Alex Arnold

Professor Torres

4-7-25

IT 0115-41197

Lab 10

**Part A: Windows (Netlabve3Links to an external site.--> Lab 01)**

**Create a PowerShell script to:**

**add 10 users on a system (locally)**

**grant 2 users Local admin rights to the system**

# Define the base username and password

$BaseUsername = "TestUser"

$Password = ConvertTo-SecureString "P@ssw0rd123!" -AsPlainText -Force

# Create 10 users

for ($i = 1; $i -le 10; $i++) {

$Username = "$BaseUsername$i"

# Check if the user already exists

if (-not (Get-LocalUser -Name $Username -ErrorAction SilentlyContinue)) {

New-LocalUser -Name $Username -Password $Password -FullName "User $i" -Description "Local test user $i"

Write-Output "Created user: $Username"

} else {

Write-Output "User $Username already exists."

}

}

# Add TestUser1 and TestUser2 to the local Administrators group

$AdminUsers = @("TestUser1", "TestUser2")

foreach ($AdminUser in $AdminUsers) {

try {

Add-LocalGroupMember -Group "Administrators" -Member $AdminUser

Write-Output "Added $AdminUser to Administrators group."

} catch {

Write-Error "Failed to add $AdminUser to Administrators: $\_"

}

}

A screenshot of a computer

AI-generated content may be incorrect.

**Create a PowerShell script to:**

**add 10 users on a system (Domain)**

**grant 2 users Domain admin rights to the system**

# Load the Active Directory module

Import-Module ActiveDirectory

# Set base username and default password

$BaseUsername = "DomainUser"

$SecurePassword = ConvertTo-SecureString "P@ssw0rd123!" -AsPlainText -Force

# Set user OU - adjust to match your domain

$OU = "OU=Users,DC=Contoso,DC=com" # <-- Replace with your actual OU path

# Create 10 users

for ($i = 1; $i -le 10; $i++) {

$Username = "$BaseUsername$i"

$UserPrincipalName = "$Username@Contoso.com"

# Check if user already exists

if (-not (Get-ADUser -Filter { SamAccountName -eq $Username } -ErrorAction SilentlyContinue)) {

New-ADUser `

-Name $Username `

-SamAccountName $Username `

-UserPrincipalName $UserPrincipalName `

-AccountPassword $SecurePassword `

-Path $OU `

-Enabled $true `

-ChangePasswordAtLogon $false `

-GivenName "Test" `

-Surname "User$i" `

-DisplayName "Domain User $i" `

-Description "Automated test domain user"

Write-Output "Created user: $Username"

} else {

Write-Output "User $Username already exists."

}

}

# Grant Domain Admin rights to User1 and User2

$AdminUsers = @("DomainUser1", "DomainUser2")

foreach ($AdminUser in $AdminUsers) {

try {

Add-ADGroupMember -Identity "Domain Admins" -Members $AdminUser

Write-Output "Added $AdminUser to Domain Admins group."

} catch {

Write-Error "Failed to add $AdminUser to Domain Admins: $\_"

}

}

A blue screen with white text

AI-generated content may be incorrect.

**Use PowerShell to:**

**list out the users on a system (both local and domain)**

# List Local Users

Write-Output "`n--- Local Users ---"

Get-LocalUser | Select-Object Name, Enabled, LastLogon

if (Get-Module -ListAvailable -Name ActiveDirectory) {

Import-Module ActiveDirectory

Write-Output "`n--- Domain Users ---"

# List domain users (first 100 for demo)

Get-ADUser -Filter \* -Property DisplayName, Enabled |

Select-Object SamAccountName, DisplayName, Enabled -First 100

} else {

Write-Warning "Active Directory module not found. Cannot list domain users."

}

A screenshot of a computer

AI-generated content may be incorrect.

**Part B: Linux(Netlabve2Links to an external site.--> Lab 11b)**

**Create a Bash script to:**

**add 10 users on a system**

**grant 2 users sudo/root privileges on the system**

#!/bin/bash

# Base username and password (change password after first login!)

BASE\_USER="user"

PASSWORD="P@ssw0rd123"

# Create 10 users

for i in {1..10}; do

USERNAME="${BASE\_USER}${i}"

# Check if user already exists

if id "$USERNAME" &>/dev/null; then

echo "User $USERNAME already exists. Skipping..."

else

useradd -m "$USERNAME"

echo "$USERNAME:$PASSWORD" | chpasswd

echo "Created user: $USERNAME"

fi

done

# Grant sudo/root privileges to user1 and user2

ADMIN\_USERS=("user1" "user2")

for ADMIN in "${ADMIN\_USERS[@]}"; do

if id "$ADMIN" &>/dev/null; then

usermod -aG sudo "$ADMIN"

echo "Granted sudo access to $ADMIN"

else

echo "User $ADMIN does not exist. Skipping..."

fi

done

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer

AI-generated content may be incorrect.

**Create a Bash script that lists the users and their privileges on a system**

#!/bin/bash

echo "🔍 Listing local users and their privileges..."

echo "---------------------------------------------"

printf "%-20s %-10s %-20s\n" "Username" "UID" "Privileges"

echo "---------------------------------------------"

# Get all users with UID >= 1000 (normal users, not system accounts)

getent passwd | while IFS=: read -r user \_ uid \_ \_ \_ \_; do

if [[ "$uid" -ge 1000 && "$uid" -lt 65534 ]]; then

# Check if user is in the sudo group

if id -nG "$user" 2>/dev/null | grep -qw "sudo"; then

priv="sudo"

else

priv="standard"

fi

printf "%-20s %-10s %-20s\n" "$user" "$uid" "$priv"

fi

done

echo "---------------------------------------------"

A screenshot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Part C:**

**Create a PowerShell script that will list out all the applications installed on a system, to include versions**

# Combine both 32-bit and 64-bit uninstall registry paths

$registryPaths = @(

"HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\\*",

"HKLM:\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\\*",

"HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall\\*",

"HKCU:\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\\*"

)

$apps = foreach ($path in $registryPaths) {

Get-ItemProperty $path -ErrorAction SilentlyContinue | Where-Object {

$\_.DisplayName -and $\_.DisplayVersion

} | Select-Object DisplayName, DisplayVersion, Publisher, InstallDate

}

# Output results

Write-Output "`n--- Installed Applications ---"

$apps | Sort-Object DisplayName | Format-Table -AutoSize

A screenshot of a computer

AI-generated content may be incorrect.

**Create a bash script that will list out all applications installed on a system, to include versions**

#!/bin/bash

echo "🔍 Detecting system package manager..."

# Determine which package manager is available

if command -v dpkg &> /dev/null; then

echo "🧾 Debian-based system detected (dpkg)"

echo "Listing installed packages..."

dpkg -l | awk 'NR>5 {printf "%-40s %-20s\n", $2, $3}' | sort

elif command -v rpm &> /dev/null; then

echo "🧾 RHEL-based system detected (rpm)"

echo "Listing installed packages..."

rpm -qa --qf '%-40{name} %-20{version}-%{release}\n' | sort

else

echo "❌ No supported package manager found (dpkg or rpm required)."

exit 1

fi

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.