-Orr
BIM-040022-Doyle
BIM-040026-Phillips
BIM-040030-Haas
BIM-040031-Fuller
BIM-040056-Grimes
BIM-040058-Rosales
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Mitarbeiterdaten_Lesen Properties
SG_BIM-Mitarbeiterdaten_Lesen
Members:
BIM-010009-Wiley
BIM-010028-Hoffman
BIM-010033-George
BIM-010035-Fitzpatri
BIM-010037-Kaiser
BIM-010051-Allen
BIM-010067-Norman
BIM-010070-Galvan
BIM-010076-Ellis
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Mitarbeiterdaten_Schreiben Properties
SG_BIM-Mitarbeiterdaten_Schreiben
Members:
BIM-040020-Orr
BIM-040022-Doyle
BIM-040026-Phillips
BIM-040030-Haas
BIM-040056-Grimes
BIM-040058-Rosales
BIM-040063-Hogan
BIM-040073-Dorsey
BIM-040086-Silva
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Projekte_B_Lesen Properties
SG_BIM-Projekte_B_Lesen
Members:
BIM-010009-Wiley
BIM-010033-George
BIM-010097-Potter
BIM-010110-Herrera
BIM-010111-Stevens
BIM-010131-Bullock
BIM-010183-Dillon
BIM-010197-Ewing
BIM-010207-Vega
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Vorlagen_Schreiben Properties
SG_BIM-Vorlagen_Schreiben
Members:
BIM-010001-Schwartz
BIM-010006-Burgess
BIM-010009-Wiley
BIM-010016-Khan
BIM-010028-Hoffman
BIM-010033-George
BIM-010034-Maldonado
BIM-010035-Fitzpatri
BIM-010037-Kaiser
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Projekte_B_Schreiben Properties
SG_BIM-Projekte_B_Schreiben
Members:
BIM-010016-Khan
BIM-010028-Hoffman
BIM-010035-Fitzpatri
BIM-010037-Kaiser
BIM-010039-Montoya
BIM-010051-Allen
BIM-010064-French
BIM-010067-Norman
BIM-010070-Galvan
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Projekte_A_Schreiben Properties
SG_BIM-Projekte_A_Schreiben
Members:
BIM-010001-Schwartz
BIM-010006-Burgess
BIM-010009-Wiley
BIM-010033-George
BIM-010034-Maldonado
BIM-010041-Rodgers
BIM-010042-Marks
BIM-010078-Byrd
BIM-010097-Potter
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Projekte_A_Lesen Properties
SG_BIM-Projekte_A_Lesen
Members:
BIM-010028-Hoffman
BIM-010035-Fitzpatri
BIM-010037-Kaiser
BIM-010051-Allen
BIM-010067-Norman
BIM-010070-Galvan
BIM-010076-Ellis
BIM-010080-Li
BIM-010082-Whitaker
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Vertraege_Schreiben Properties
SG_BIM-Vertraege_Schreiben
Members:
BIM-040020-Orr
BIM-040022-Doyle
BIM-040026-Phillips
BIM-040030-Haas
BIM-040056-Grimes
BIM-040058-Rosales
BIM-040063-Hogan
BIM-040073-Dorsey
BIM-040086-Silva
Changes to a user's group membership are not effective until the next time the user logs on.

SG_BIM-Vertraege_Lesen Properties
SG_BIM-Vertraege_Lesen
Members:
BIM-010009-Wiley
BIM-010028-Hoffman
BIM-010033-George
BIM-010035-Fitzpatri
BIM-010037-Kaiser
BIM-010051-Allen
BIM-010067-Norman
BIM-010070-Galvan
BIM-010076-Ellis
Changes to a user's group membership are not effective until the next time the user logs on.

Figure 11: Showing all the groups

3.2.10 Managing NTFS Permissions Using icacls

The most convenient way to change NTFS permissions via the command line is by using the icacls program. icacls has the ability to list, change, remove, back up, and restore NTFS permissions or change the owner of a file system object.[20, 21]
The permissions that will be assigned to the groups are Read and Write and Read and Execute for the read and write security groups, respectively.

Before setting these permissions, the Administrator group is assigned as the owner of the root directory using the following command. The /T option ensures the operation is applied recursively.[20]

```
icacls "$root" /setowner "Administrators" /T
```

Next, the permissions of the directory are reset to prevent inheritance issues. While these issues are unlikely to occur, this step is included as a precaution because mistakes in previous attempts led to incorrect permissions being applied. The following commands accomplish this:

```
# Reset permissions of the root directory
icacls "$root" /reset /T
# Remove inheritance and apply base permissions
icacls "$root" /inheritance:d /T
```

Then, permissions for the Users and Authenticated Users groups are recursively removed. This ensures that only explicitly set permissions are applied, avoiding any unnecessary inherited permissions. The /remove:g option removes all granted permissions for the specified group.

```
icacls "$root" /remove:g "Users" /T
icacls "$root" /remove:g "Authenticated Users" /T
```

Now, the Administrators group is granted full control over the root folder and all its subdirectories and files. The /grant:r option grants the specified permissions, replacing any previously granted ones. The following notation is used:

- OI (Object Inherit) – Files within the directory inherit the specified permissions.
- CI (Container Inherit) – Subdirectories inherit the specified permissions.
- F (Full Control) – Grants complete access to the targeted files and directories.

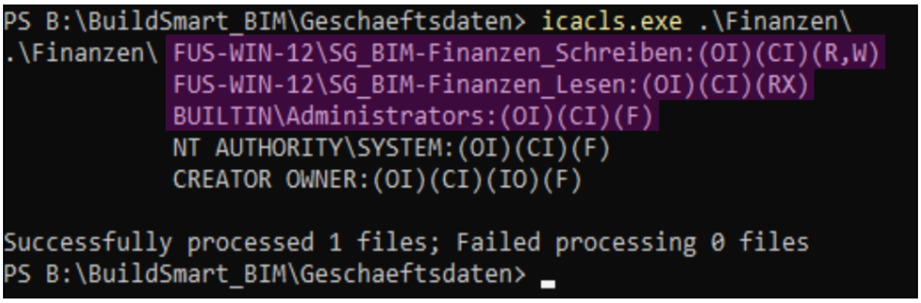
[20, 22]

```
# Grant Administrators full control to the root and all subdirectories
icacls "$root" /grant:r "Administrators:(OI)(CI)F" /T
```

The same logic is applied to all subdirectories and the security groups for read and write access. The only difference is that RX (Read and Execute) is used for read access, while RW (Read and Write) is used for write access. This process is repeated for the remaining groups and directories.

```
icacls "$root\Projekte\Projekt_A" /grant:r
       "$secPrefix-Projekte_A_Lesen:(OI)(CI)RX" /T
icacls "$root\Projekte\Projekt_A" /grant:r
       "$secPrefix-Projekte_A_Schreiben:(OI)(CI)RW" /T
```

To view the NTFS permissions we set earlier, we can use `icacls` followed by any directory or file name. This can be used to verify if we set the permissions correctly, as shown for one of the directories in Figure 12.



```
PS B:\BuildSmart_BIM\Geschaeftsdaten> icacls.exe .\Finanzen\
.\Finanzen\ FUS-WIN-12\SG_BIM-Finanzen_Schreiben:(OI)(CI)(R,W)
            FUS-WIN-12\SG_BIM-Finanzen_Lesen:(OI)(CI)(RX)
            BUILTIN\Administrators:(OI)(CI)(F)
            NT AUTHORITY\SYSTEM:(OI)(CI)(F)
            CREATOR OWNER:(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files
PS B:\BuildSmart_BIM\Geschaeftsdaten>
```

Figure 12: Viewing the NTFS permissions of a directory

3.2.11 Sharing the Directories via SMB

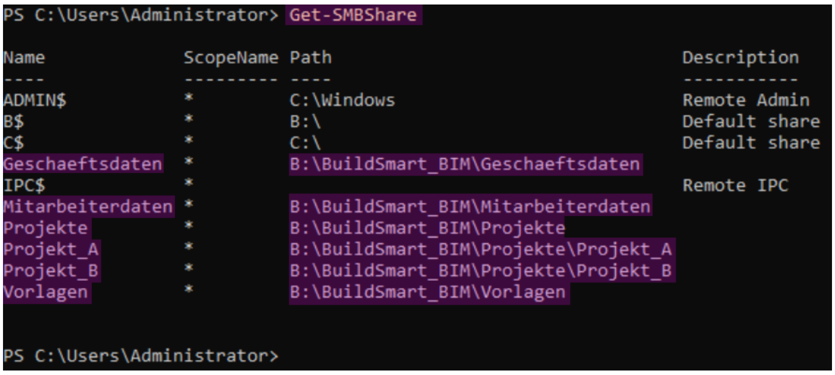
To create a new SMB share, an array of parameters must first be defined. These parameters are then used with the `New-SmbShare` command to create the SMB share.
The required parameters are straightforward and consistent across all shares, as we only need the following:

```
$Projekte = @{
    Name = "Projekte"
    Path = "$root\Projekte"
    FullAccess = "Administrators","Authenticated Users"
}
```

The share is then created using the following command:

```
New-SmbShare @Projekte
```

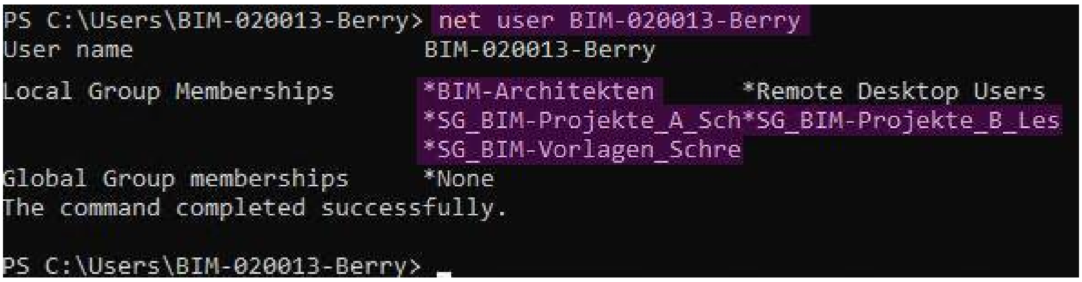
This process is repeated for the following directories, each with its own parameters: `Projekt_A`, `Projekt_B`, `Mitarbeiterdaten`, `Geschaeftsdaten`, and `Vorlagen`.[23]
The reason the share permissions are set to `Full Control` is that Windows follows the principle of least privilege. This means that more restrictive NTFS permissions will override the broader share-level permissions, eliminating the need to set permissions twice. The creation of the shares can be verified using the `Get-SMBShare` command, as shown in Figure 13.[24]



```
PS C:\Users\Administrator> Get-SMBShare

Name             ScopeName Path                                    Description
----             --------- ----                                    -----------
ADMIN$           *         C:\Windows                              Remote Admin
B$               *         B:\                                     Default share
C$               *         C:\                                     Default share
Geschaeftsdaten  *         B:\BuildSmart_BIM\Geschaeftsdaten
IPC$             *                                                 Remote IPC
Mitarbeiterdaten *         B:\BuildSmart_BIM\Mitarbeiterdaten
Projekte         *         B:\BuildSmart_BIM\Projekte
Projekt_A        *         B:\BuildSmart_BIM\Projekte\Projekt_A
Projekt_B        *         B:\BuildSmart_BIM\Projekte\Projekt_B
Vorlagen         *         B:\BuildSmart_BIM\Vorlagen


PS C:\Users\Administrator>
```

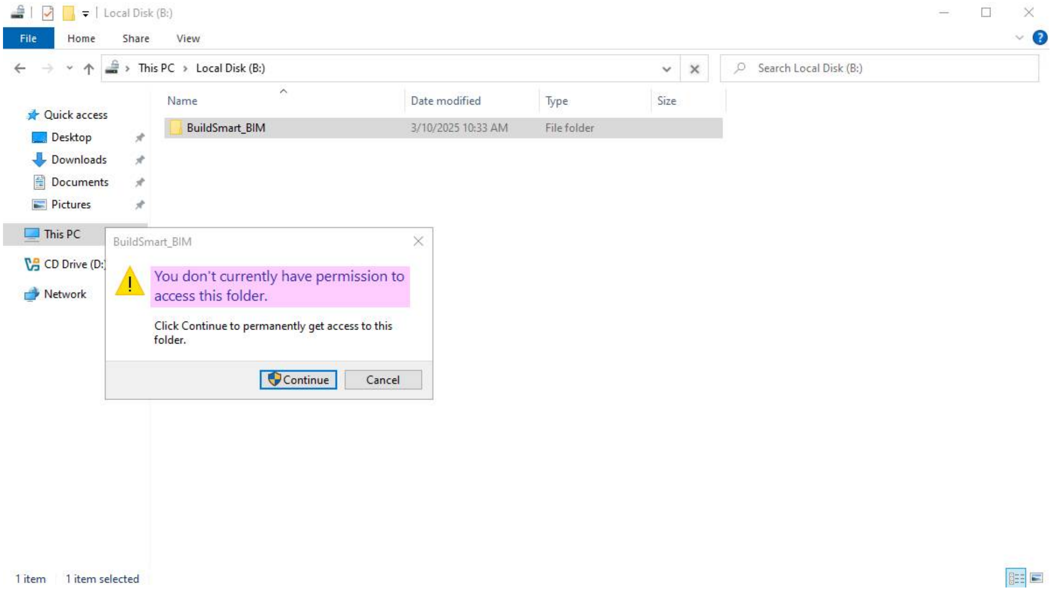
Figure 13: Listing all the active SMB shares

Now, users can only access the files they need via the network share. Since in Section 3.2.10, permissions to the root directory were removed, users do not have access to the entire file structure. In this scenario, this is a central server where each employee either has a local workstation or a thin client and connects to the server remotely.
For example HR employees access the server using Remote Desktop, while engineers have dedicated workstations but store and share their files through the network share. Because of this setup, Remote Desktop is enabled for all employees, and local access to storage is both restricted and unnecessary.
Figures 14 to 18 verify the correct functioning of the permissions.



```
PS C:\Users\BIM-020013-Berry> net user BIM-020013-Berry
User name                    BIM-020013-Berry

Local Group Memberships      *BIM-Architekten      *Remote Desktop Users
                             *SG_BIM-Projekte_A_Sch*SG_BIM-Projekte_B_Les
                             *SG_BIM-Vorlagen_Schre
Global Group memberships     *None
The command completed successfully.

PS C:\Users\BIM-020013-Berry> _
```

Figure 14: Showing the groups that the test user is part of

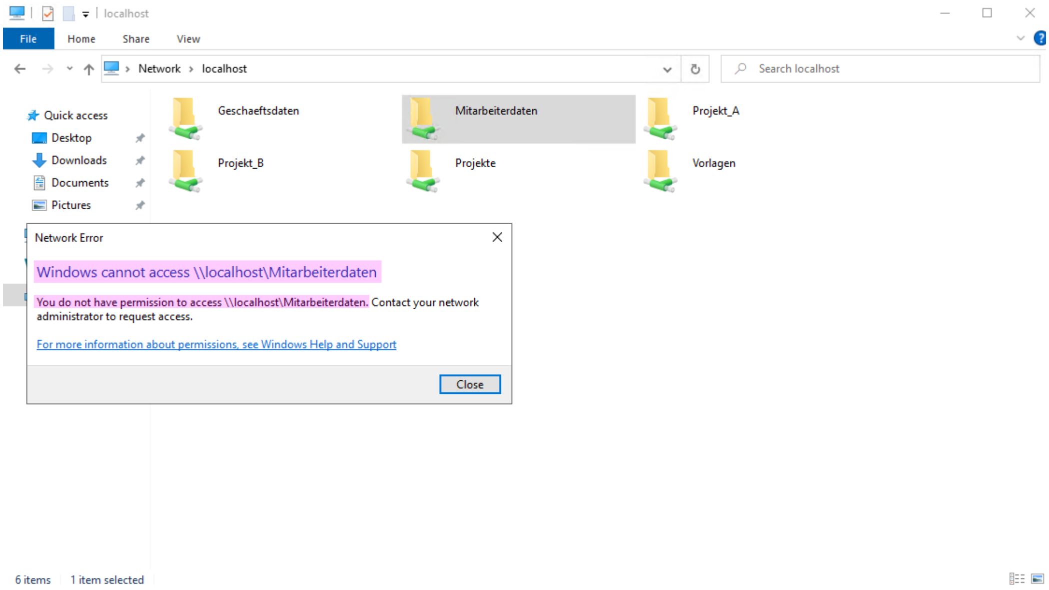


BuildSmart_BIM

You don't currently have permission to access this folder.

Click Continue to permanently get access to this folder.

Figure 15: Trying to access the directories locally on the drive



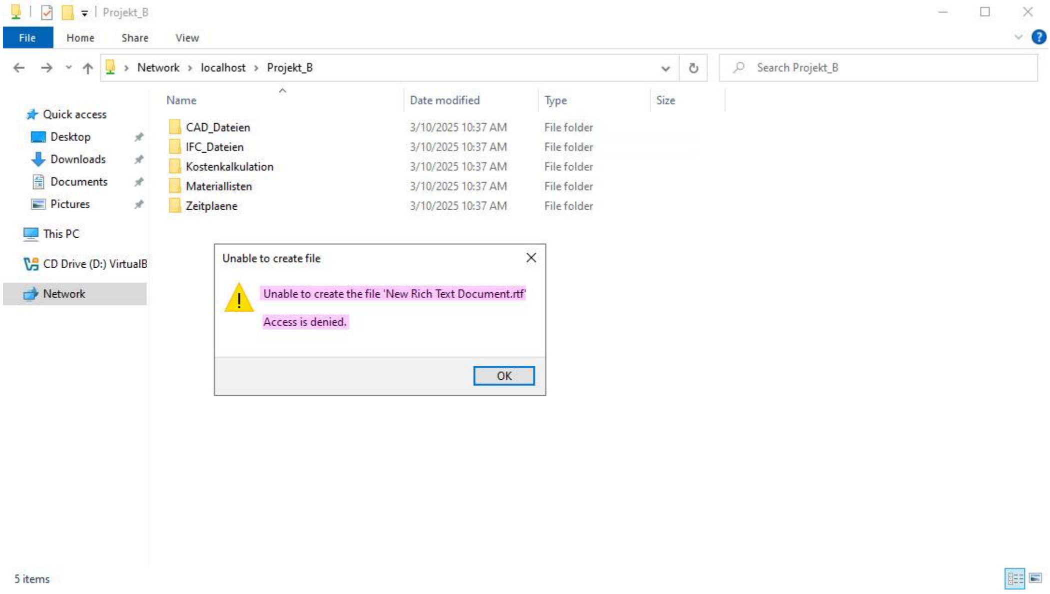
Network Error

Windows cannot access \\localhost\Mitarbeiterdaten

You do not have permission to access \\localhost\Mitarbeiterdaten. Contact your network administrator to request access.

For more information about permissions, see Windows Help and Support

Figure 16: Trying to access a share that the user has no permission to open



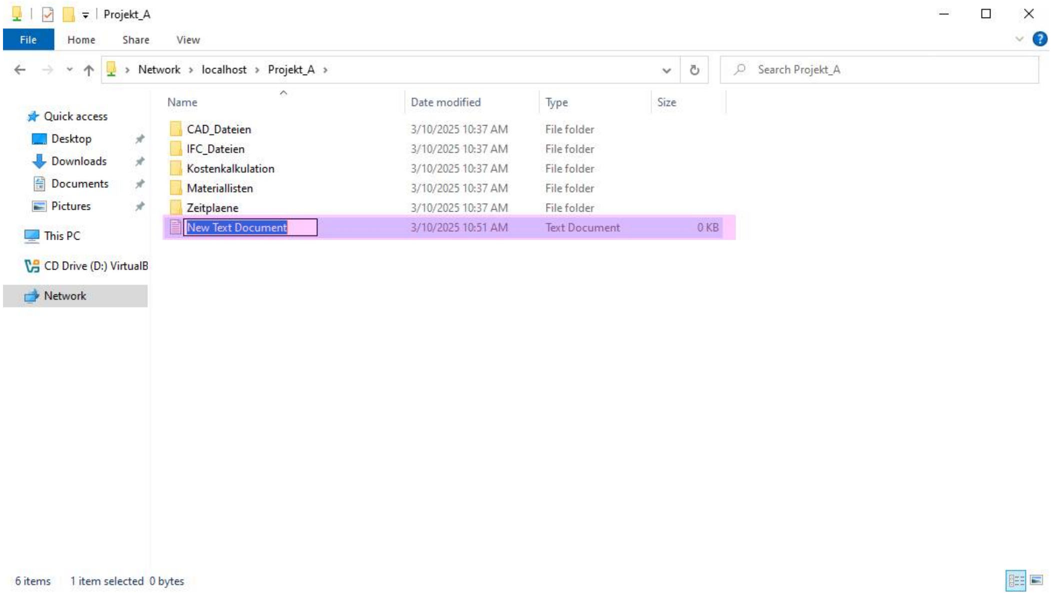
Name	Date modified	Type	Size
CAD_Dateien	3/10/2025 10:37 AM	File folder	
IFC_Dateien	3/10/2025 10:37 AM	File folder	
Kostenkalkulation	3/10/2025 10:37 AM	File folder	
Materiallisten	3/10/2025 10:37 AM	File folder	
Zeitplaene	3/10/2025 10:37 AM	File folder	

Unable to create file

Unable to create the file 'New Rich Text Document.rtf'

Access is denied.

Figure 17: Trying to create a file in a directory where the user only has read permissions



Name	Date modified	Type	Size
CAD_Dateien	3/10/2025 10:37 AM	File folder	
IFC_Dateien	3/10/2025 10:37 AM	File folder	
Kostenkalkulation	3/10/2025 10:37 AM	File folder	
Materiallisten	3/10/2025 10:37 AM	File folder	
Zeitplaene	3/10/2025 10:37 AM	File folder	
New Text Document	3/10/2025 10:51 AM	Text Document	0 KB

6 items 1 item selected 0 bytes

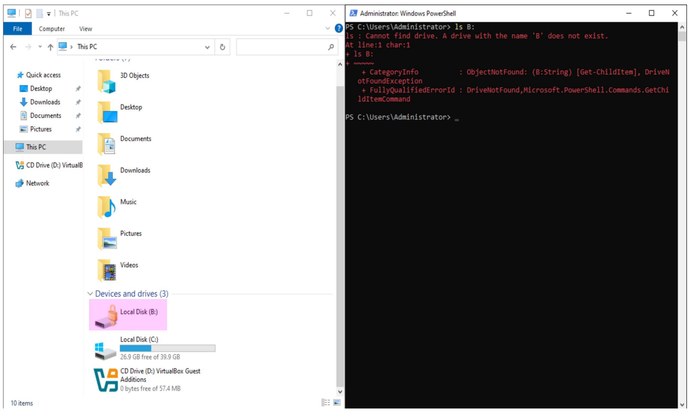
Figure 18: Creating a file in a directory where the user has read and write permissions

3.2.12 Encrypting the Volume using BitLocker

To encrypt the volume storing company data using BitLocker, the `Enable-BitLocker` command is used. The `-MountPoint` parameter specifies the drive letter of the volume to be encrypted, which in this case is ”B:”. The `-EncryptionMethod` parameter sets the encryption method, which in our case is `AES` with a `128-bit key`. The `-PasswordProtector` parameter specifies that a password will be used to protect the BitLocker-encrypted drive, with the `-Password` parameter setting the password to the variable that was created at the start of the script. [25]

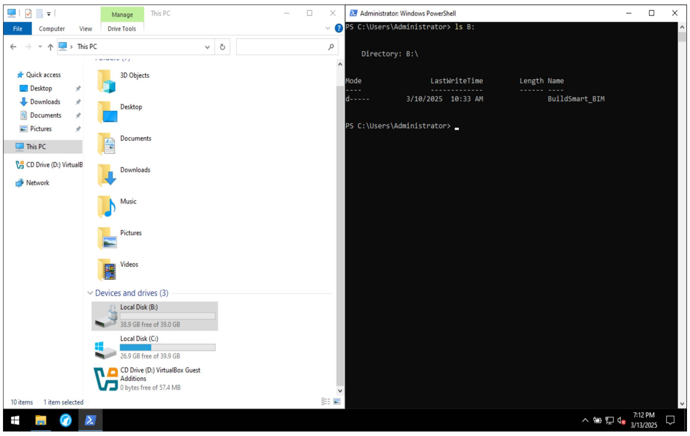
```
Enable-BitLocker -MountPoint "B:" -EncryptionMethod Aes128
                 -PasswordProtector -Password $supersurepassword
```

Now, every time the server is booted, an administrator must unlock the drive before it can be used. This issue could be addressed using Network Unlock or other options, but since the server is intended to run with 24/7 uptime, an administrator can simply unlock it after each reboot, as shown in Figures 19 and 20.



```
PS C:\Users\Administrator> ls B:
ls : Cannot find drive. A drive with the name 'B' does not exist.
At line:1 char:1
+ ls B:
+ ~~~~~
    + CategoryInfo          : ObjectNotFound: (B:String) [Get-ChildItem], DriveN
   otFoundException
    + FullyQualifiedErrorId : DriveNotFound,Microsoft.PowerShell.Commands.GetChi
   ldItemCommand

PS C:\Users\Administrator> _
```

Figure 19: Trying to list the contents of the encrypted volume



```
PS C:\Users\Administrator> ls B:


    Directory: B:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/10/2025  10:33 AM                BuildSmart_BIM


PS C:\Users\Administrator>
```

Figure 20: Trying to list the contents of the encrypted volume after decrypting it

4 References

References

[1] “Localized Names of Users and Groups in Windows | Nicos Blog,” May 2018, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://blog.nicoboehr.de/2014/08/20/localized-names-of-users-and-groups-in-windows

[2] sdwheeler, “Invoke-WebRequest (Microsoft.PowerShell.Utility) - PowerShell,” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-7.5

[3] Z. Agoston, “PowerShell Abbreviation Table | OpenTechTips,” *OpenTechTips*, Sep. 2020. [Online]. Available: https://opentechtips.com/powershell-abbreviation-table

[4] sdwheeler, “Invoke-Expression (Microsoft.PowerShell.Utility) - PowerShell,” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-expression?view=powershell-7.5

[5] ——, “Set-ExecutionPolicy (Microsoft.PowerShell.Security) - PowerShell,” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/set-executionpolicy?view=powershell-7.5

[6] JasonGerend, “Install-WindowsFeature (ServerManager),” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/servermanager/install-windowsfeature?view=windowsserver2025-ps

[7] ——, “BitLocker Module,” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/bitlocker/?view=windowsserver2025-ps

[8] sdwheeler, “Rename-Computer (Microsoft.PowerShell.Management) - PowerShell ,” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/rename-computer?view=powershell-7.5

[9] “How to Enable Remote Desktop Using PowerShell?” Dec. 2024, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://powershellfaqs.com/enable-remote-desktop-using-powershell

[10] JasonGerend, “New-ScheduledTask (ScheduledTasks),” Feb. 2025, [Online; accessed 11. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/scheduledtasks/new-scheduledtask?view=windowsserver2025-ps

[11] sdwheeler, “ConvertTo-SecureString (Microsoft.PowerShell.Security) - PowerShell,” Feb. 2025, [Online; accessed 12. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/convertto-securestring?view=powershell-7.5

[12] ——, “New-LocalUser (Microsoft.PowerShell.LocalAccounts) - PowerShell,” Feb. 2025, [Online; accessed 12. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/new-localuser?view=powershell-5.1

[13] ——, “Add-LocalGroupMember (Microsoft.PowerShell.LocalAccounts) - PowerShell,” Feb. 2025, [Online; accessed 12. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/add-localgroupmember?view=powershell-5.1

[14] JasonGerend, “Resize-Partition (Storage),” Feb. 2025, [Online; accessed 12. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/storage/resize-partition?view=windowsserver2025-ps

[15] Laxmansingh@twc, “What is difference between Partition, Volume and Logical Drive ,” *Windows Club*, Apr. 2022. [Online]. Available: https://www.thewindowsclub.com/difference-between-partition-volume-logical-drive

[16] JasonGerend, “New-Partition (Storage),” Feb. 2025, [Online; accessed 12. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/storage/new-partition?view=windowsserver2025-ps

[17] G. Watumull, “Cluster size recommendations for ReFS and NTFS,” *TECHCOMMUNITY.MICROSOFT*, Oct. 2019. [Online]. Available: https://techcommunity.microsoft.com/blog/filecab/cluster-size-recommendations-for-refs-and-ntfs/425960

[18] JasonGerend, “Format-Volume (Storage),” Feb. 2025, [Online; accessed 12. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/storage/format-volume?view=windowsserver2025-ps

[19] Amanda, “What Is Allocation Unit Size & How to Change It - MiniTool Partition Wizard,” *MiniTool*, Jul. 2023. [Online]. Available: https://www.partitionwizard.com/partitionmanager/file-allocation-unit-size.html

[20] robinharwood, “icacls,” Nov. 2024, [Online; accessed 13. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/icacls

[21] “iCACLS: List and Manage Folder and File Permissions on Windows – TheITBros,” Mar. 2025, [Online; accessed 13. Mar. 2025]. [Online]. Available: https://theitbros.com/using-icacls-to-list-folder-permissions-and-manage-files

[22] “NTFS Permissions: A Comprehensive Guide - Petri IT Knowledgebase,” Feb. 2024, [Online; accessed 13. Mar. 2025]. [Online]. Available: https://petri.com/ntfs-permissions

[23] JasonGerend, “New-SmbShare (SmbShare),” Feb. 2025, [Online; accessed 13. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/smbshare/new-smbshare?view=windowsserver2025-ps

[24] “NTFS vs. Share Permissions: Which to Use When and Why,” Mar. 2025, [Online; accessed 13. Mar. 2025]. [Online]. Available: https://albusbit.com/blog/ntfs-vs-share-permissions

[25] JasonGerend, “Enable-BitLocker (BitLocker),” Feb. 2025, [Online; accessed 13. Mar. 2025]. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/bitlocker/enable-bitlocker?view=windowsserver2025-ps

5 List of figures

List of Figures

6 Attachments

Setup script:
`https://github.com/Stefanistkuhl/obsidianschule/blob/main/3.Klasse/itsi/aufgaben/windoof/setup.ps1`
Main script:
`https://github.com/Stefanistkuhl/obsidianschule/blob/main/3.Klasse/itsi/aufgaben/windoof/script.ps1`