HRPCUG

The power of PowerShell

Created by Ryan Woodward

North Central Missouri College

Table of Contents

H
R
P
C
U
G

About me

- Name: Ryan Woodward
- Occupation: Computer Wizard (of the Network/Security variety)
- I have worked at NCMC between 2 and 3 years.
- Since starting here, I have been a Database administrator, Systems administrator, and currently our Network/Security administrator.

H
R
P
C
U
G

What is PowerShell?

- What is PowerShell?
  - PowerShell is what happens when command prompt takes steroids.
  - This powerhouse can do almost anything command prompt can do, plus much more in a more script friendly environment.
  - For those who have used Linux before, PS may seem familiar as it is kind of Microsoft's take on the popular Linux's BASH Shell.

- Where has PS been my whole life?
  - It's fairly new when compared to other scripting/programming languages, PowerShell 1.0 came out in November 2006.
  - Current version of PowerShell is PowerShell 5.1.
  - PowerShell Core 6.0 is also out.
    - New "variety" of PowerShell that is compatible with Mac and Linux machines.
    - Unlike normal PowerShell, Core is not always backwards compatible.

Installing/Starting PowerShell

- How do I install PowerShell?
  - Good News! If you have a Windows 7, Windows Server 2008 or newer operating system, a version of PowerShell should already be installed by default.
  - On servers, PowerShell can be manually installed through roles and features in the server manager.
  - In order to install on desktop machines, and to upgrade PowerShell, the Windows Management Framework will need to be downloaded and installed.
    - Free from Microsoft's website.
    - Standard PowerShell is backward compatible.

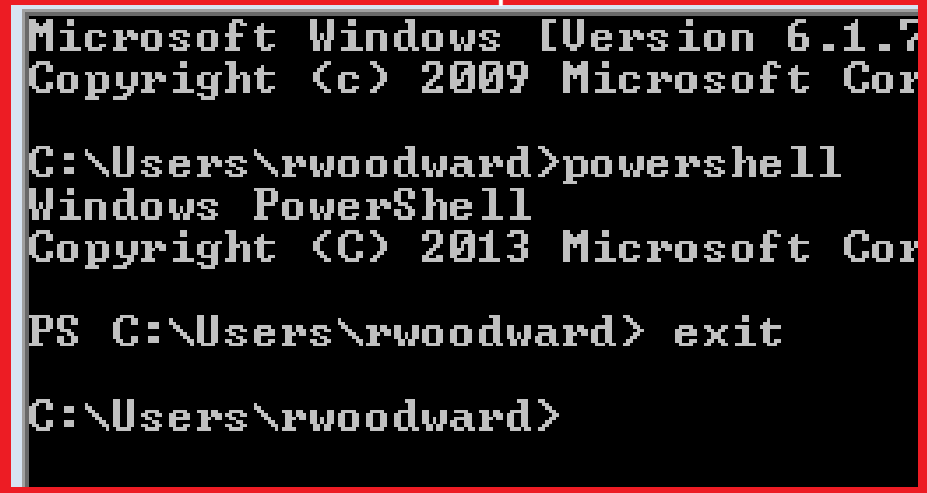
Installing/Starting PowerShell (Cont.)

- Normal PowerShell VS PowerShell ISE
  - A normal PowerShell window looks like a cmd prompt window with a PS to the left of the input prompt.
    - Typing “powershell” into a cmd window will turn it into a PS window.
  - Reads one line at a time as you hit enter.

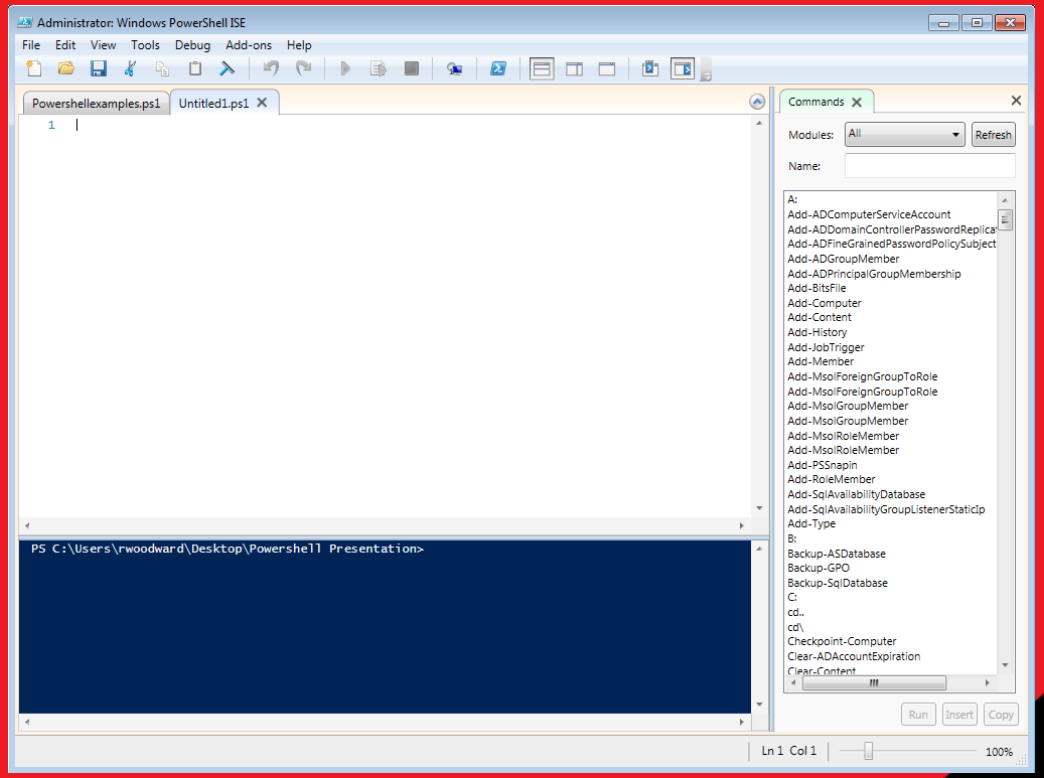
  - PowerShell ISE (Interactive Scripting Environment)
    - Program that incorporates both a text editor and a PowerShell prompt.
    - Scripts can be created in the text editor and then ran, or partially ran, in the PowerShell Prompt.
      - The text editor will point out some errors, along with offering some auto complete options.

Installing/Starting PowerShell (Cont.)

Prompt



ISE



PowerShell Basics
- Variables

HRPCUG

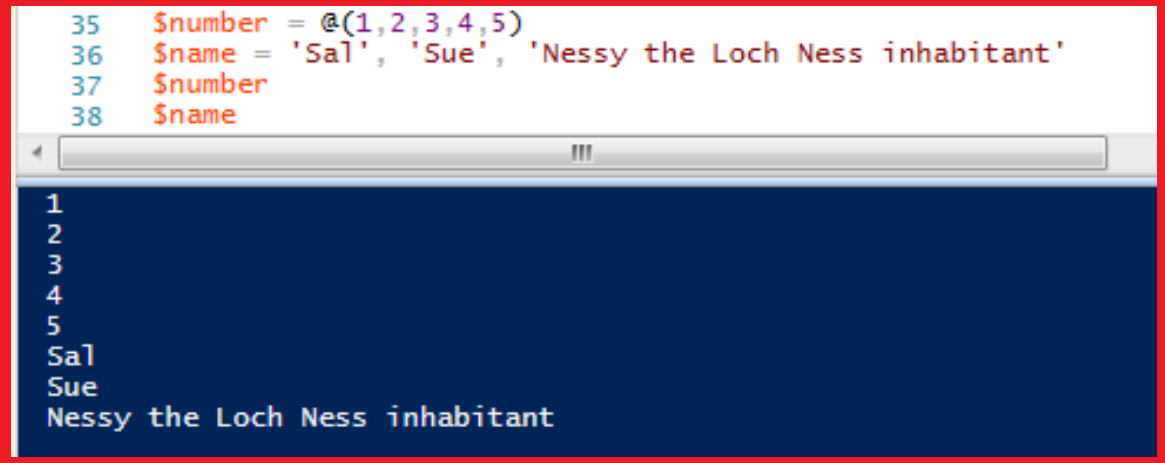
- Variables – scripting/programming storage containers ($)
  - A variable is where PowerShell (and other scripting/programming languages) stores data. This can include data produced/manipulated by PowerShell or data from an external source like a .txt file.
  - A variable is declared when a '$' is put in front of a word.
    - For example "$dog" declares a variable.
    - Following a variable with an "=" symbol will assign the following value to that variable, for example "$dog = 'Sally'" will assign Sally to $dog. Anytime $dog is called it will be replaced with this value.
  - A variable containing a string (word or series of words) needs to be surrounded by single or double quotes.

```
$yes = 'yes'
$fifty = 49
$state = "We are in Missouri"
$date = get-date
```

PowerShell Basics
- Arrays

HRPCUG

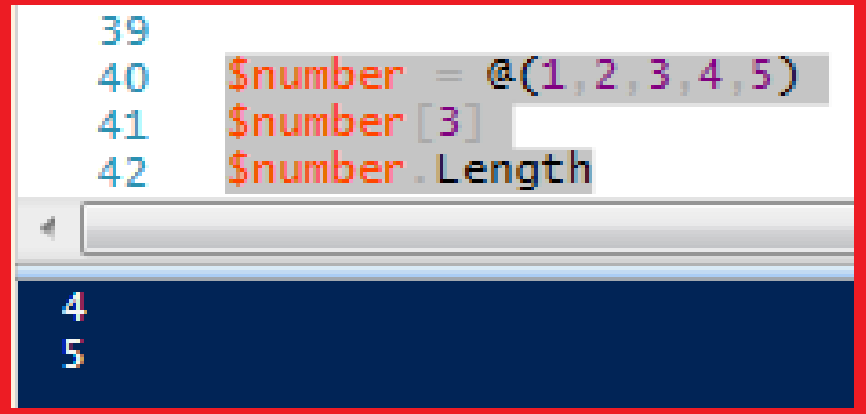
- Arrays - Simple way to group object, variables, or whatever (@())
  - Arrays are used to group/store data in an easy to use data structure.
  - Can be used to store variables, data, or something more complex like an AD object or output from a SQL command.
  - Can be created the same way as a variable ($something =), but the variable is equal to several members. For example: $something = ‘Everything’, ‘Nothing’, ‘Sort of something’
  - Can also be declared with @(something) for example: $numbers = @(1,2,3,4,5)



PowerShell Basics – Arrays (cont.)

HRPCUG

- $numbers = @(1,2,3,4,5)
- Data can be retrieved from an array by the array name followed by [#] (no space between the name and [#]), where # is a number corresponding to the spot in the array. For example: $number[3] will return 4.
  - Why 4 and not 3? Arrays start counting at 0, so the 1 in the array is actually in the 0's spot.
- The length of an array can be found by running the array name followed by ".length" or ".count"



PowerShell Basics
- cmdlets

- cmdlets - Scripting done easy (verb-noun)
  - Cmdlets are pre-constructed bits of code (PowerShell version of functions) that, when called, will do something as simple as outputting data to the screen or something more complex like running a mathematical formula on a set of data or copying files to a new location.
  - For ease of use, cmdlets come in a verb-noun format.
    - For example, “get-date” will output the current date to the prompt.

HRPCUG

```
23    get-module
24    get-date
25    Add-Content
26    Remove-ADGroup
27    Add-ADGroupMember
```

```
45    get-date

Wednesday, June 13, 2018 4:18:19 PM
```

Powershell Basics – cmdlets (cont.)

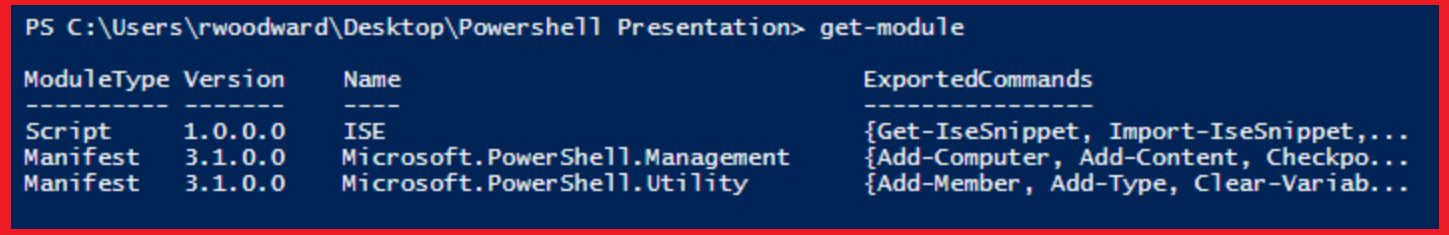
- Some cmdlets require just the cmdlet to work, others require additional arguments to get the desired results.
  - The standard format for adding arguments is: “verb-noun –argumentname “argument” –argumentname2 “arugment2”…”
- A “pipe” can be used to combine two different cmdlets in a way that allows them to flow together.
  - The pipe symbol is the vertical line above the return key on standard keyboards (|).
  - Placing this between two cmdlets, where the output from the first cmdlet can be used as input for the second cmdlet, will cause the resulting data to flow from one cmdlet into another without worrying about additional scripting.
  - For example: “get-date | out-file –filepath C:\folder\file.txt” will get the date/time and output it into a file called file.txt

```
get-date | out-file -filePath C:\folder\file.txt
```

Powershell Basics
- Modules

HRPCUG

- Modules - The power behind PowerShell (Import-module)
  - PowerShell has an extensive library of pre-constructed code to use, much of which comes in convenient modules.
  - Various Cmdlets are stored in these modules; import the module and its cmdlets become easily available.
  - Many of these modules need to be imported at the top of the script before they can be used.
  - For example, the command “import-module activedirectory” will import many useful cmdlets that can be used with Active Directory (Domain controllers have this module imported to PS by default)
  - Running the “Get-module” command will list modules that you currently have installed, and modules that you can install. Additional modules can be downloaded and installed.



Powershell Basics – If and For loops

HRPCUG

- If and For loops – Standard programming with a slight twist
  - Like most other languages, PowerShell can use if and for loops to iterate through data, however there are a few differences.

  - If statement – A logical bit of code that, if true, will do one thing, otherwise it will do something else or do nothing.
    - An if statement in a different language may look like
      if ($one == 1) {$two = $one + $one} else {$two = $two}
    - In PowerShell, the '==' needs to be replaced with –eq (or –ne, -like, etc)
      if ($one -eq 1) {$two = $one + $one} else {$two = $two}

PowerShell Basics – If and For loops (cont.)

- For loop – a way to iterate through a series of values
  - PowerShell can use a more traditional for-loop, or it can use a convenient piece of code called “foreach”.
  - A standard for-loop may look like:
    for ($number = 1; $number -lt 10; $number +=1)
    {$currentNumber = “The current number is $number”}

  However, if we are iterating through an array, we can use a foreach loop to easily grab each value in the array without worrying about array length or keeping track of where in the array a value is. For Example:
  $places = ‘here’, ‘there’, ‘somewhere’, ‘anywhere’
  Foreach ($place in $places)
  {“Its not $place”}

PowerShell Basics – If and For loops (cont.)

HRPCUG

Standard for-loop

```
54   for ($number = 1; $number -lt 10; $number +=1)
55       {$currentNumber = "The current number is $number"
56        $currentNumber
57       }
```

```
The current number is 1
The current number is 2
The current number is 3
The current number is 4
The current number is 5
The current number is 6
The current number is 7
The current number is 8
The current number is 9
```

Foreach for-loop

```
50   $places = 'here', 'there', 'somewhere', 'anywhere'
51   Foreach ($place in $places)
52   {"Its not $place"}
```

```
Its not here
Its not there
Its not somewhere
Its not anywhere
```

PowerShell Basics – Saving data

- There are several ways to save data from PowerShell.
- Using the “out-file’ cmdlet will write simple data out to a file.
  Out-file –filepath C:\folder\file.txt
- The “export-csv” command can be used to export a group of data out into a structured csv file.

More advance PowerShell tricks - SQL

- How can PowerShell help me with my Power Campus duties?
  - If you are like me, sometimes you like to go directly to the database to get what you need.
  - PowerShell cmdlets are able to easily access SQL databases to pull out information, such as user ids and names for accounts, or security questions for easy accounts unlocks.
  - The following example is a portion of a script we use to check AD accounts, reset passwords/unlock AD account, and create new student AD accounts using information we pull from our PowerCampus database.

More advance PowerShell tricks – SQL (cont.)

```
if ($x -match "2"){
    cls
    $r = read-host Please enter student ID
    sqlcmd -S ncmcpcampus-85 -d campus6 -Q "select people_id,maiden,city from userdefinedind where people_id = $r"
    $answer = read-host "Did student correctly verify? (y/n):"

    if ($answer -match "y"){


        echo "Action,Type,Name,EmailAddress,Password,FirstName,LastName,DisplayName,ForceChangePassword" | out-file -encoding ascii \\ncmcdc1\liveedu\resetadpassword$r.csv

        sqlcmd -S ncmcpcampus-85.office.ncmissouri.edu -h -1 -s "," -d campus6 -Q "set nocount on
        set ansi_warnings off

        select distinct 'Add' as 'Action',
           'Mailbox' as 'Type',
           people.people_id as 'Name',
           people.people_id + '@pirates.ncmissouri.edu' as 'EmailAddress',
           convert(varchar,datepart(YYYY, People.birth_date)) + right(people.government_id,4) as 'Password',
           people.first_name as 'FirstName',
           left(people.last_name,1) as 'LastName',
           rtrim(people.FIRST_NAME) + ' ' + left(people.last_name,1) as 'DisplayName',
           '0' as 'ForceChangePassword'
        from PEOPLE
           INNER JOIN ACADEMIC ON PEOPLE.PEOPLE_CODE_ID = ACADEMIC.PEOPLE_CODE_ID
           LEFT OUTER JOIN USERDEFINEDIND ON PEOPLE.PEOPLE_CODE_ID = USERDEFINEDIND.PEOPLE_CODE_ID
        where people.people_id = '$r'" -o "\\ncmcdc1\liveedu\resetadpassword$r.txt" -W

        $temp = get-content \\ncmcdc1\liveedu\resetadpassword$r.txt

        echo $temp | Out-File -Encoding ascii \\ncmcdc1\liveedu\resetadpassword$r.csv -append

        $userreset = Import-Csv \\ncmcdc1\liveedu\resetadpassword$r.csv

        foreach ($user in $userreset){
            $userid = $user.name
            $password = $user.password

            $session = new-PSSession ncmcstudc1.labs.ncmissouri.edu -credential $credential@labs.ncmissouri.edu

            invoke-command -session $session -ScriptBlock { Import-Module activedirectory }
            invoke-command -session $session -scriptblock { Set-ADAccountPassword $args[0] -Reset -NewPassword (ConvertTo-SecureString -AsPlainText $args[1] -Force) } -argumentlist $useri
            invoke-command -session $session -scriptblock { unlock-adaccount $args[0] } -ArgumentList $userid
            invoke-command -session $session -ScriptBlock { get-aduser $args[0] -properties * | select name,samaccountname,passwordlastset,lockedout } -argumentlist $userid
            Remove-PSSession $session
            del \\ncmcdc1\liveedu\resetadpassword$r.csv
            del \\ncmcdc1\liveedu\resetadpassword$r.txt
            read-host "If no errors are displayed above, then password was successfully reset to $password and account was unlocked if locked.  Press enter to return to main menu."
        mainmenu
    }
        }

        if ($answer -match "n"){ mainmenu }


        }
```

Select the #2 option for unlocking account

Query the PowerCampus database for their security questions

If answered right, query the information used for their password

Resets the password for the account and unlocks it

If answered wrong, do nothing

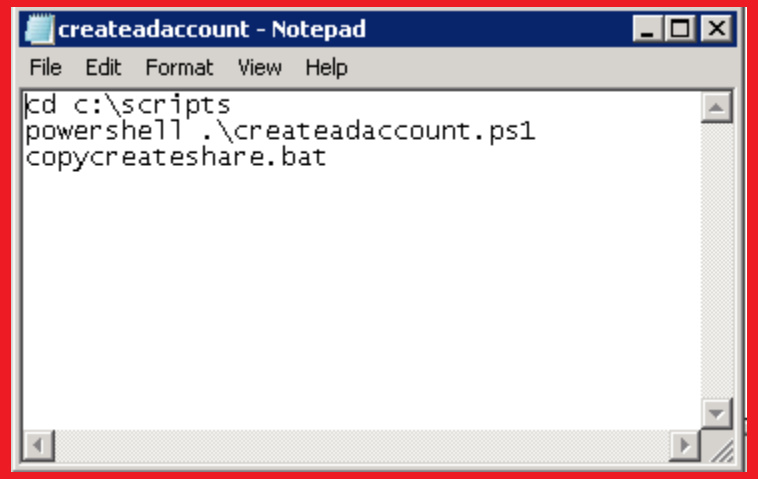
More advance PowerShell tricks – SQL (cont.)

Another script we used to find all the ‘BAD ADDRESS’s in our database and list them in a .txt file

```
<#
This code will be used to find and print out anny accounts that had "bad address" as their first address from ca
#>


$uid = "[redacted]"
$pass = ""
$listOfBadAddresses = @()

$conn = New-Object System.Data.SqlClient.SqlConnection
$conn.connectionString = "Server=;Database=Campus6;User ID=$uid;Password=$pass"
$conn.open()

for ($i = 0; $i -lt 353; $i++){
$badAddress = "Select PEOPLE_ORG_ID, FIRST_NAME, MIDDLE_NAME, LAST_NAME, ADDRESS_LINE_2, CITY, STATE, ZIP_CODE
from people join address on people.PEOPLE_ID = PEOPLE_ORG_ID where ADDRESS_LINE_1 = 'BAD ADDRESS'"
$command = New-Object system.data.sqlclient.sqlcommand($badAddress, $conn)
$reader = $command.executeReader()
$rdr = $reader.read()

$listOfBadAddresses += $reader[0].tostring() + '  ' + $reader[1].tostring() + ' '
+ $reader[2].tostring() + ' ' + $reader[3].tostring() + '   ' + $reader[4].tostring() + ', '
+ $reader[5].tostring() + ', ' + $reader[6].tostring() + ' ' + $reader[7].tostring()
}




$reader.close()
$conn.close()

$listOfBadAddresses | out-file '.\Desktop\listOfBadAddresses.txt'
```

Automating PowerShell

- PowerShell can be run straight from Windows Task Scheduler, or can be ran in a .bat file that is ran form the Task Scheduler.



Questions?

Thank you for coming to my presentation!