Lab 4: User Management

East Tennessee State University

CSCI 4417/5417: Introduction to System Administration

Spring 2016

Pramod Nepal

Purpose

To explore the process of adding users, deleting users and elevating privileges on Windows and Ubuntu

Materials

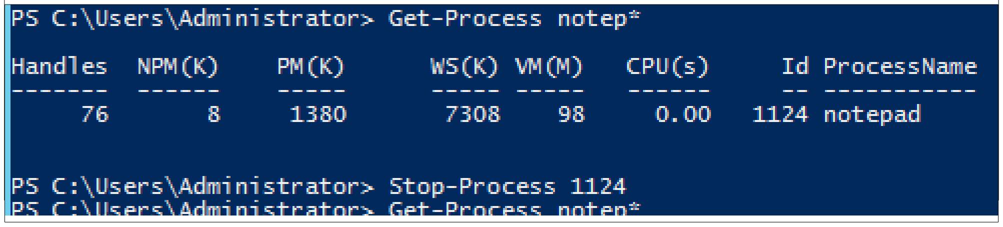
- Windows Server 2012 R2 AWS Instance
- Ubuntu 14.04 AWS Instance
- Provided files i.e., museradd.sh, users.txt and users.csv

Procedure and Results

Windows

After launching the Windows Server VM, PowerShell was launched. It was made sure that PowerShell was running with Administrative privilege. The title bar read 'Administrator: Windows PowerShell' to indicate this. The first task that was done was update the help files with **update-help** command. The help files provide instructions and examples for running different commands (also known as cmdlets). PowerShell provides tab completion for commands just like in Bash shell in Linux. One notable difference of PowerShell to Linux commands is that the commands are case-insensitive. So 'get-help or 'Get-Help' refers to the same command. Next, process management cmdlet was explored. Notepad was started. From PowerShell **Get-Process notep*** command was run (see Fig. 1).

Figure: 1 Running Get-Process and Stop-Process commands



```
PS C:\Users\Administrator> Get-Process notep*

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     76       8     1380       7308    98     0.00   1124 notepad


PS C:\Users\Administrator> Stop-Process 1124
PS C:\Users\Administrator> Get-Process notep*
```

This command lists the process Id and the name of executing processes (notepad in this case) among other parameters just like the **top** command in Linux. The process can be killed using the **Stop-Process**

<Id> command (see Fig 1). There is no equivalent of **killall** command. In other words **Stop-Process** cannot kill an application using process name.

PowerShell Integrated Scripting Environment

PowerShell supports scripting. PowerShell Integrated Scripting Environment (ISE) provides help on writing such scripts. ISE was launched by right-clicking the PowerShell icon on the taskbar and selecting 'Run ISE as Administrator'. Clicking the Script button on the top right launched the Script Pane. Following script was written in the Script Editor and saved as **museradd.ps1**.

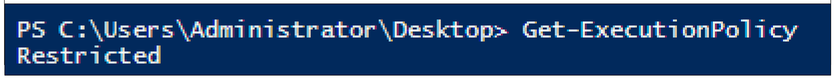
```
Import-Csv users.csv | New-ADUser -PassThru | Set-ADAccountPassword `

-Reset -NewPassword (ConvertTo-SecureString -AsPlainText 'Passw0rd!' `
-Force) -PassThru | Enable-ADAccount
```

Te script reads a list of users from users.csv file. Each line contains Name,GivenName,Surname,SamAccountName and UserPrincipalName. The SamAccountName is a username that has maximum length of 20 characters. It is used to support legacy versions of Windows like Windows 98 and earlier. UserPrincipalName column contains the email address for Active Directory domain server. Above command also uses pipes to forward the output of each cmdlet to the following cmdlets. Among many things the script imports the users from users.csv, adds them to Active Directory and enables the account. Before executing the file UserPrincipalName column had to be changed so that it reflected the Active Directory domain created by each student in the lab.

One more thing that needed to be done before executing **museradd.ps1** was to change the PowerShell execution policy. PowerShell has four execution policies named Restricted, AllSigned, RemoteSigned and Unrestricted. The default policy in PowerShell can be seen by executing **Get-ExecutionPolicy** command (see Fig. 2).

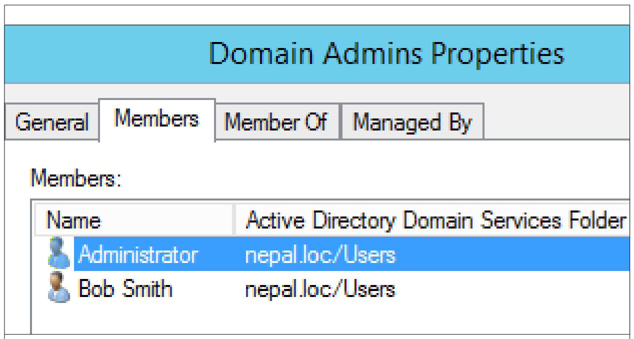
Figure 2: Getting default policy of PowerShell



```
PS C:\Users\Administrator\Desktop> Get-ExecutionPolicy
Restricted
```

Execution policy was changed to RemoteSigned using **Set-ExecutionPolicy RemoteSigned** command. RemoteSigned policy allows the local script to be run without any prompts. It is the default execution policy in Server 2012 R2, but in a Desktop the default policy was Restricted.

Once the script was run from PowerShell ISE, Server Manager was opened. When 'Active Directory Users and Computers' was browsed the newly added users could be seen by clicking the Users folder. Next, the Security Groups was examined by double clicking the Domain Admins. Domain Admins Properties tab show tabs named General, Member, MemberOf and ManagedBy. Under the Members tab 'Administrator' and 'Bob Smith' were shown to be domain administrators (see Fig. 3). Administrator is the default user with administration privileges created when the Operating System was installed. User 'Bob Smith' was added as Domain Admin in the previous lab during the Active Directory installation task.


Domain Admins Properties

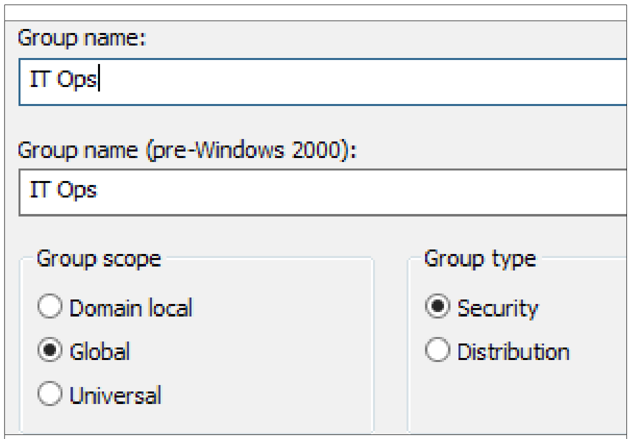
General | Members | Member Of | Managed By

Members:

Name	Active Directory Domain Services Folder
Administrator	nepal.loc/Users
Bob Smith	nepal.loc/Users

Figure 3: Domain Admins

The lab also experimented with creating new Security Groups. The Users folder in the left pane was right-clicked and from the menu Group was selected under New menu option. The group was named 'IT Ops' (see Fig. 4)


Group name: IT Ops

Group name (pre-Windows 2000): IT Ops

Group scope: Domain local [✓ Global] Universal

Group type: [✓ Security] Distribution

Figure 4: Creating IT Ops Security Group

As can seen from Fig.4. there are two group types. Security group is used to control access to resources. Distribution group have similar permissions as Security group, but it cannot be used for access control. Both groups can be used as e-mail distribution lists. There are three scopes for each group types. Universal group can be used anywhere in the same Windows forest. Global group is the primary scope in which users are placed in Mixed-mode domains. Domain Local group can be used for assignment of access policies on resources in Active Directory such as file server shares and printer queues. Domain Local group should not be used to assign permission on Active Directory objects, because they cannot be evaluated in other domains. Bob Smith was added to this group. To add Bob to IT Ops group, from the properties window Members tab was selected and the Add button was clicked. This displayed the search window. Users could be searched using Advanced option, but 'Bob' was searched in the presented search window and added by clicking the Add button. Added users can be confirmed by visiting the Members tab from the Properties window of the group.

Another way to create PowerShell Scripts

Another way of creating PowerShell scripts was also explored. Instead of writing the script in PowerShell ISE, following script to delete a list of users was written in notepad.

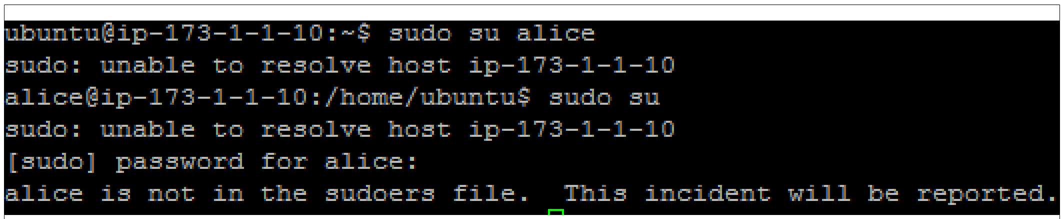
```
Import-Csv users.csv | foreach-object {
remove-adduser -Identity $_.$amAccountName -Confirm:$false
}
```

The script was saved as dusers.ps1 in Desktop. In PowerShell CLI Desktop directory was visited and .**dusers.ps1** was run. It was also made sure that users.csv file was in Desktop. Before executing the dusers.ps1 file 'Get-ADUser -Filter * > currentusers.txt' command was used to read the list of users. The output of the list command was redirected to currentusers.txt file. The **Get-ADUser** command was run before and after executing the dusers.ps1 file to confirm that the users have been deleted with the

PowerShell script. Get-Aduser-Filter lists one user named **krbtgt**. The Kerberos Key Distribution Center encrypts a user's session ticket with a key it derives from the password of the krbtgt user. This is used to provide access to the Operating System. Therefore, krbtgt user should not be deleted from AD DS.

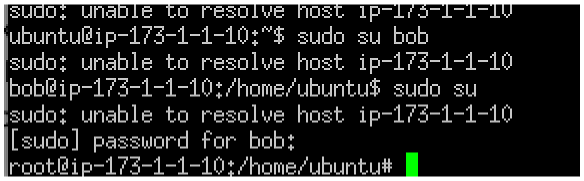
Ubuntu

Second part of the lab was to experiment with user management in Ubuntu. After login into Ubuntu a user name **alice** was created using 'sudo adduser alice' command. Another user named 'Bob Thomas' was also created using the adduser command. Bob was also added to the sudo group using 'sudo usermod -G sudo bob' command. Bob could also have been added to the sudo group during account creation using 'sudo adduser bob sudo' command. A user can also be given administrative privileges by editing the sudoers file using 'visudo' command. If a user does not belong to sudo group, sudo command cannot be added to alleviate the privileges to root while running a command. This can be seen by trying to switch the user from alice account to root (see Fig. 5). Since Bob was in sudo group, sudo command could be used to switch to the root user from Bob's shell (Fig. 6).



```
ubuntu@ip-173-1-1-10:~$ sudo su alice
sudo: unable to resolve host ip-173-1-1-10
alice@ip-173-1-1-10:/home/ubuntu$ sudo su
sudo: unable to resolve host ip-173-1-1-10
[sudo] password for alice:
alice is not in the sudoers file.  This incident will be reported.
```

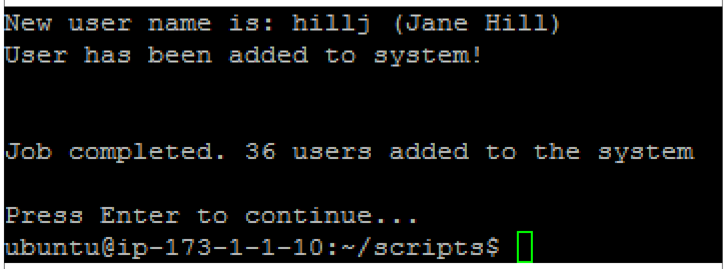
Figure 5: Switching users failed for alice



```
sudo: unable to resolve host ip-173-1-1-10
ubuntu@ip-173-1-1-10:~$ sudo su bob
sudo: unable to resolve host ip-173-1-1-10
bob@ip-173-1-1-10:/home/ubuntu$ sudo su
sudo: unable to resolve host ip-173-1-1-10
[sudo] password for bob:
root@ip-173-1-1-10:/home/ubuntu#
```

Figure 6: Figure 6: Switching to root from sudo group user

After examining /etc/passwd it was found that user accounts uid starts from 1000. Next task in the lab was to automate user addition and deletion using two scripts. Both of these scripts read the user list from users.txt file. Each line of users.txt contained users listed as "**username Firstname Lastname**" . User addition script 'museradd.sh' was transferred to the Ubuntu server using 'psftp' command. First the connection was established with '**psftp -i 4417key.ppk <ip-address>'** command. The files were transferred using '**put <path/filename>** command from the ftp shell. The files copied to the server could be verified with 'ls' command. The file users.txt was also transferred to the server. Since the transferred files don't have execute permission, 'chmod 775 museradd.sh' command was used to make the script executable. This command gives read/write/execute permission to user and group. It also gives read and execute permission to others. Files in current working directory can be executed by prepending the file with './'. To add users in users.txt 'sudo ./museradd.sh users.txt' command was executed (see Fig. 7). The **useradd** command in museradd.sh creates each user using 'pass' as password text. It creates a home directory and sets default shell for the user. The mail directory for each user was added using **mkdir** command. After the directory was created the user and group permissions were set to username.



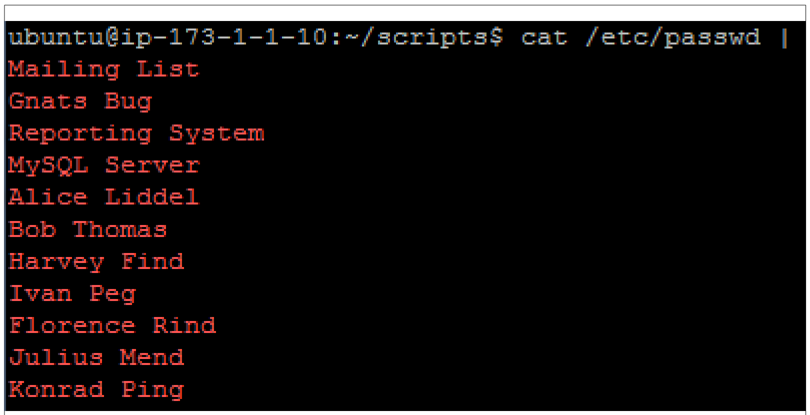
```
New user name is: hillj (Jane Hill)
User has been added to system!


Job completed. 36 users added to the system

Press Enter to continue...
ubuntu@ip-173-1-1-10:~/scripts$
```

Figure 7: Add users using museradd.sh

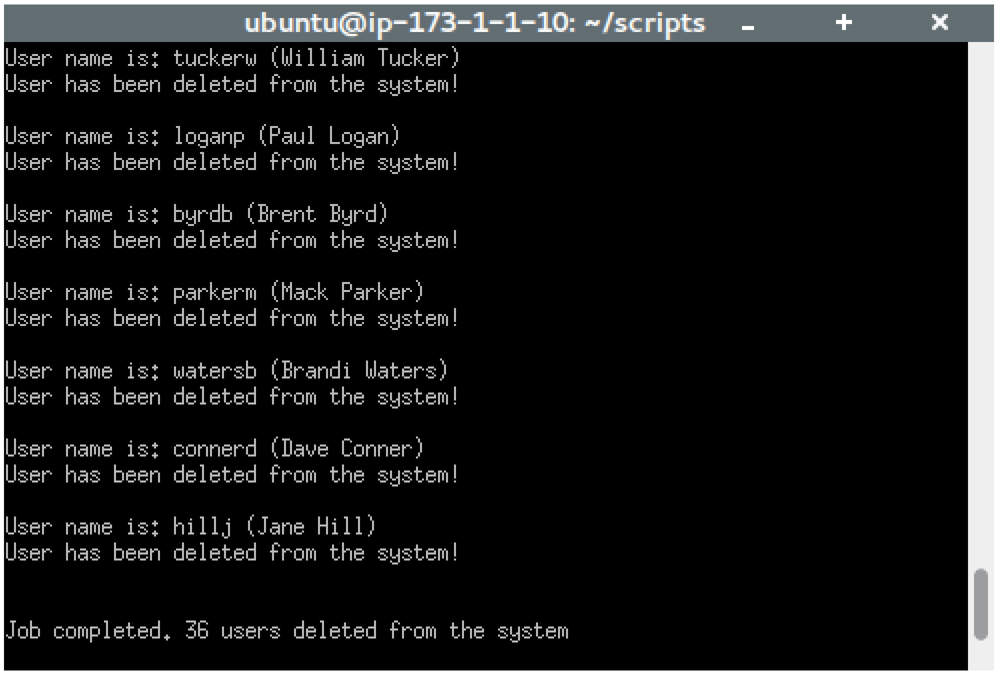
The added users can be verified by examining /etc/passwd file (see Fig. 8).



```
ubuntu@ip-173-1-1-10:~/scripts$ cat /etc/passwd |
Mailing List
Gnats Bug
Reporting System
MySQL Server
Alice Liddel
Bob Thomas
Harvey Find
Ivan Peg
Florence Rind
Julius Mend
Konrad Ping
```

Fig 8: Verify added users

Next part of the assignment was to delete the users using another script called 'duser.sh'. The duser.sh script also used users.txt to read the users to delete from the server. Code from museradd.sh was modified to write duser.sh. First the /etc/passwd file was checked for usernames that matched the names from users.txt. Once such a user was found, the mail directory for that user was deleted using 'rm -rf /var/mail/<username>' command. The user was then deleted with 'userdel <username>' command. Finally, the home directory of the user was deleted with 'rm -rf /home/<username>' command. This script was run in the same directory that contained museradd.sh and users.txt using 'sudo ./dusers.sh users.txt' command (see Fig. 9).



```
User name is: tuckerw (William Tucker)
User has been deleted from the system!

User name is: loganp (Paul Logan)
User has been deleted from the system!

User name is: byrdb (Brent Byrd)
User has been deleted from the system!

User name is: parkerm (Mack Parker)
User has been deleted from the system!

User name is: watersb (Brandi Waters)
User has been deleted from the system!

User name is: connerd (Dave Conner)
User has been deleted from the system!

User name is: hillj (Jane Hill)
User has been deleted from the system!


Job completed. 36 users deleted from the system
```

Figure 9: Deleting users using dusers.sh script

Observations

In Windows, like in Linux system commands should be run as an Administrator. For this reason it is made sure PowerShell starts with Administrator privilege. Another part of the the Admin's job is also to make sure that they explore new functionalities and parameters added to their favorite commands. It is for this reason the help file for the commands need to be updated. The help files could also update descriptions for existing commands. Being able to manage accounts (which was seen in later part of the lab) and processes from CLI is a great asset for a System Admin. PowerShell is a powerful tool, but it is not as extensive as Linux CLI. It was seen that a process could be killed with process Id, but not using the process's name. It would be more convenient to kill a process using its name just like **killall** command in Linux.

PowerShell Integrated Scripting Environment is a power tool that can be used to write and execute scripts from within the application. It provides context sensitive help and ability to search for cmdlets. Color coding in the editor also greatly helps while writing a script. Being able to read user's from a csv file while creating or deleting the users make life a System Admin little less hectic. It saves the Admin from having to type and click through countless GUI windows. The security provided by the PowerShell execution policy is a great addition that protects the system from having to run random scripts. After executing a script the Admin can change the policy to Restricted so that scripts are not executed without the user (with Administrator previlege) not knowing the implications.

Another protection that can be experimented are group types and group scopes. Group scopes can be changed in Active Directory from security group. Different levels of restrictions can be applied to different groups which helps to build a healthy domain for users with different usage of the system.

One of the powerful features of Linux as hinted above is to be able do almost everything (at least system administration tasks) from CLI. System Admin should be able to automate every day tasks using the power of Shell scripts (like Bash) and all the commands that is installed or could be installed in Linux. In addition to Bash several high level languages such as Python and Ruby can be used to automate work that contains complex data set and calculation.