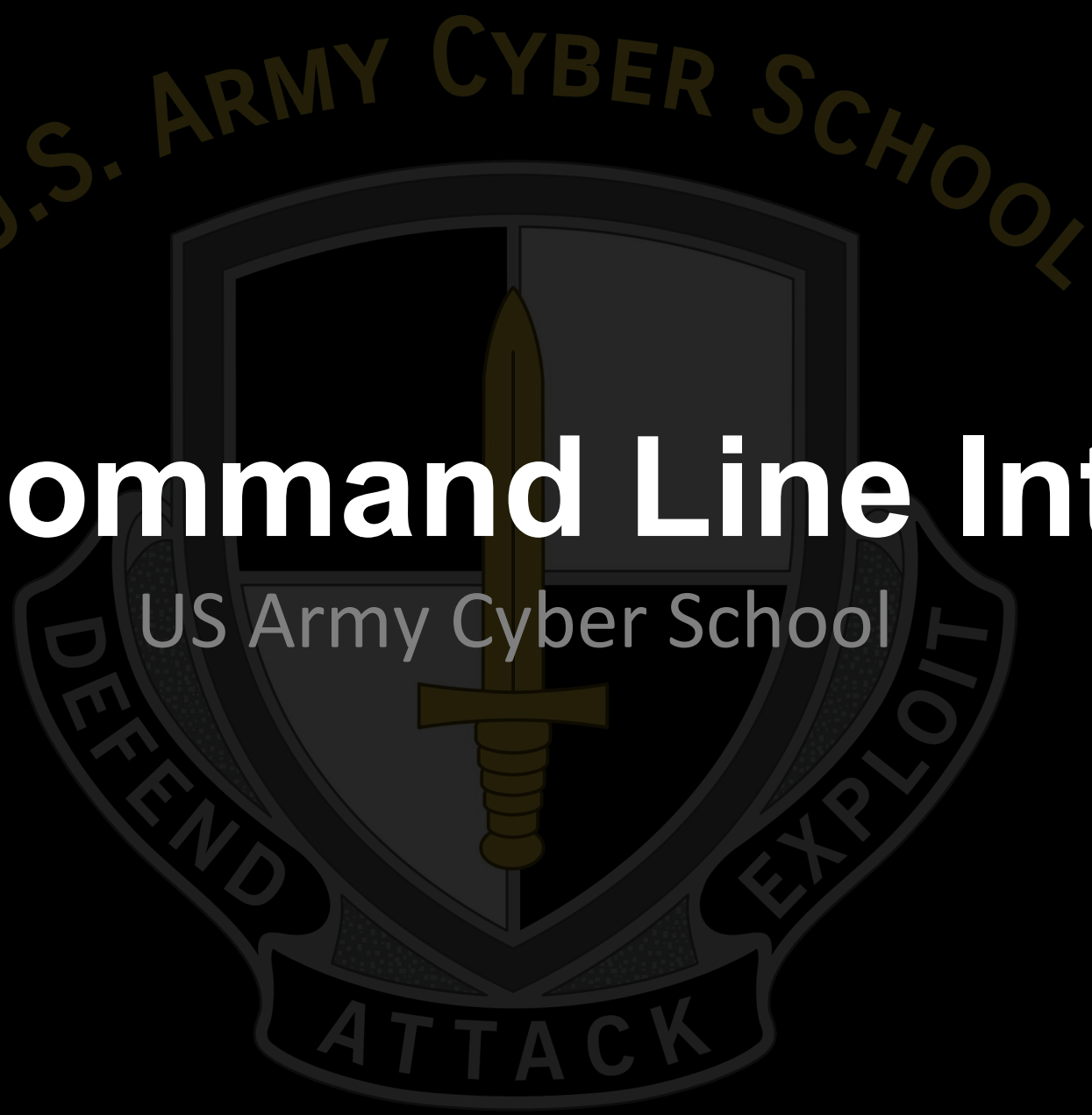



# W01 - Command Line Interface

US Army Cyber School

# CCTC Windows Module Layout

- **CCTC - Windows Module**
  - W01 - Command Line Tools
  - W02 - Processes
  - W03 - Registry
  - W04 - System Hardening / Auditing Logs
  - W05 - Windows Networking
  - W06 - Tactical Survey

# Windows Section 1 - Command Line Tools

- **SKILL CCWE01: Employ commands using COMMAND LINE INTERFACE**
  - CCWE01.01 - Use command line commands to gain situational awareness of the current workstation
  - CCWE01.02 - Use System Internal tools to gain situational awareness of the current workstation

- **SKILL CCWE02: Employ commands using WINDOWS MANAGEMENT INSTRUMENTATION COMMAND line**
  - CCWE02.01 - Use WMIC commands to gain situational awareness of the current workstation

# Windows Section 1 - Command Line Tools

- **SKILL CCWE03: Employ commands using POWERSHELL**
  - CCWE03.01 - Identify the purpose of using PowerShell in operations
  - CCWE03.02 - Demonstrate basic functionality of PowerShell
  - CCWE03.03 - Describe the main components of PowerShell

- **SKILL CCWE04: Develop SCRIPTS**
  - CCWE04.01 - Discuss the purpose of creating a script
  - CCWE04.02 - Create a batch script that will perform a basic enumeration of a workstation
  - CCWE04.03 - Create a Powershell script that will perform basic enumeration of a workstation

# Day 1

# Command Line Interface (CLI)

***Why is it so important to be comfortable using the command line tools and understanding and comprehending the output?***

# Command Line Interface (CLI)

***Why is it so important to be comfortable using the command line tools and understanding and comprehending the output?***

- Understanding the output of command line tools could make or break your operation
- The GUI may not be available in all situations you encounter
- Often times a command line tool must be used
- Facilitates standardization and automation (Repeatable, scriptable)

# CMD.exe Basic Native Commands

| | | |
|---|---|---|
| `set` | - | view all env variables in command shell |
| `where` | - | find executables within the PATH variable |
| `echo` | - | outputs strings passed to it |
| `dir` | - | list folder contents |
| `type` | - | output contents of a file |
| `findstr` | - | windows grep |
| `hostname` | - | system hostname |
| `date /t` | - | output system date (/t keeps it from trying to set) |
| `time /t` | - | output system time (/t keeps it from trying to set) |

- This is a just a SMALL SUBSET of the available commands. These commands can be put together and saved into a Batch (.bat) file and run automatically. This allows for automation of tasks.

# CMD.exe Native Command Controllers

## REDIRECTION OPERATORS

`>` - redirect STDOUT. Create/overwrite

`>>` - redirect STDOUT. Create/append

`|` - Piping sends output of one command to input of another

## CONDITIONAL PROCESSING

`&` and `;` - execute second command regardless of success/failure of first

`&&` - execute second command ONLY if the first is successful

`||` - execute second command ONLY if the first fails

## NESTING COMMANDS

`( )` - nest commands for complex arrangement

# CMD.exe Command Demonstration

## echo - print text, usually to the screen

| | | |
|---|---|---|
| `echo Hello` | - | prints |
| `echo` | - | shows if echo is on or off |
| `echo .` | - | print a blank line |
| `@echo off` | - | the @ suppresses display of the line in a batch file |

## set - view all environment variables in command shell

| | | |
|---|---|---|
| `set` | - | show all environment variables |
| `set A=4` | - | defines a new variable |
| `echo A=%A%` | - | prints the new variable |
| `echo "A=%A%"` | - | spaces are retained, as are quotes |
| `echo %COMPUTERNAME%` | - | built-in environment variable |

# CMD.exe Command Demonstration




**where - find executables within the PATH variable; are commands native to cmd**

```
where
```

```
where dir          - internal to cmd
where where        - built into windows
where pslist       - third party (sysinternals)
```

# CMD.exe Command Demonstration

## dir - show directory contents

| | | |
|---|---|---|
| `dir` | - | current directory |
| `dir .` | - | current directory also |
| `dir ..` | - | parent directory |
| `dir C:\` | - | specific directory (absolute path) |
| `dir /b` | - | /b for bare, just show full filenames |
| `dir /s` | - | /s subdirectories, recursive |
| `dir /a:h` | - | show hidden files |

# CMD.exe Command Demonstration

**type** - print a file, usually to the screen

```
echo Hello > hello.txt
```
- create a file

```
echo There >> hello.txt
```
- append to the file, create if needed

```
type hello.txt
```
- prints two lines

**date / time** - view all environment variables in command shell

```
date /t
```
- /t prevents setting date. MM/DD/YYYY

```
time /t
```
- /t prevents setting time. HH:MM AM/PM

# CMD.exe Command Demonstration

## findstr - find a substring

```
type hello.txt | findstr “There”
```
- search for a pattern

```
type hello.txt | findstr /i “there”
```
- /i ignore case

```
type hello.txt | findstr /r “h.*r”
```
- regular expression (like grep)

```
type hello.txt | findstr lo
```
- piping sends text from left into right

## hostname - show computer name

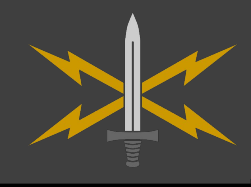
```
hostname
```
- works

```
echo %COMPUTERNAME%
```
- works also

## path - where to find executable programs

```
path
```
- works

```
echo %PATH%
```
- works also



# CMD.exe Command Demonstration



## CONDITIONAL PROCESSING

| | | |
|---|---|---|
| `echo a & echo b & echo c` | - | unconditional separator |
| `dir hello.txt && echo exists` | - | only do second command if first succeeded |
| `dir noway.nohow \|\| echo failed` | - | only do second command if first failed |
| `dir noway.nohow && echo failed` | - | won't print failed, assuming the file doesn't exist |
| `echo %ERRORLEVEL%` | - | zero means success, otherwise failure code |

## NESTING STATEMENTS

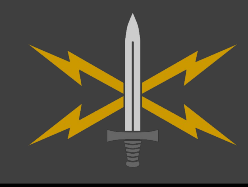
| | | |
|---|---|---|
| `((dir hello.txt && echo success) \|\| echo failure)` | - | success |
| `((dir noway.nohow && echo success) \|\| echo failure)` | - | failure |

# CMD.exe Example



```
((dir desktop && echo “success”) || (echo
“failure”)) & ((dir desktopP && echo
“success”) || (echo “failure”))
```

**What will the above set of commands do when executed in one line?**



# CMD.exe Command Demonstration



## BASIC WINDOWS ACCOUNTS

| | | |
|---|---|---|
| `net` | | |
| `where net` | - | find net |
| `net /?` | - | let's see what windows can do |
| `net help` | - | how to find help |
| `net help user` | - | expanded help on user |
| `net user` | - | list of all users |
| `net user admin` | - | detailed output of admin account / local group membership |
| `net help localgroup` | - | help for localgroup |
| `net localgroup` | - | view all local groups on computer |
| `net localgroup administrators` | - | view members of administrator group |

# CMD.exe Command Demonstration

## MAPPING A REMOTE DRIVE

```
net use t:\\computername\c$ /persistent:no
```
- establish a temporary drive

```
dir t:\users
```
- use temp drive

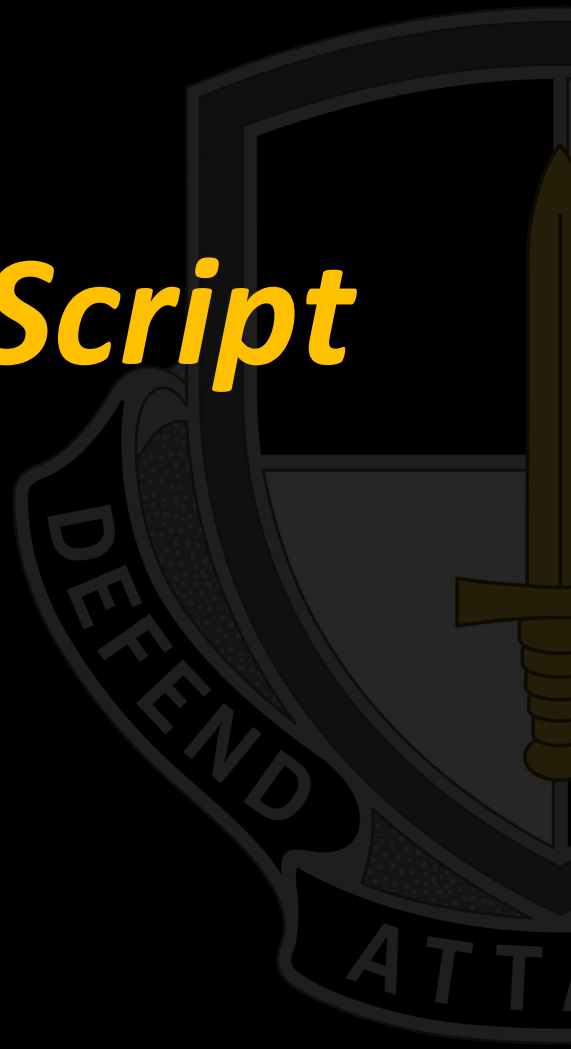
```
net use t: /delete
```
- remove the t: drive label

# ACTIVITY TIME

# *ACTIVITY: Windows Batch Script*

*(Using only CMD commands)*

CLICK ME FOR ACTIVITY PROMPT!



# WMIC Command Basics

## WMIC PROCESSES

| | |
|---|---|
| `wmic /?` | - explain global switches and aliases |
| `wmic process /?` | - shows all the running processes. This is object oriented and can be sorted by headers. |
| `wmic process get /all /format:list` | - all process details in list format |
| `wmic process list brief` | - shows an output similar to tasklist in a legible format. |
| `wmic service list brief` | - shows all the services that are running |

# WMIC Command Basics

## WMIC ACCOUNTS

`wmic useraccount list brief` - shows all the users on the machine

## WMIC NETWORKING

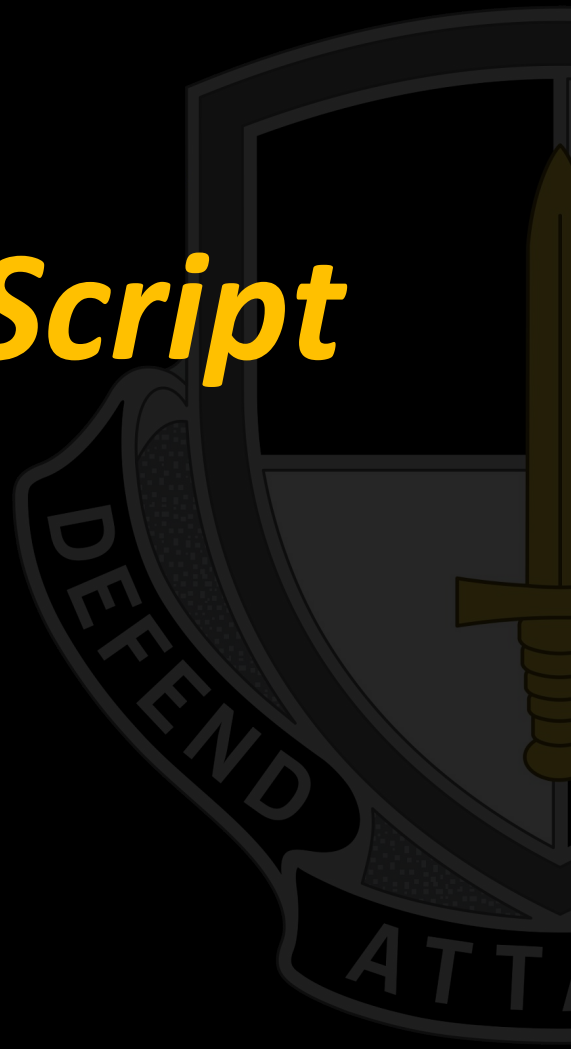
`wmic nicconfig list brief` - NIC information

## WMIC LOGGING AND ACCOUNTING

`wmic nteventlog list brief` - list logs

`wmic ntevent /?` - query individual log entries

# ACTIVITY TIME




## *ACTIVITY: Windows Batch Script*

*(Using only WMIC commands)*

**CLICK ME FOR ACTIVITY PROMPT!**

# Day 2

# Introduction to Powershell



## Powershell Integrated Scripting Environment is available for use

## Different versions of Powershell across windows

- Powershell 1.0 - Nov 2006 (Win XP)
- Powershell 2.0 - Oct 2009 (Win 7)
- Powershell 3.0 - Sep 2012 (Win 8)
- Powershell 4.0 - Oct 2013 (Win 8.1)
- Powershell 5.0 - Apr 2014 (Win 10)

## Powershell uses commandlets (cmdlets)

- Unique to Powershell
- Follow a ‘verb-noun’ pattern : `get-process`

# Introduction to Powershell

## Why use Powershell?

- Much more functionality than cmd
- cmd.exe may be disabled, powershell may not

## Powershell is OBJECT ORIENTED

## `get-help <content>`

- You can search for help and commands using the above syntax

## `get-help <command>`

- You can get help on specific commands using the above syntax

# Introduction to Powershell



External commands (cmd.exe or sysinternals) return a string

Running a Powershell command will return an object

- Demo: `tasklist | get-member`
- Demo: `get-process | get-member`

An object is a data structure that contains properties and methods

- PROPERTIES = data
- METHODS = functions

`tasklist | get-member VS get-process | get-member`

`object.<PropertyName> VS object.<method>(args)`

# Main Components of Powershell

**EXTERNAL COMMANDS - spawn new process (attrib)**

**INTERNAL COMMANDS - runs inside powershell process (ping, dir)**

- cmd.exe is parent of all internal command

**OBJECTS**

- Output from a powershell command

**CLASSES**

- General term for grouped objects

**CIM and WMIC**

- Common Information Model (CIM) - meant to be cross platform
- Windows Management Instrumentation (WMI) - Windows specific

# Main Components of Powershell

## VARIABLES

```
$a="Hello World"
```

## COMMAND SUBSTITUTION

```
$(get-process).name
```

## LOOPING

```
$(get-process).name | foreach-object {echo "$_is a running process"}
```

```
$a=1..5; foreach ($i in $a) {echo "$i is a number"}
```

```
$x=0; while ($x -lt 100) {echo "this is loop number $x"; $x++}
```

## INDEXING - Indices always start at 0

```
$(get-process)[4] --OR-- $(get-process)[0..4]
```

# Main Components of Powershell

## ARITHMETIC

```
1+1 == 2                                1 + “dog” == “1dog”

“cat” + “dog” == “catdog”               $a=“1”; $a+1 == “11”
```

*strings will concatenate, use typecast:* `[int]$a+1 == 2`

## WHILE

```
While ($true) {$date = get-date -format hh:mm; if ($date -eq
“05:00”){break}}
```

## FUNCTIONS

A list of commands chained together to serve a purpose

Once function is declared, issue name of function as command to execute commands in the function.

```
Function dostuff {get-date; get-process; get-service}
```

# Main Components of Powershell

## MULTITHREADING

- A technique that allows a single set of code to be used by several processors at different stages of execution
- To multithread in Powershell, use jobs
  - *https://www.youtube.com/watch?v=4QnJPCqaOWQ*
  - *https://www.youtube.com/watch?v=kj98OhCW-xs*

# Powershell Demonstration

## NAMESPACES

WMI is organized into namespaces, folders that correlate products/technology

```
get-ciminstance -namespace root\securitycenter2 -classname antispywareproduct
```

## FORMAT OBJECT OUTPUT

cmdlet run in PowerShell has default output format

```
get-wmiobject -class win32_BIOS
```

overrie default and format output by piping to format cmdlet

`format-table` , `format-list` , etc

# Powershell Demonstration

```
get-process | get-member
```

Look at associated properties

```
get-process | select threads, processname, id
```

Choose a few properties to view

```
get-process | select threads, processname, id | where {$_.id
-lt 1000}
```

get more granular and view specific process properties

# Powershell Enumeration Scripting



PRO TIP: PowerShell supports tab completion for files, commands, and options

Must set the ExecutionPolicy before running any PowerShell scripts

```
set-ExecutionPolicy Unrestricted -Scope CurrentUser
```

Aliases are available to help all types of users (cmd.exe and bash)

Ex. `get-childitem == ls, dir, gci`

The ForEach (alias %) command has two distinct forms

```
ForEach ($f in Get-ChildItem) {
   Write-output "$($f.Length) $($f.FullName)" }

Get-ChildItem | ForEach {
   Write-output "$($_.Length) $($_.FullName)" }
```

# Powershell Enumeration Scripting



The Where command can be used to filter (e.g. files over 1,000 bytes)

```
get-ChildItem | Where { $_.Length -gt 1000 } | Select Length,
Name
```

Output can be controlled with Format-Table or Format-List

```
get-ChildItem | Format-Table -AutoSize
get-ChildItem | Format-List | more
```

Standard options are available

```
Remove-Item does_not_exist.txt
Remove-Item does_not_exist.txt -ErrorAction SilentlyContinue
New-Item -Type File it_exists.txt
Remove-Item it_exists.txt -Verbose
```



UNCLASSIFIED // FOUO

# Powershell Functions

**Functions in Powershell have unexpected quirks**

```
function summer($a,$b) {
   $total = $a + $b
   echo “a=$a b=$b sum=$total”
   return $total
}

$sum = summer 9 10
Write-Output “Sum is $sum”

summer 1 2
summer 5, 6
summer(7, 8)
summer(7, 8) (9,10)
```

# *EXERCISE: Through the Wire*

**CLICK ME FOR ACTIVITY PROMPT!**

**Blackboard -> Windows Section 1: Command Line Tools -> Exercise: Through the Wire**

# Day 3

# System Enumeration Batch Script



**Demonstrate a batch script that will perform basic enumeration of a windows workstation**

**PRO TIP: Variable (i.e. %i) must be doubled in a batch file (%%i)**



UNCLASSIFIED // FOUO



# Dir hidden files updated in current month



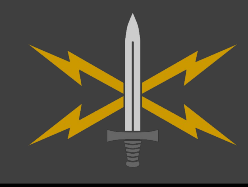
```
@ECHO OFF
SETLOCAL

REM Find all hidden files touched since the first of the current month

REM Make sure they provided a directory name. Else quit.
IF "%~1" == "" ECHO Usage: %~nx0 directory && GOTO :EOF
SET dir=%~1
SET dirout=%TEMP%\%~n0.temp

REM Runs the DATE /T command, and puts the second token in %i and the fourth token in %j
FOR /F "tokens=2,4 delims=/ " %%i in ('DATE /T') DO SET datevar=%%i/01/%%j

ECHO Starting %~nx0 at %DATE% %TIME%

REM Collect all the files changed since the first of the month, whether hidden or not
FORFILES /P "%dir%" /S /D +%datevar% /C "CMD /C IF @isdir=="FALSE" ECHO @path" > "%dirout%"
2>%TEMP%\delete.me

SET CNT=0
REM Reads the output from FORFILES and looks for hidden files
FOR /F "delims=" %%i in ( %dirout% ) do (
  DIR /A:H %%i > %TEMP%\delete.me 2>&1
  REM Returns an error if the file is not hidden, Success means the file is hidden
  IF NOT ERRORLEVEL 1 (
    REM Prints the attributes and the file name (A=Archived, S=System, H=Hidden, I=Not Indexed)
    ATTRIB %%i
    SET /A CNT=CNT+1
  )
)

ECHO Finished %~nx0 at %DATE% %TIME%. Count = %CNT%
```



# Dir hidden files updated in current month



```
Starting Hidden.bat at Wed 01/01/2017 15:46:32.94
A   H   I    C:\users\fred\NTUSER.DAT
A   H        C:\users\fred\AppData\Local\IconCache.db
A  SH   I    C:\users\fred\AppData\Local\Microsoft\Credentials\68A8F6097AFFD807BE8B31CAB79E2CF7
A  SH   I    C:\users\fred\AppData\Local\Microsoft\Credentials\7A5C2BCD963C43DD1CABCCC685110AA9
A   H        C:\users\fred\AppData\Local\Microsoft\Windows\UsrClass.dat
A  SH   I    C:\users\fred\AppData\Local\Microsoft\Windows\History\History.IE5\MSHist012017073120170807\container.dat
A  SH   I    C:\users\fred\AppData\Local\Microsoft\Windows\History\History.IE5\MSHist012017080720170808\container.dat
A  SH   I    C:\users\fred\AppData\Local\Microsoft\Windows\History\History.IE5\MSHist012017080820170809\container.dat
A  SH   I    C:\users\fred\AppData\Local\Microsoft\Windows\History\History.IE5\MSHist012017080920170810\container.dat
A   H        C:\users\fred\AppData\Local\Microsoft\Windows\Notifications\WPNPRMRY.tmp
A  SH   I
C:\users\fred\AppData\Local\Packages\Microsoft.Windows.Cortana_cw5n1h2txyewy\AC\AppCache\63Z2SVEB\46\container.dat
A  SH   I    C:\users\fred\AppData\Local\Packages\Microsoft.WindowsMaps_8wekyb3d8bbwe\AC\INetCache\container.dat
A  SH   I    C:\users\fred\AppData\Local\Packages\Microsoft.WindowsMaps_8wekyb3d8bbwe\AC\INetCookies\container.dat
    H        C:\users\fred\AppData\Roaming\Microsoft\Office\Recent\index.dat
A  SH C:\users\fred\AppData\Roaming\Microsoft\Protect\S-1-5-21-1584042266-1357540265-3178510380-1001\39359ce9-b42b-46f7-b894-f54f8ca23cd0
A  SH C:\users\fred\AppData\Roaming\Microsoft\Protect\S-1-5-21-1584042266-1357540265-3178510380-1001\fb38b817-be0b-4898-9437-8eea7329c2ce
A  SH        C:\users\fred\IntelGraphicsProfiles\Brighten Video.man.igpi
A  SH        C:\users\fred\IntelGraphicsProfiles\Darken Video.man.igpi
A  SH        C:\users\fred\IntelGraphicsProfiles\Enhance Video Colors.man.igpi
Finished Hidden.bat at Wed 01/01/2017 15:48:17.53. Count = 19
```



# PowerShell hidden files updated in 7 days



```
[CmdletBinding()]
Param(
    [Parameter(Mandatory=$True,Position=1)] [String]$Dir
)

Get-ChildItem -Recurse $Dir -Force -ErrorAction
SilentlyContinue |
  Where-Object { ($_.mode -match "h") -and
    ($_.LastWriteTime -gt (Get-Date).AddDays(-7))
  }
```

# PowerShell hidden files updated in 7 days

WE04.02

```
@echo off       #disable echo
Rem             #remark
echo %1 %2 %3   #command line arguments
set foo=bar     #set a variable. avoid quotes here
set x=4         #use 'set /a' for numeric
set /a y=x+1    #now y is 5
echo %x% %y%    #prints '4 5'
echo %foo% + %bar%

set foo="This is some magic foo"    #should never use quotes on a 'set'
set bar=This is some magic bar      #all spaces are kept, including trailing spaces
echo %bar%
echo %foo:~1,-1%    #use variable manipulation to remove the leading and trailing quotes

setlocal        #use local context for variables
endlocal        #end local context (automatic at end of a .bat file)

if %foo% == 4 (echo yep) else (echo nope)     #can be multi-line: '(' at end of line, ')' at beginning
if exist c:\tmp\file.txt echo There it is     #if-exist
if errorlevel 1 ...                           #if previous command failed with %ERRORLEVEL% >= 1

for /?
for %i in ( 1 3 7 ) do @echo %i                 #iterate through a list of values
for /l %n in (0,1,5) do @(echo Round %n)        #for loop (start,increment,stop)
for /f $a in ( "file" ) do ...                  #read values from a file (for /?)

Be aware that %i, etc all must be doubled inside a .bat file. I.e.,

for %%i in ( 1 3 7 ) do @echo %%i               #iterate through a list of values
```

# ACTIVITY TIME

## *ACTIVITY: DLL Enumeration*

CLICK ME FOR ACTIVITY PROMPT!

# SysInternals Tools

`<tool name> /?` - provides help menu for each tool

## PROCESSES

| | | |
|---|---|---|
| `psinfo` | - | shows basic system info, remote capabilities |
| `psinfo -h -s -d -nobanner` | | |
| `pslist` | - | shows processes in tree format |
| `procmon` | - | view, monitor, filter processes (GUI based) |
| `autoruns` | - | checks autorun registry locations |
| `handle` | - | shows handles of all processes |
| `handle -p <process name>` | - | specific process (matches partial names) |

# SysInternals Tools

`<tool name> /?` - provides help menu for each tool

## USERS

| | | |
|---|---|---|
| `logonsessions` | - | lists all currently logged in sessions |
| `logonsessions -p` | - | lists processes running in each logon sessions |
| `psloggedon` | - | more functionality regarding remote users sessions |

## NETWORKING

| | | |
|---|---|---|
| `tcpview` | - | robust netstat viewer/monitor |

# ACTIVITY TIME


## *RESEARCH ACTIVITY:*

## *Sysinternals suite*

`psinfo` `pslist` `handle`

`logonsessions` `psloggedon` `tcpview`

# ACTIVITY TIME

## *ACTIVITY: Groups by User*

CLICK ME FOR ACTIVITY PROMPT!

# ACTIVITY TIME

## *ACTIVITY: Rootkit Hunter*

CLICK ME FOR ACTIVITY PROMPT!