```
<#
    • Name: Yong Seung Rho (W0447442@nscc.ca)
    • Class: INET3700/Server Operating Systems and Scripting
    • Date: November 26th, 2021
    • Description:
        This PowerShell script performs user management on Windows as follows
        1.      Add a Local User
        2.      Change Password
        3.      Add a User to an Existing Local Group
        4.      Remove a User
        5.      Log all User Changes as an Event Log [Write-EventLog]
        6.      Include What-If functionality
#>

# Set computer name using env: variable
$ComputerName = $env:COMPUTERNAME

# Set log name for the event log
$LogName = "INET3700DA-Log"

# Set log name for the event log
$EventSource = "INET3700DA"

# AddWindowsUser with parameters for “UserName”, “Password”, “LocalGroup”
function AddWindowsUser ($username, $pword, $localgroup) {
    $op = Get-LocalUser | Where-Object Name -eq $username
    if (-not $op) {
        try {
            New-LocalUser -Name $username -Password $pword -Description "a new account for $username"
            Write-Host "$username has been created."
            Write-EventLog -ComputerName $ComputerName -EntryType SuccessAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username has been created."
        } catch {
            Write-Host "Oops, ran into an issue"
            Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "[AddWindowsUser]Oops, ran into an issue"
        }

        if ($null -ne $localgroup) {
            AddToLocalGroup $username, $localgroup
        }
    } else {
        Write-Host "$username already exists."
        Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username already exists."
    }
}

# ChangeUserPassword with parameters for “UserName”, “Password”
function ChangeUserPassword ($username, $pword) {
    $op = Get-LocalUser | Where-Object Name -eq $username
    if ($op) {
        try {
            Set-LocalUser -Name $username -Password $pword
            Write-Host "The password of $username has been changed."
            Write-EventLog -ComputerName $ComputerName -EntryType SuccessAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "The password of $username has been changed."
        } catch {
            Write-Host "Oops, ran into an issue"
            Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "[ChangeUserPassword]Oops, ran into an issue"
        }
    } else {
        Write-Host "$username does not exist."
        Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username does not exist."
    }
}

# AddToLocalGroup with parameters for “UserName”, “LocalGroup”
function AddToLocalGroup ($username, $localgroup) {
```

```
    $op = Get-LocalUser | Where-Object Name -eq $username
    if ($op) {
        if ($null -ne $localgroup) {
            $op = Get-LocalGroup | Where-Object Name -eq $localgroup
            if (-not $op) {
                Write-Host "LocalGroup $localgroup does not exist."
                Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "The LocalGroup $localgroup does not exist."
            } else {
                $groupObj =[ADSI]"WinNT://$ComputerName/$localgroup,group"
                $membersObj = @($groupObj.psbase.Invoke("Members"))
                $members = ($membersObj | foreach {$_.GetType().InvokeMember("Name", 'GetProperty', $null, $_,
$null)})
                if ($members -contains $username) {
                    Write-Host "$username exists in the LocalGroup $localgroup"
                    Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -
Source $EventSource -EventID 7001 -Message "[AddToLocalGroup]$username exists in the LocalGroup $localgroup"
                } else {
                    # Write-Host "$username not exists in the group $localgroup"
                    try {
                        Add-LocalGroupMember -Group $localgroup -Member $username
                        Write-Host "$username has been added to the LocalGroup $localgroup."
                        Write-EventLog -ComputerName $ComputerName -EntryType SuccessAudit -LogName $LogName -
Source $EventSource -EventID 7001 -Message "$username has been added to the LocalGroup $localgroup."
                    } catch {
                        Write-Host "Oops, ran into an issue"
                        Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -
Source $EventSource -EventID 7001 -Message "[AddToLocalGroup]Oops, ran into an issue"
                    }
                }

            }
        }
    } else {
        Write-Host "$username does not exist"
        Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username does not exist."
    }
}

# RemoveWindowsUser with parameters for “UserName”
function RemoveWindowsUser {
    param (
        [string]$username,
        [switch]$WhatIf
    )

    $op = Get-LocalUser | Where-Object Name -eq $username
    if ($op) {
        if ($WhatIf.IsPresent) {
            # WhatIf switch is on.
            Remove-LocalUser -Name $username -WhatIf
        } else {
            # WhatIf switch is off.
            try {
                Remove-LocalUser -Name $username
                Write-Host "$username has been removed."
                Write-EventLog -ComputerName $ComputerName -EntryType SuccessAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username has been removed."
            } catch {
                Write-Host "Oops, ran into an issue"
                Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "[RemoveWindowsUser]Oops, ran into an issue"
            }
        }
    } else {
        Write-Host "$username does not exist."
        Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username does not exist."
    }
}
```

```
# RemoveFromLocalGroup with parameters for “UserName”, “LocalGroup”
function RemoveFromLocalGroup ($username, $localgroup) {
    $op = Get-LocalUser | Where-Object Name -eq $username
    if ($op) {
        if ($null -ne $localgroup) {
            $op = Get-LocalGroup | Where-Object Name -eq $localgroup
            if ($op) {
                try {
                    Remove-LocalGroupMember -Group $localgroup -Member $username
                    Write-Host "$username has been removed from the LocalGroup $localgroup."
                    Write-EventLog -ComputerName $ComputerName -EntryType SuccessAudit -LogName $LogName -
Source $EventSource -EventID 7001 -Message "$username has been removed from the LocalGroup $localgroup."
                } catch {
                    Write-Host "Oops, ran into an issue"
                    Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -
Source $EventSource -EventID 7001 -Message "[RemoveFromLocalGroup]Oops, ran into an issue"
                }
            } else {
                Write-Host "The LocalGroup $localgroup does not exist."
                Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "The LocalGroup $localgroup does not exist."
            }

        }
    } else {
        Write-Host "$username does not exist."
        Write-EventLog -ComputerName $ComputerName -EntryType FailureAudit -LogName $LogName -Source
$EventSource -EventID 7001 -Message "$username does not exist."
    }
}

# Create an event log
$op = Get-EventLog -LogName $LogName
if (-not $op) {
    Write-Host "<=== New-EventLog ===>"
    New-EventLog -ComputerName $ComputerName -LogName $LogName -Source $EventSource
}

# Perform user management
do {
    Write-Host "================================================"
    Write-Host "1. Add a Local User"
    Write-Host "2. Change Password"
    Write-Host "3. Add a User to an Existing Local Group"
    Write-Host "4. Remove a User from an Existing Local Group"
    Write-Host "5. Remove a User"
    Write-Host "6. Clear the event log"
    Write-Host "7. Show the event log"
    Write-Host "0. Exit"
    Write-Host "================================================"
    $option = Read-Host "Enter an option"
    switch ($option) {
        1 {
            Write-Host "<=== AddWindowsUser ===>"
            $UserName = Read-Host "Enter a new user"
            $UserPassword = Read-Host "Enter a new password" -AsSecureString
            AddWindowsUser $UserName $UserPassword
        }
        2 {
            Write-Host "<=== ChangeUserPassword ===>"
            $UserPassword = Read-Host "Enter a new password to change" -AsSecureString
            ChangeUserPassword $UserName $UserPassword
        }
        3 {
            Write-Host "<=== AddToLocalGroup ===>"
            #LocalGroup for test
            $LocalGroup = Read-Host "Enter a LocalGroup"
            AddToLocalGroup $UserName $LocalGroup
        }
        4 {
            Write-Host "<=== RemoveFromLocalGroup ===>"
            RemoveFromLocalGroup $UserName $LocalGroup
```

```
        }
        5 {
            Write-Host "<=== RemoveWindowsUser ===>"
            RemoveWindowsUser $UserName -WhatIf
            $RemoveUser = Read-Host "Do you really want to remove the user? (y/n)"
            if ($RemoveUser -eq "Y") {
                RemoveWindowsUser $UserName
            }
        }
        6 {
            Write-Host "<=== ClearEventLog ===>"
            Clear-EventLog -ComputerName $ComputerName -LogName $LogName
        }
        7 {
            Write-Host "<=== ShowEventLog ===>"
            Get-EventLog -ComputerName $ComputerName -LogName $LogName
        }
        0 {
            Write-Host "Good bye!!!"
            break
        }
        default {
            Write-Host "Please select an option again"
        }
    }
} while (0 -ne $option)
```