****

**­**

**Министр науки и высшего образования Российской̆**

**Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

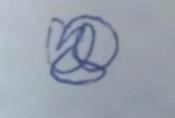
Рабочий протокол и отчет по лабораторной работе №2

Основы работы с Active Directory в Windows Server.

**Выполнил(а) студент группы** № **M33091**

Сидорцов Владимир Сергеевич

**Подпись:**



**Проверил:**

Санкт-Петербург

2022г.

1. **Цель работы.**

Получить базовые навыки развертывания службы каталогов Active Directory на основе Windows Server, управления объектами AD, их правами и групповыми политиками.

1. **Задачи, решаемые при выполнении работы.**

* Создание и настройка локальной сети.
* Работа с оснастками Active Directory.
* Настройка ролей контроллера домена.
* Работа с правами NTFS и SMB.
* Работа с групповыми политиками.
* Создание и управление объектами Active Directory.

1. **Объект исследования.**

* Windows Server 2012 r2
* Active Directory
* Powershell

1. **Исходные данные.**

● CSV file with information about AD users:

First\_name;Last\_name;Patronymic;Post;Department;Email;Phone;Login;Password;Ou\_name;Group\_names;Path\_to\_home\_folder;Path\_to\_profile

Alexei;Tsydenov;Gomboevich;Student;M33091;;89500852022;Leha;135TAGLorr!;unit-for-delete;"gM33091,gManagers";;

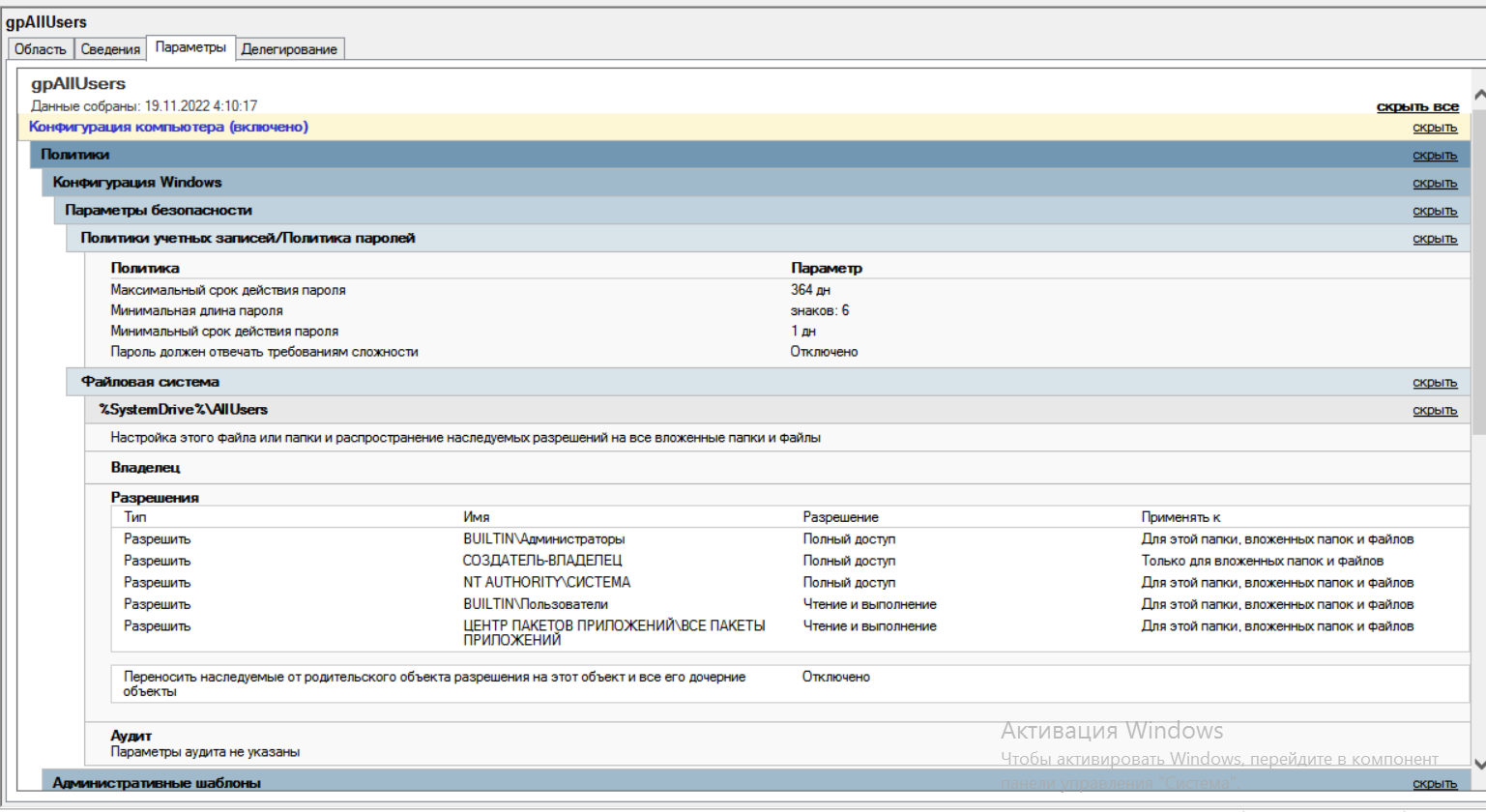
Renat;Mirzabekov;Eldarovich;Student;M33091;;;Reno;135MRERayn!;unit-for-delete;"gM33091";;

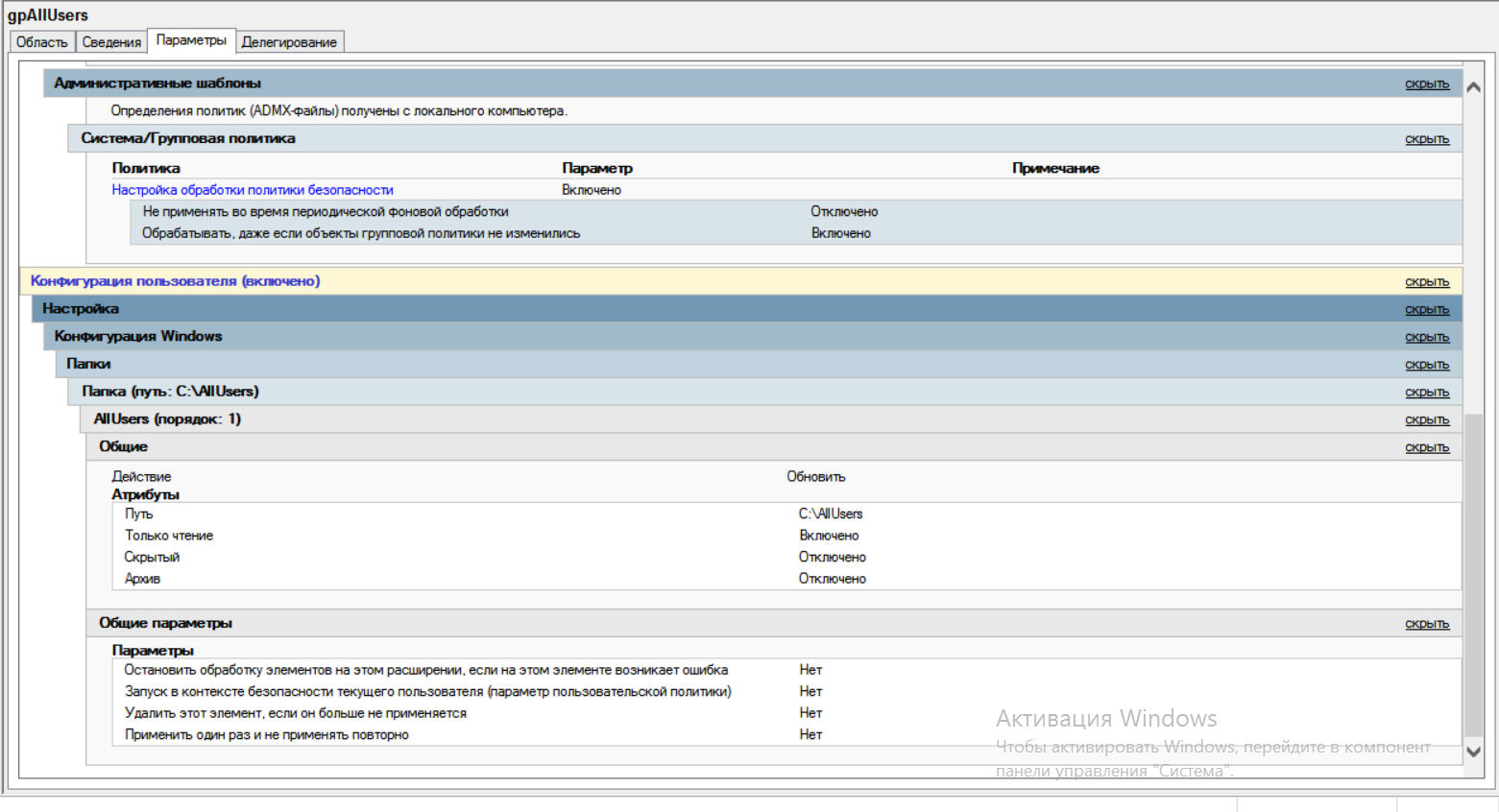
Lev;Maksimov;Sergeevich;Student;M33091;;;Leva;135MLSyhp!;unit-for-delete;"gM33091";;

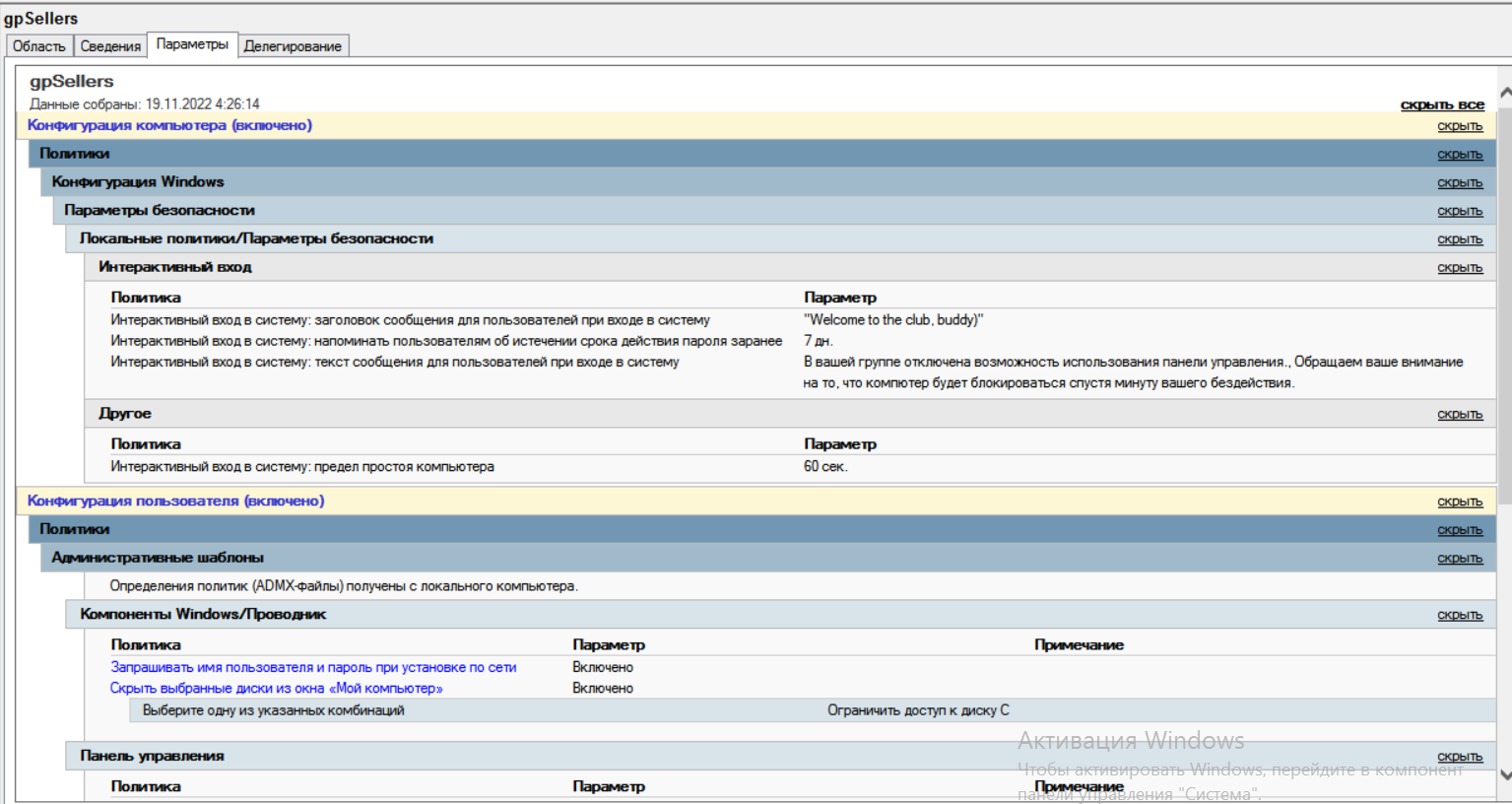
Nikita;Fomkin;S;Student;M33091;;;Nekit;135FNKachok!;unit-for-delete;"gM33091";;

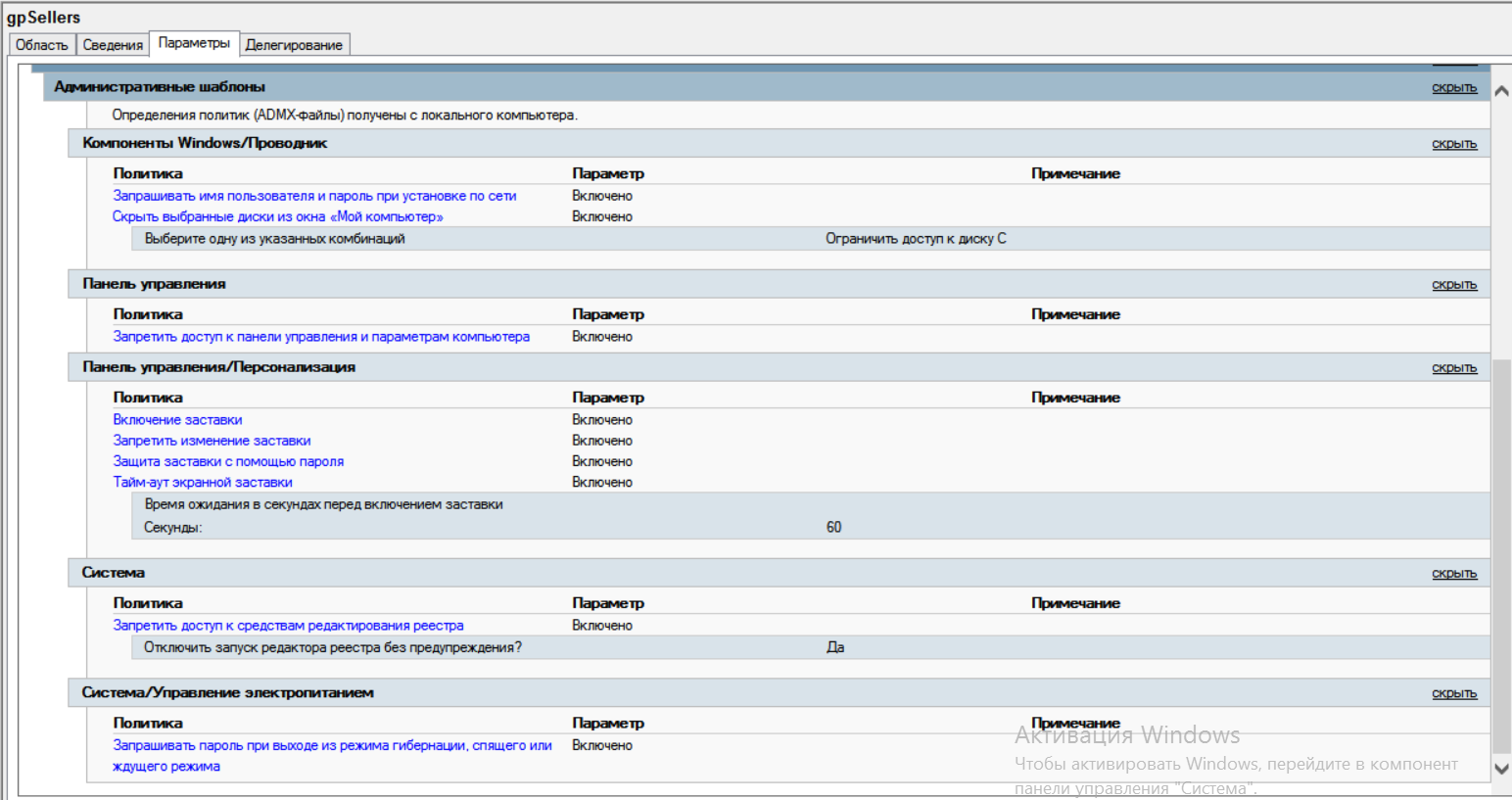
1. **Окончательные результаты.**

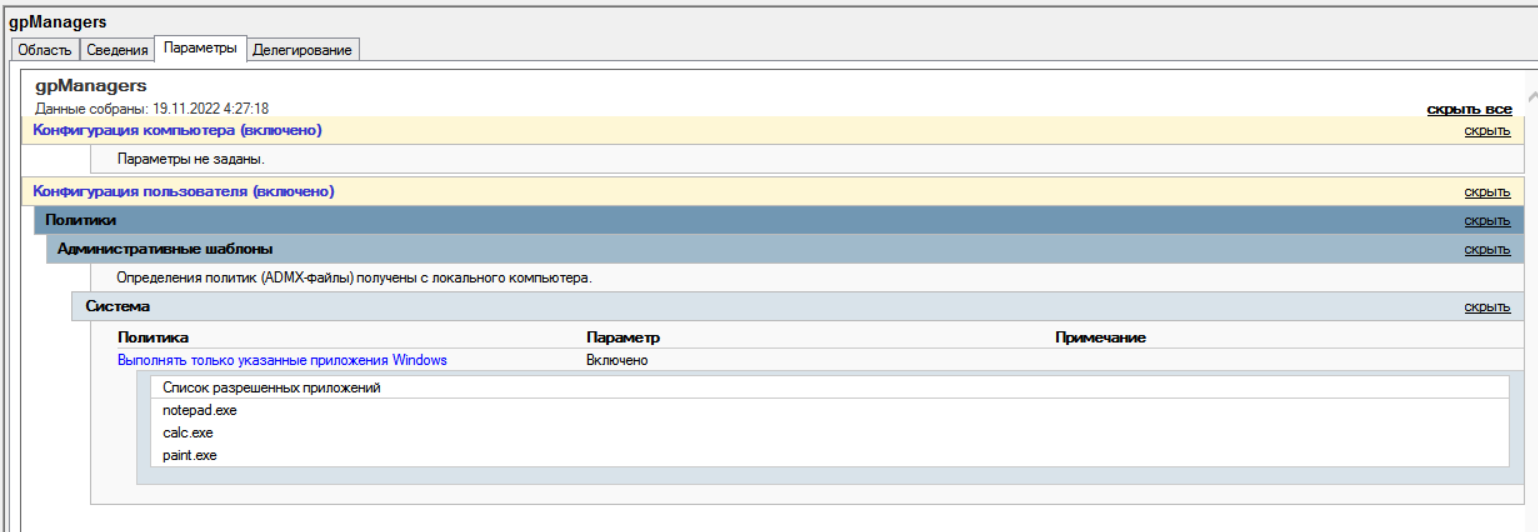
Скриншоты групповых политик из пункта 4:

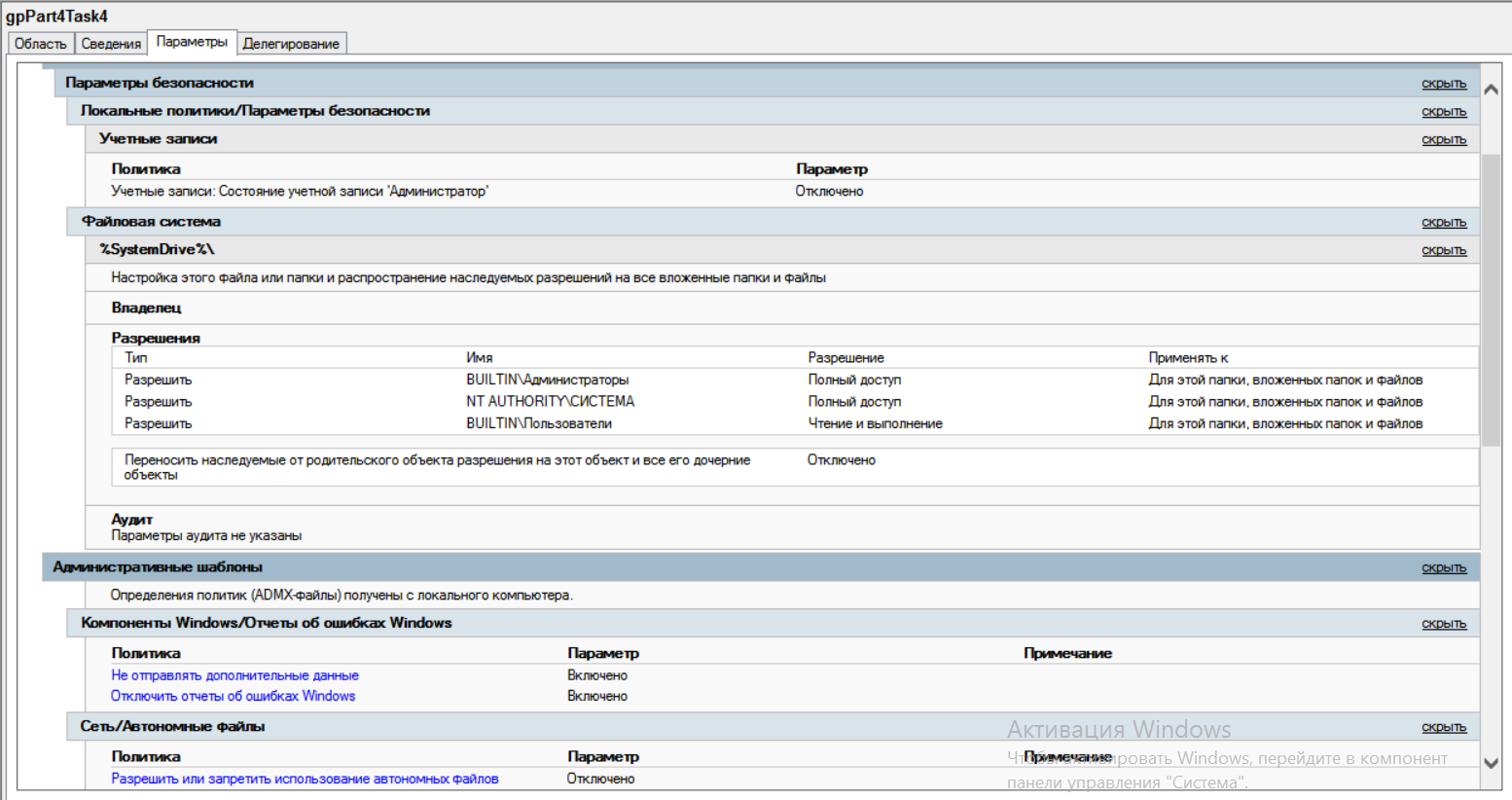


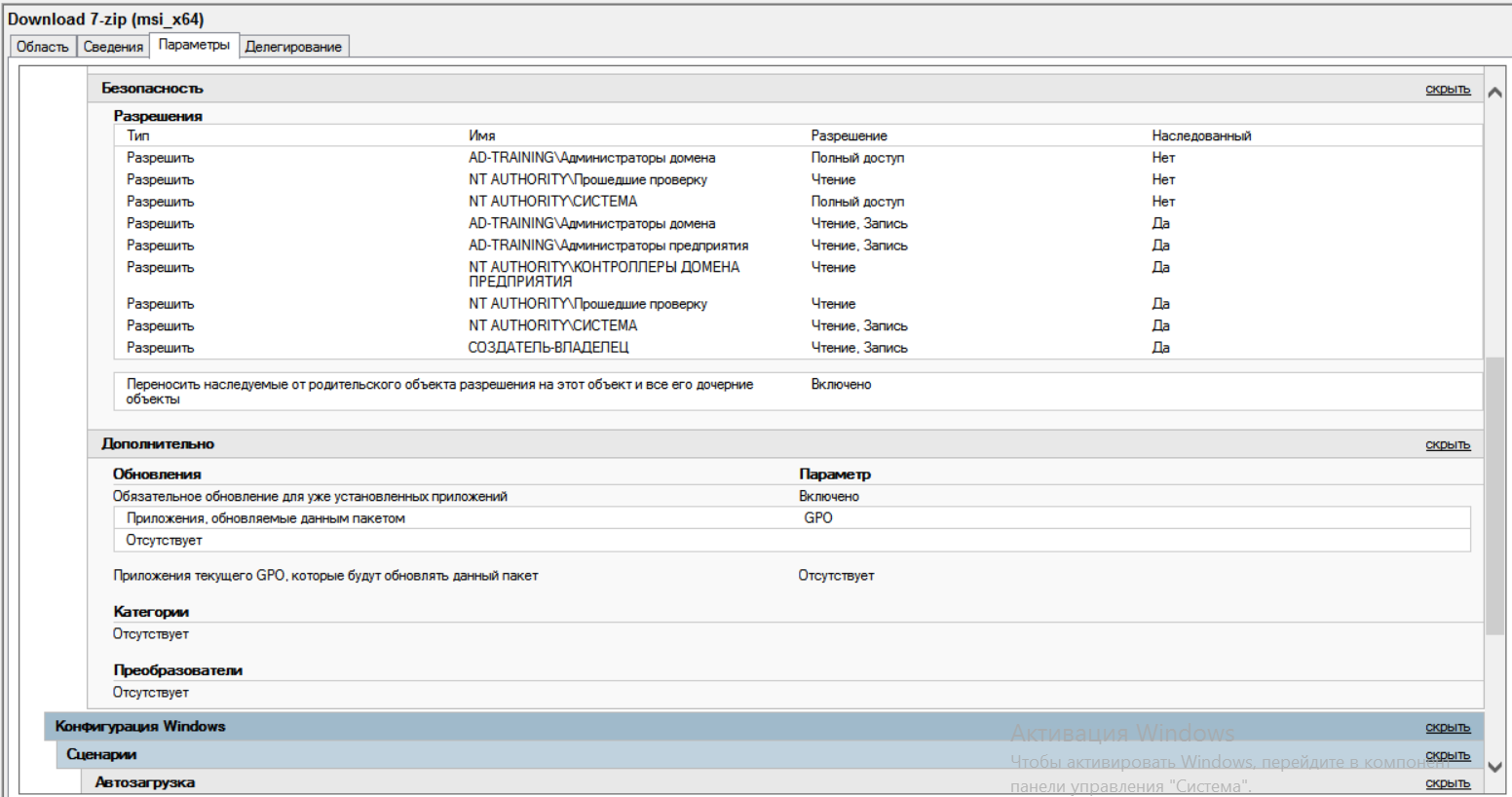
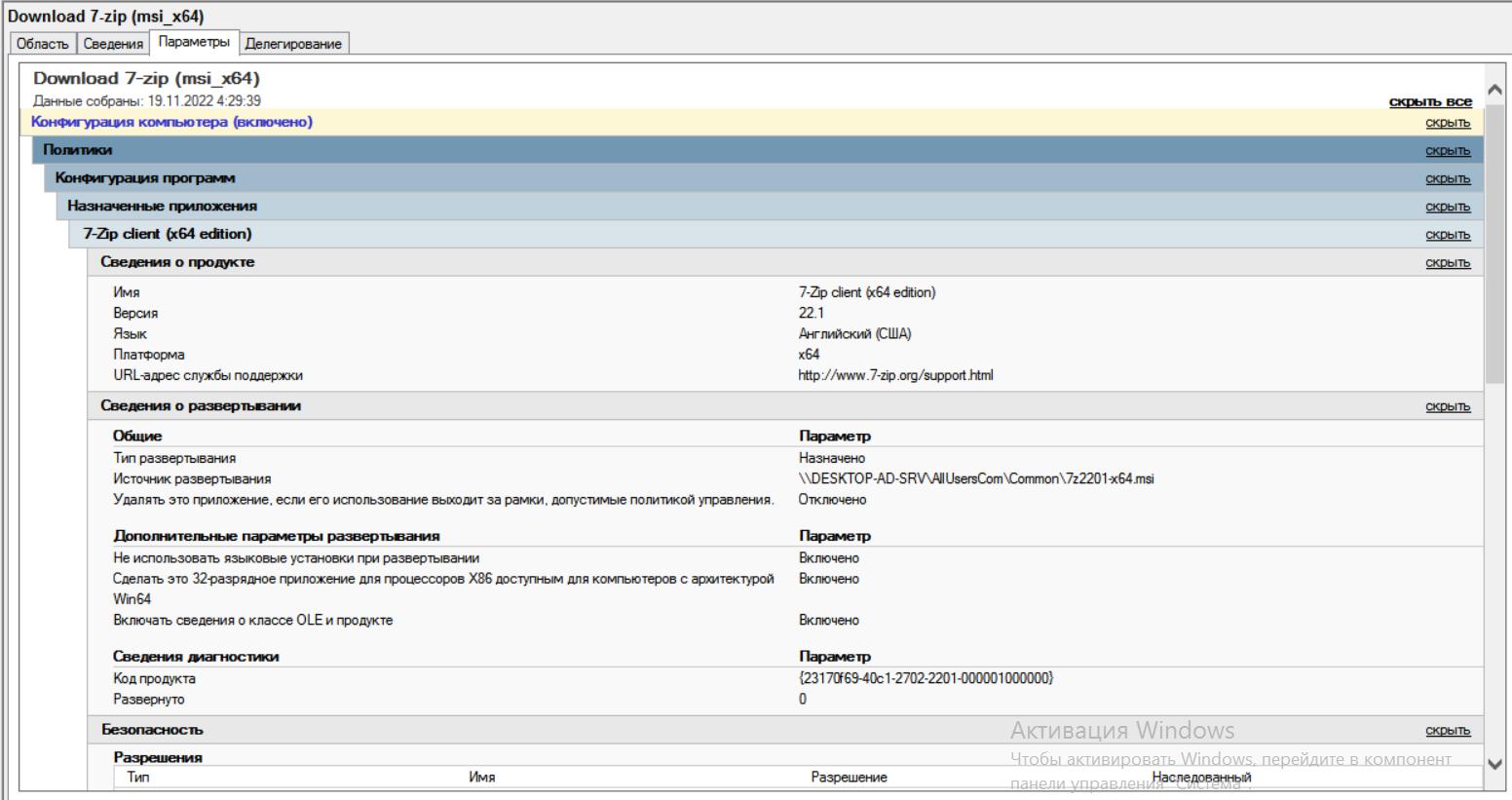


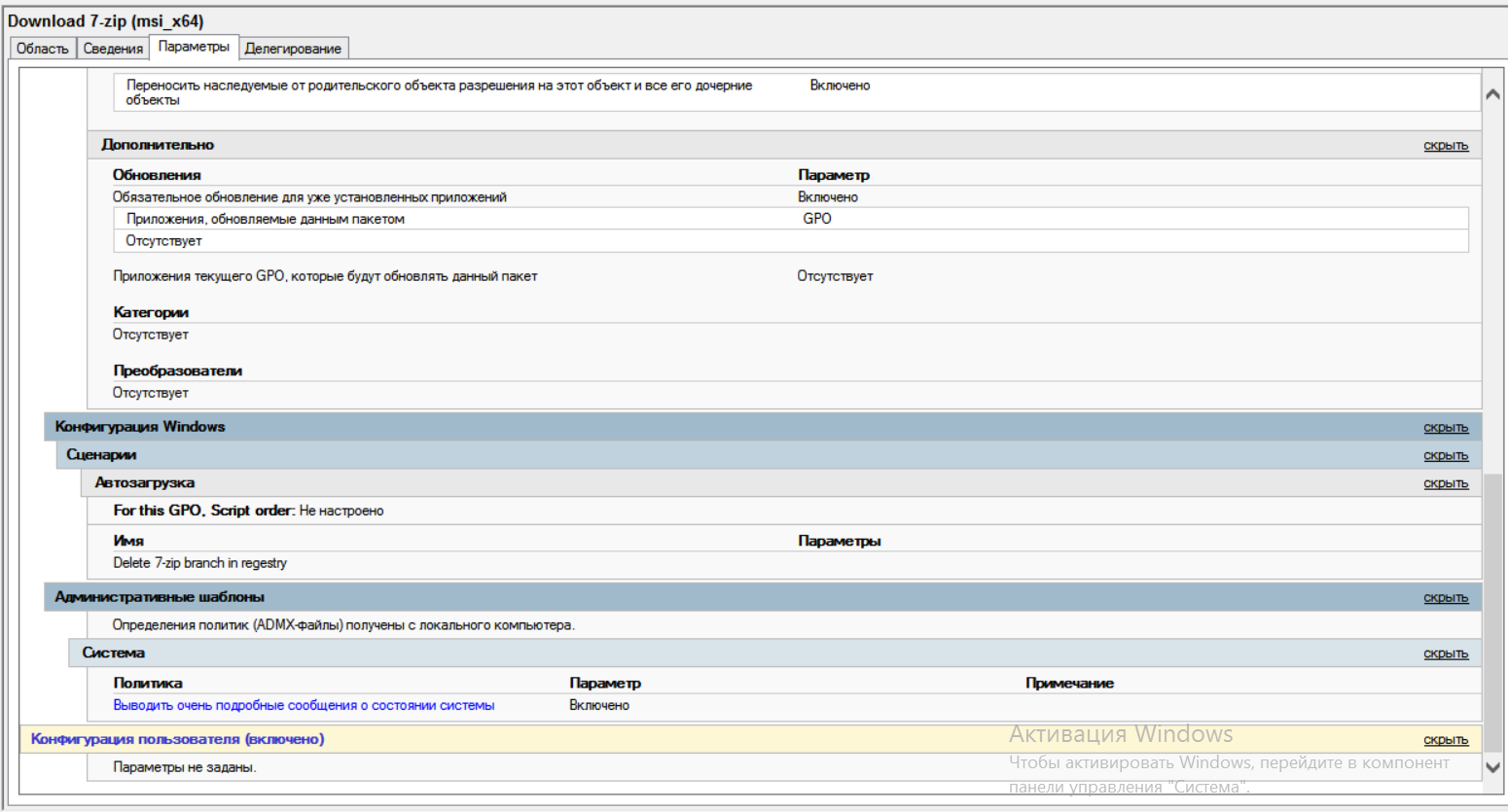












Скрипт из 5 части лабораторной:

param (

[string]$Path2CsvFile,

[string]$Delimiter

)

function PrintOperationBegin {

param (

[string]$string

)

Write-Host $string -BackgroundColor DarkYellow -ForegroundColor Black

}

function PrintOperationSuccess {

param (

[string]$string

)

Write-Host $string -BackgroundColor DarkGreen -ForegroundColor Black

}

function PrintInfo {

param (

[string]$string

)

Write-Host $string -BackgroundColor Cyan -ForegroundColor White

}

function PrintCheck {

param (

[string]$string

)

Write-Host $string -BackgroundColor DarkMagenta -ForegroundColor Black

}

function PrintAlert {

param (

[string]$string

)

Write-Host $string -BackgroundColor Red -ForegroundColor Black

}

function PrintNewLine {

Write-Host ""

}

$CreatedOuCounter = 0

$CreatedGroupsCounter = 0

$CreatedUserCounter = 0

$Log = ""

if (!(Test-Path $Path2CsvFile -PathType Leaf)) {

PrintAlert "Can't open the file: file is not exist!"

break

}

Import-Module ActiveDirectory

$Users = Import-CSV -Path $Path2CsvFile -Delimiter $Delimiter -Encoding UTF8

$FQDN = [System.Net.Dns]::GetHostEntry((hostname)).HostName

# [System.Net.Dns]::GetHostEntry($env:COMPUTERNAME).HostName

$ComputerName, $Domain, $CommonDomainComponent = $FQDN -split '.', 0, "simplematch"

$DomainName = $Domain + "." + $CommonDomainComponent

# Write-Host $DomainName # ad-training.loc

# Write-Host fqdn = $FQDN # DESKTOP-AD-SRV.ad-training.loc

foreach ($User in $Users) {

$First\_name = $User.First\_name

$Last\_name = $User.Last\_name

$Patronymic = $User.Patronymic

$Post = $User.Post

$Department = $User.Department

$Email = $User.Email

$Phone = $User.Phone

$Login = $User.Login

$SoursePassword = [string]$User.Password

$Ou\_name = $User.Ou\_name

$Group\_names = $User.Group\_names

$Path\_to\_home\_folder = $User.Path\_to\_home\_folder

$Path\_to\_profile = $User.Path\_to\_profile

$DisplayName = $First\_name + " " + $Last\_name

$Username = $Last\_name + $First\_name[0] + $Patronymic[0]

$UserPrincipalName = $Login + "@" + $DomainName

$Password = ConvertTo-SecureString $SoursePassword -AsPlainText -Force

$DistinguishedPath2Ou = "$($(Get-ADDomain).DistinguishedName)"

$DistinguishedPath2OuSubs = "OU=$($Ou\_name),$($DistinguishedPath2Ou)"

$DistinguishedPath2OuUser = "CN=$($Username),$($DistinguishedPath2OuSubs)"

$UserObj = $Null

try {

$UserObj = Get-ADUser -Identity $DistinguishedPath2OuUser

}

catch {

PrintNewLine

PrintCheck "User ($($DisplayName)) does not exist. This user will be created."

}

if (!($Null -eq $UserObj)) {

PrintInfo "Sorry, but user $($Last\_name) $($First\_name) $($Patronymic) already exist."

continue

}

# Write-Host typeof((Get-ADOrganizationalUnit -Identity "$($DistinguishedPath2Ou)"))

# # (![adsi]::Exists("LDAP://OU=$($Ou\_name),DC=$($Domain),DC=$($CommonDomainComponent)"))

# if (!(Get-ADOrganizationalUnit -Identity "$($DistinguishedPath2Ou)")) {

# $newOu = New-ADOrganizationalUnit -Name $Ou\_name -Path $DistinguishedPath2Ou -PassThru

# }

try {

PrintNewLine

PrintOperationBegin "Creating organization unit..."

New-ADOrganizationalUnit -Name $Ou\_name -Path $DistinguishedPath2Ou

PrintOperationSuccess "Organization unit created."

$CreatedOuCounter += 1

$Log += "Organization unit: $($Ou\_name) created.`n"

}

catch {

PrintNewLine

PrintInfo "Specified organization unit already exist, working..."

}

$Path2MainFolder = "C:\UsersHome\$($Username)"

PrintNewLine

PrintOperationBegin "Creating main folder..."

New-Item -Path $Path2MainFolder -ItemType Directory -Force -ea Stop

try {

PrintOperationBegin "Sharing the main folder..."

New-SmbShare -Name "$($Username)$" -Path $Path2MainFolder -ea Stop

PrintOperationSuccess "Main folder was created and shared."

}

catch {

PrintInfo "Main folder is alredy shared, working..."

}

$HomeFolderMask = "X:"

$Path2HomeFolder = $Path\_to\_home\_folder

$DefaultPath2HomeFolder = "C:\UsersHome\$Username\HomeFolder"

if (($Null -eq $Path2HomeFolder) -or ($Path2HomeFolder -eq "")) {

$Path2HomeFolder = $DefaultPath2HomeFolder

}

elseif (!(Test-Path -Path $Path\_to\_home\_folder)) {

$Path2HomeFolder = $DefaultPath2HomeFolder

}

elseif (!(Test-Path -IsValid -Path $Path\_to\_home\_folder)) {

$Path2HomeFolder = $DefaultPath2HomeFolder

}

# New-SmbMapping -LocalPath "C:\UsersHome\$($Username)\HomeFolder" -RemotePath $Path2HomeFolder -UserName $Username -Password $SoursePassword -Persistent $True -SaveCredentials -HomeFolder -ea Stop

PrintNewLine

PrintOperationBegin "Creating user home folder..."

New-Item -Path $Path2HomeFolder -ItemType Directory -Force -ea Stop

PrintOperationSuccess "User home folder was created."

# try {

# Write-Host "Sharing user home folder..."

# New-SmbShare -Path $Path2HomeFolder -ea Stop

# }

# catch {

# Write-Host "User home folder is alredy shared"

# }

$Path2Profile = $Path\_to\_profile

$DefaultPath2Profile = "C:\UsersHome\$Username\Profile"

if (($Null -eq $Path2Profile) -or ($Path2Profile -eq "")) {

$Path2Profile = $DefaultPath2Profile

}

elseif (!(Test-Path -Path $Path2Profile)) {

$Path2Profile = $DefaultPath2Profile

}

elseif (!(Test-Path -IsValid -Path $Path2Profile)) {

$Path2Profile = $DefaultPath2Profile

}

PrintNewLine

PrintOperationBegin "Creating user profile folder..."

New-Item -Path $Path2Profile -ItemType Directory -Force -ea Stop

PrintOperationSuccess "User profile folder created."

PrintNewLine

PrintNewLine

PrintOperationBegin "Creating user..."

$\_Name = $Username

$\_GivenName = "$($First\_name) $($Patronymic)"

$\_Surname = $Last\_name

$\_DisplayName = $DisplayName

$\_UserPrincipalName = $UserPrincipalName

$\_SamAccountName = $Username

$\_Title = $Post

$\_Department = $Department

$\_EmailAddress = $Email

$\_MobilePhone = $Phone

$\_AccountPassword = $Password

$\_PasswordNeverExpires = $true

$\_Path = $DistinguishedPath2OuSubs #"OU=$($Ou\_name),DC=$($Domain),DC=$($CommonDomainComponent)"

$\_Enabled = $true

$\_HomeDrive = $HomeFolderMask

$\_HomeDirectory = $Path2HomeFolder

$\_ProfilePath = $Path2Profile

New-ADUser -Name $\_Name -GivenName $\_GivenName -Surname $\_Surname -DisplayName $\_DisplayName -UserPrincipalName $\_UserPrincipalName -SamAccountName $\_SamAccountName -Title $\_Title -Department $\_Department -EmailAddress $\_EmailAddress -MobilePhone $\_MobilePhone -AccountPassword $\_AccountPassword -PasswordNeverExpires $\_PasswordNeverExpires -Path $\_Path -Enabled $\_Enabled -HomeDirectory $\_HomeDirectory -HomeDrive $\_HomeDrive -ProfilePath $\_ProfilePath

Get-ADUser -Identity $DistinguishedPath2OuUser

PrintOperationSuccess "User $($DisplayName) was created."

$CreatedUserCounter += 1

$Log += "User: $($DisplayName) created.`n"

$Group\_names += ",gAllUsers"

foreach ($groupname in ($Group\_names -split ',', 0, "simplematch")) {

$Path2Group = $Null

PrintNewLine

PrintNewLine

if(!(Get-ADGroup -Filter { Name -eq $groupname })) {

PrintCheck "Group $($groupname) does not exist."

PrintOperationBegin "Creating group..."

$Path2Group = $DistinguishedPath2OuSubs

New-ADGroup -Name $groupname -GroupCategory Security -GroupScope Global -Path $Path2Group

PrintOperationSuccess "Group was created."

$CreatedGroupsCounter += 1

$Log += "Group: $($groupname) created.`n"

} else

{

PrintInfo "Group $($groupname) is alredy exist."

}

PrintNewLine

PrintOperationBegin "Adding $($DisplayName) to $($groupname) group..."

$members = Get-ADGroupMember -Identity $groupname -Recursive | Select-Object -ExpandProperty UserPrincipalName

if ($members -notcontains $DistinguishedPath2OuUser) {

Add-ADGroupMember $groupname -Members $Username

PrintOperationSuccess "$($DisplayName) was added to $($groupname) group."

} else

{

PrintInfo "$($DisplayName) is already in this group."

}

}

PrintNewLine

PrintNewLine

$UserObj = Get-ADUser -Identity $DistinguishedPath2OuUser

PrintOperationBegin "Setting permissions to main folder..."

$AccessControlType = [System.Security.AccessControl.AccessControlType]::Allow

$FileSystemRights = [System.Security.AccessControl.FileSystemRights]::ReadAndExecute

$InheritanceFlags = [System.Security.AccessControl.InheritanceFlags]::ContainerInherit

$PropagationFlags = [System.Security.AccessControl.PropagationFlags]::InheritOnly

$AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule ($UserObj.SID, $FileSystemRights, $InheritanceFlags, $PropagationFlags, $AccessControlType)

$ACL = Get-Acl $Path2MainFolder

# Отключаем наследование правил:

$ACL.SetAccessRuleProtection($True, $True) # 1st True = является ли этот объект защищенным, 2nd True = нужно ли скопировать NTFS разрешения.

$ACL.AddAccessRule($AccessRule)

Set-Acl -Path $Path2MainFolder -AclObject $ACL -ea Stop

PrintOperationSuccess "Permissions to main folder are ready."

PrintNewLine

PrintOperationBegin "Setting permissions to user home folder..."

$AccessControlType = [System.Security.AccessControl.AccessControlType]::Allow

$FileSystemRights = [System.Security.AccessControl.FileSystemRights]::Modify

$InheritanceFlags = [System.Security.AccessControl.InheritanceFlags]"ContainerInherit, ObjectInherit"

$PropagationFlags = [System.Security.AccessControl.PropagationFlags]::InheritOnly

$AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule ($UserObj.SID, $FileSystemRights, $InheritanceFlags, $PropagationFlags, $AccessControlType)

$ACL = Get-Acl $Path2HomeFolder

$ACL.SetAccessRuleProtection($True, $True)

$ACL.AddAccessRule($AccessRule)

Set-Acl -Path $Path2HomeFolder -AclObject $ACL -ea Stop

PrintOperationSuccess "Permissions to to user home folder are ready."

PrintNewLine

PrintOperationBegin "Setting permissions to user profile folder..."

$AccessControlType = [System.Security.AccessControl.AccessControlType]::Allow

$FileSystemRights = [System.Security.AccessControl.FileSystemRights]::Modify

$InheritanceFlags = [System.Security.AccessControl.InheritanceFlags]"ContainerInherit, ObjectInherit"

$PropagationFlags = [System.Security.AccessControl.PropagationFlags]::InheritOnly

$AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule ($UserObj.SID, $FileSystemRights, $InheritanceFlags, $PropagationFlags, $AccessControlType)

$ACL = Get-Acl $Path2Profile

$ACL.SetAccessRuleProtection($True, $True)

$ACL.AddAccessRule($AccessRule)

Set-Acl -Path $Path2Profile -AclObject $ACL -ea Stop

PrintOperationSuccess "Permissions to user profile folder are ready."

}

$Html = "<html>`n" + $Log + "`nUsers created: $($CreatedUserCounter).`nGroups created: $($CreatedGroupsCounter).`nOrganization units created: $($CreatedOuCounter).`n" + "</html>"

Set-Content -Path "Creation\_log.html" -Value $Html

1. Как с помощью Powershell восстановить удаленный объект AD?

* Надо каким-либо способом выбрать нужный объект
* Восстанавливаем с помощью Restore-ADObject

1. Конвейер команд из ч.6 п.3:

* dsquery user "OU=unit-for-delete,DC=ad-training,DC=loc" | dsrm –noprompt

1. Конвейер команд из ч.6 п.4:

* Get-ADObject -SearchBase 'CN=Deleted Objects,DC=ad-training,DC=loc' -Filter {isDeleted -eq $True -and lastKnownParent -eq 'OU=unit-for-delete,DC=ad-training,DC=loc'} -IncludeDeletedObjects | Restore-ADObject

1. **Выводы и анализ результатов работы.**

Понял

1. **Дополнительные задания.**

Вопросы:

1. Раскройте смысл терминов дерево доменов, лес и схема Active Directory?

* Дерево доменов – это пространство имен, которое формируется за счет нескольких доменов, использующих общую схему и конфигурацию Лес - это набор ссылочных объектов, который не формирует пространство имен, но содержит в себе несколько деревьев домена (или одно дерево) Схема AD - набор определений для классов, атрибутов и других объектов, которые могут существовать в лесу AD

1. Где на контроллере домена хранится данные об объектах Active Directory в виде файлов? Какие файлы за что отвечают?

* База данных Active Directory хранится в файле по имени Ntds.dit, который по умолчанию расположен в папке %systemroot %\NTDS. Эта папка содержит также следующие файлы. Edb.chk- файл контрольных точек, который указывает, какие транзакции из журналов регистрации были записаны в базу данных Active Directory.
* Edb.log- журнал регистрации текущих транзакций. Имеет фиксированную длину - 10 Мб.
* Edbxxxxx.log. После того как Active Directory проработала некоторое время, могут появиться один или более журналов, у которых часть имени файла, обозначенная как ххххх, представляется собой увеличивающийся шестнадцатеричный порядковый номер. Эти журналы являются предшествующими журналами; всякий раз, когда текущий журнал заполнен, он переименовывается в следующий предшествующий журнал, и создается новый журнал Edb.log. Старые журналы автоматически удаляются по мере того, как изменения, представленные в журналах, переносятся в базу данных Active Directory. Каждый из этих журналов также занимает 10 Мб.
* Edbtemp.log- временный журнал, который используется тогда, когда заполнен текущий журнал (Edb.log). Новый журнал создается под именем Edbtemp.log, в нем хранятся все транзакции, а затем журнал Edb.log переименовывается в следующий предшествующий журнал. Далее журнал Edbtemp.log переименовывается в журнал Edb.log.
* Resl.log и Res2.log— резервные журналы, которые используются только в ситуации, когда на жестком диске заканчивается свободное пространство. Если текущий журнал заполнен, а сервер не может создать новый журнал, потому что на жестком диске нет свободного пространства, сервер подавит любые транзакции Active Directory, находящиеся в настоящее время в памяти, использует место для резервных журналов, а затем завершит работу Active Directory. Размер каждого из этих журналов также 10 Мб.

1. Где на контроллере домена хранятся файлы, содержащие групповые политики домена?

* GP Container (контейнер в AD), GP Template (domain\Policies\SysVol)

1. Какие компоненты автоматически устанавливаются мастером при добавлении ролей Active Directory?

* Только один тип раздела приложений каталога создается в Active Directory по умолчанию — это раздел, предназначенный для службы сервера доменной системы имен (DNS - Domain Name System). При установке первой интегрированной (integrated) зоны Active Directory создаются прикладные разделы каталога ForestDnsZones и DomainDnsZones. Раздел приложений каталога может хранить любой тип объекта Active Directory, кроме участников безопасности. Кроме того, разделы приложений каталога создаются для управления процессом репликации данных, и ни один из объектов раздела приложений каталога не может реплицироваться в раздел GC.

1. Для чего нужен пароль DSRM?

* DSRM позволяет администратору AD чинить или восстанавливать базу данных при необходимости, а пароль к DSRM позволяет иметь доступ/бэкдор к базе данных в случае поломок, но не дает доступ к домену или службам

1. Как восстановить пароль DSRM, если он был утерян после установки?

* С помощью утилиты ntdsutil (start->run->ntdsutil, пишем set dsrm password). В DSMR пишем reset password on server servername (null для локального компьютера)

1. Зачем нужно имя домена NetBIOS?

* Это сабдомен домена, например test в test.com, NetBIOS нужен для избежания конфликтов имен и для сохранения обратной совместимости со старыми версиями

1. Какие группы пользователей создаются в AD автоматически? Опишите минимум 5 из них.

* Administrators - полный доступ к компьютеру без запрещений
* Guests - почти то же, что и Users, но больше ограничений, при выходе из профиля, данные о нем удаляются
* Users - члены этой группы не могут делать изменения с системой, могут запускать большинство
* DHCP Administrators - члены этой группы могут создавать, удалять и управлять разными областями scope’а сервера, включая права на восстановление и бэкап Dynamic Host Configuration Protocol
* Domain Controller - включает в себя все контроллеры домена в домене, новые контроллеры добавляются сюда автоматически

1. Какие записи в DNS создаются специально для AD? Перечислите их, укажете их назначение.

* A(Host)
* CNAME(alias)
* SRV(service location)

1. **Замечания преподавателя (исправления, вызванные замечаниями преподавателя, также помещают в этот пункт)**