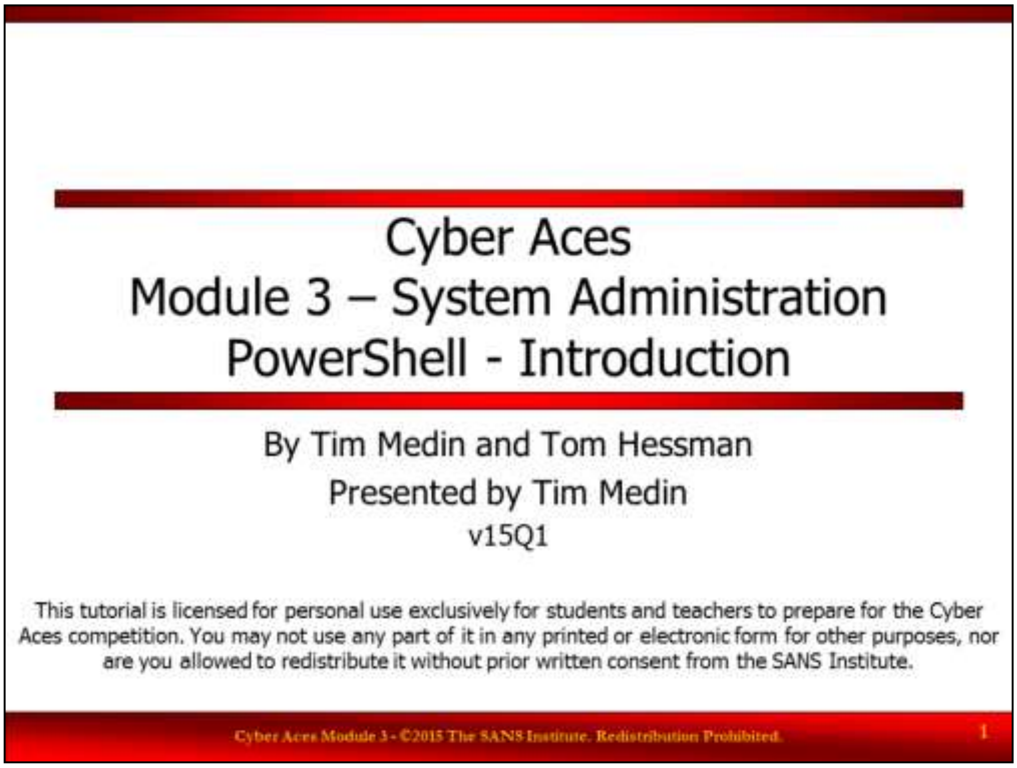


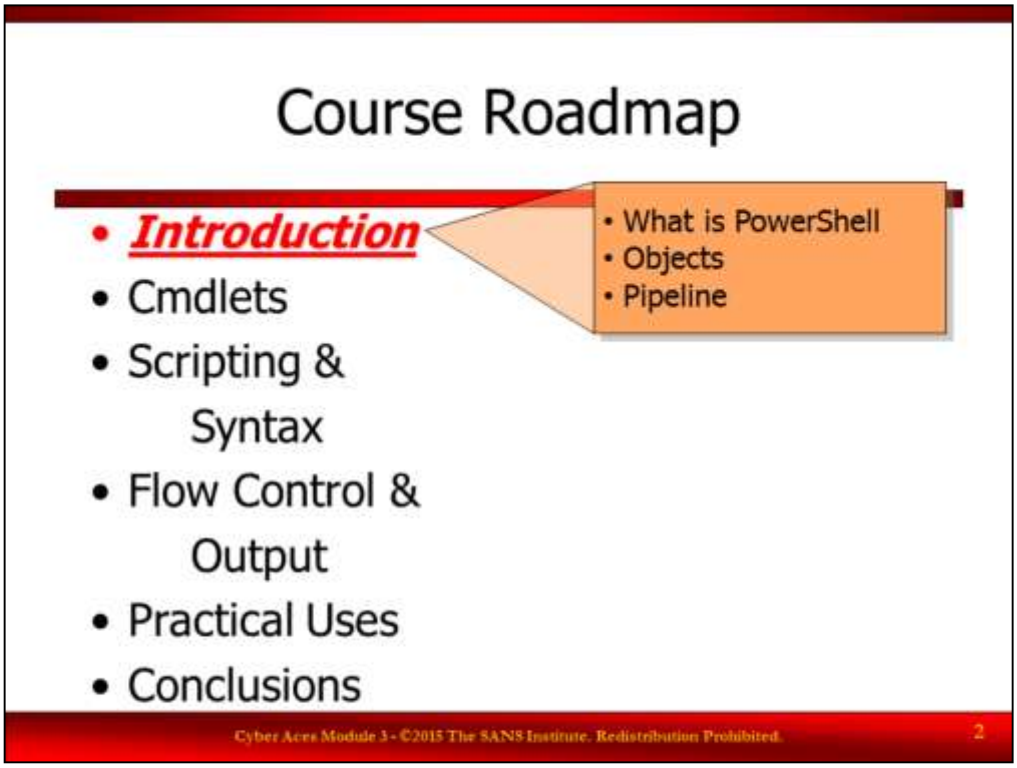
Cyber Aces Module 3 – System Administration PowerShell - Introduction

By Tim Medin and Tom Hessman

Presented by Tim Medin

v15Q1

Welcome to Cyber Aces, Module 3! This module provides an introduction to the latest shell for Windows, PowerShell.



Course Roadmap

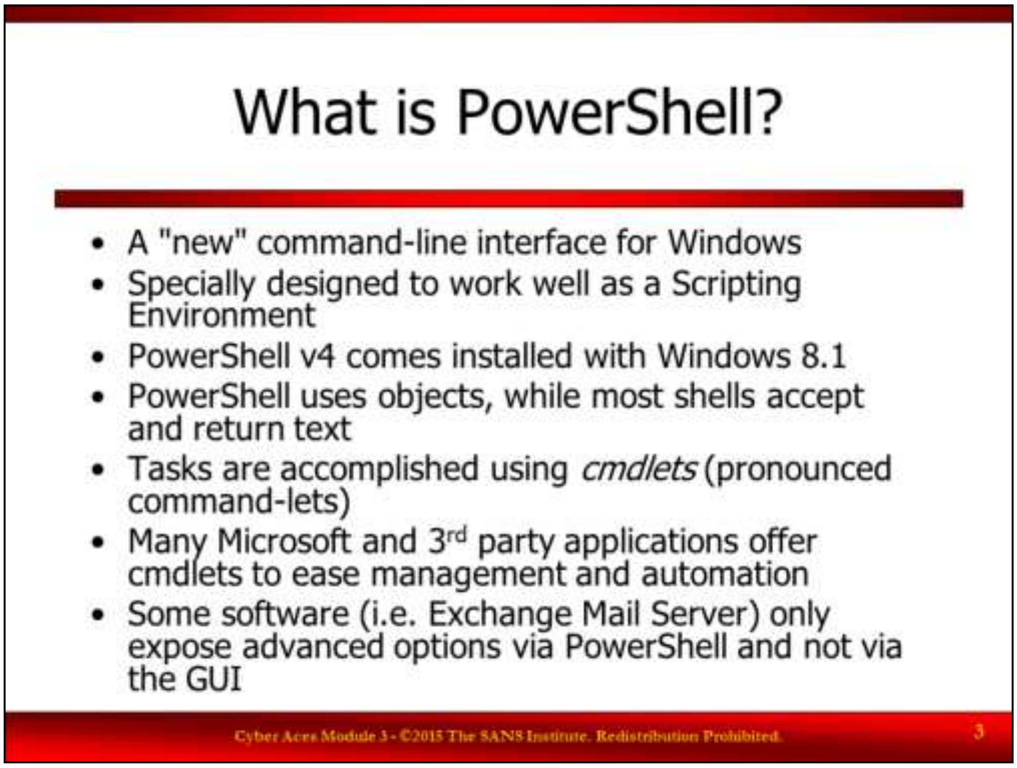
- ***Introduction***
- Cmdlets
- Scripting & Syntax
- Flow Control & Output
- Practical Uses
- Conclusions

• What is PowerShell
• Objects
• Pipeline

2

Course Roadmap

Is this section, you will be introduced to PowerShell and some basic syntax.



What is PowerShell?

- A "new" command-line interface for Windows
- Specially designed to work well as a Scripting Environment
- PowerShell v4 comes installed with Windows 8.1
- PowerShell uses objects, while most shells accept and return text
- Tasks are accomplished using *cmdlets* (pronounced command-lets)
- Many Microsoft and 3rd party applications offer cmdlets to ease management and automation
- Some software (i.e. Exchange Mail Server) only expose advanced options via PowerShell and not via the GUI

3

What is PowerShell

Originally codenamed Monad (or Microsoft Shell or MSH), it was designed as a new approach to managing Windows systems via the command line. PowerShell was originally a separate download for Windows XP, Vista, Windows Server 2003, and later for Window Server 2008 (R1), but it is not supported on Windows 2000.

PowerShell version 2.0 was integrated into Windows 7 and was released at the same time as Windows 7. Windows Server 2008R2 also came with PowerShell v2.0 installed. Separate installs were made available for previous versions of Windows. The Windows 8 family includes PowerShell v3.0. Windows 8.1 comes installed with PowerShell 4.0.

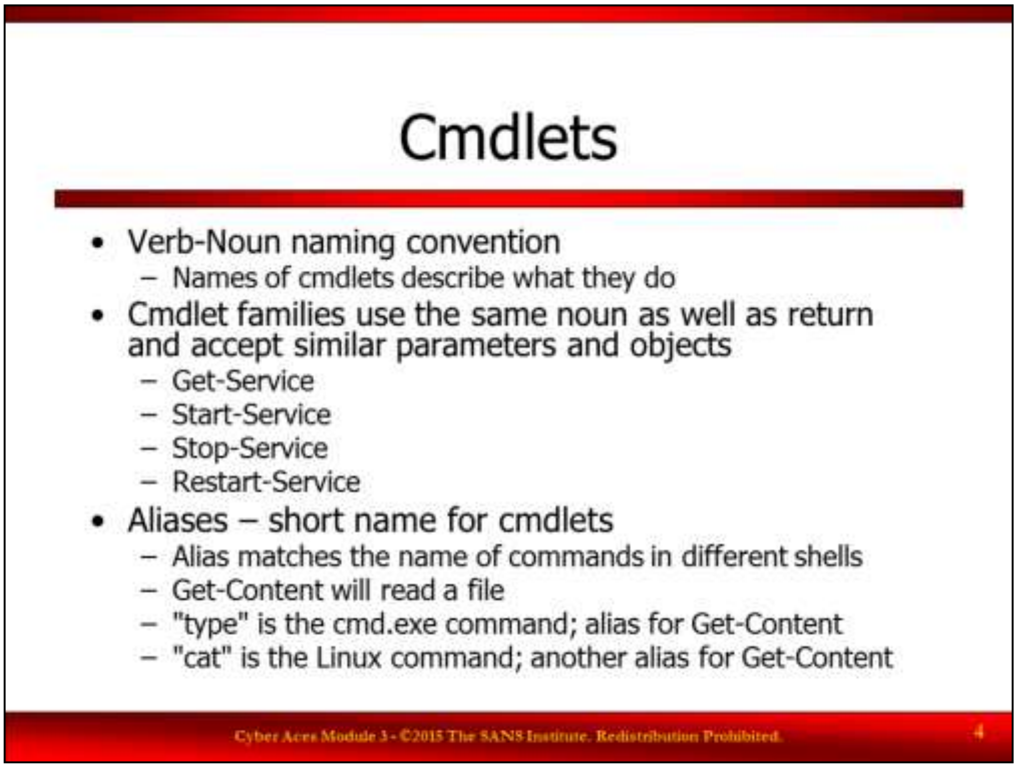
Prior to PowerShell, all major shells used text as input and output. As we'll see, the use of objects allows structured data to be used as input and output which allows for simpler manipulation of data via the command line.

PowerShell uses cmdlets (pronounced command-lets) to accomplish tasks and are very similar to *commands* used by other shells and operating systems. The cmdlets often expose more options than are available via the GUI (Graphical User Interface) and are generally the recommended approach for adjusting advanced features of many server packages that support PowerShell.

Read more about PowerShell:

http://technet.microsoft.com/en-us/scriptcenter/powershell.aspx

https://en.wikipedia.org/wiki/Windows_PowerShell



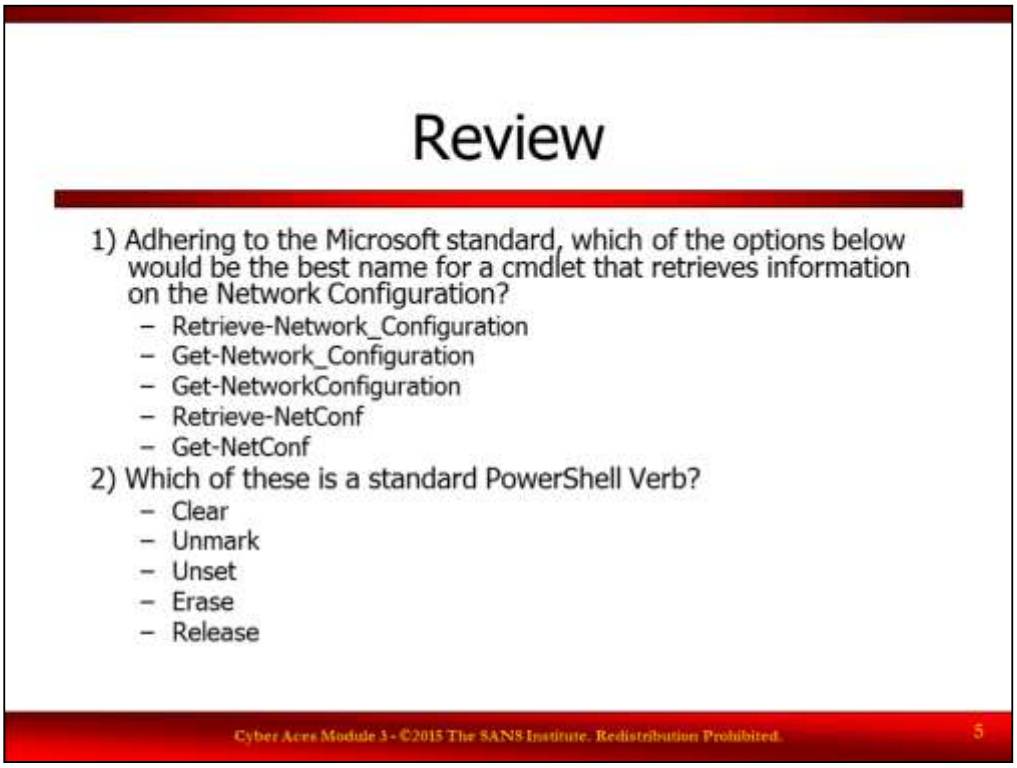
Cmdlets

- Verb-Noun naming convention
  - Names of cmdlets describe what they do
- Cmdlet families use the same noun as well as return and accept similar parameters and objects
  - Get-Service
  - Start-Service
  - Stop-Service
  - Restart-Service
- Aliases – short name for cmdlets
  - Alias matches the name of commands in different shells
  - Get-Content will read a file
  - "type" is the cmd.exe command; alias for Get-Content
  - "cat" is the Linux command; another alias for Get-Content

Cmdlets

PowerShell uses a verb-noun pair for cmdlet names. For example, Get-Date would "get" the current "date." The verbs are standardized by Microsoft (http://msdn.microsoft.com/en-us/library/ms714428(v=vs.85).aspx) to make memorization easier and to ensure consistent use of names. This standard ensures that cmdlet developers all use the verb "Add", instead of a seemingly random assortment of "Append", "Attach", "Concatenate", or "Insert".

Families of commands are grouped by nouns. It is quickly apparent that "Get-Service", "Start-Service", "Stop-Service", and "Restart-Service" are all related. As such, these cmdlets accept a similar set of parameters and return a similar set of objects.



Review

1) Adhering to the Microsoft standard, which of the options below would be the best name for a cmdlet that retrieves information on the Network Configuration?
   - Retrieve-Network_Configuration
   - Get-Network_Configuration
   - Get-NetworkConfiguration
   - Retrieve-NetConf
   - Get-NetConf
2) Which of these is a standard PowerShell Verb?
   - Clear
   - Unmark
   - Unset
   - Erase
   - Release

5

Review

1) Adhering to the Microsoft standard, which of the options below would be the best name for a cmdlet that retrieves information on the Network Configuration?

    Retrieve-Network_Configuration

    Get-Network_Configuration

    Get-NetworkConfiguration

    Retrieve-NetConf

    Get-NetConf

2) Which of these is a standard PowerShell Verb?

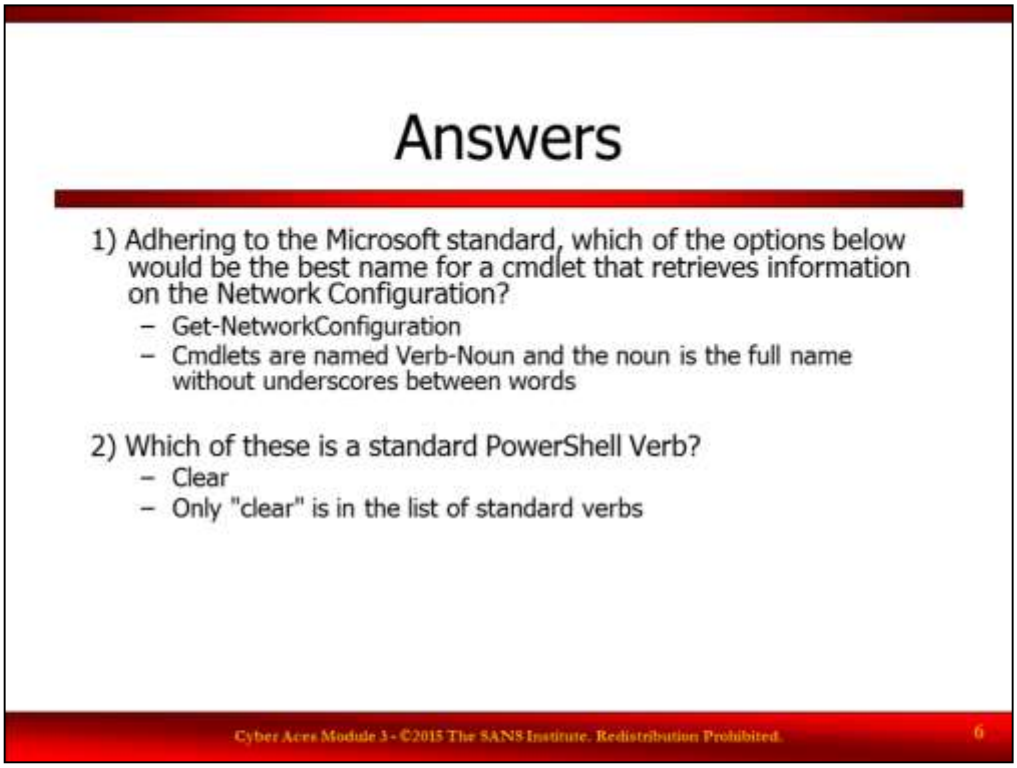
    Clear

    Unmark

    Unset

    Erase

    Release



Answers

1) Adhering to the Microsoft standard, which of the options below would be the best name for a cmdlet that retrieves information on the Network Configuration?
   - Get-NetworkConfiguration
   - Cmdlets are named Verb-Noun and the noun is the full name without underscores between words

2) Which of these is a standard PowerShell Verb?
   - Clear
   - Only "clear" is in the list of standard verbs

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited. 6

Answers

1) Adhering to the Microsoft standard, which of the options below would be the best name for a cmdlet that retrieves information on the Network Configuration?

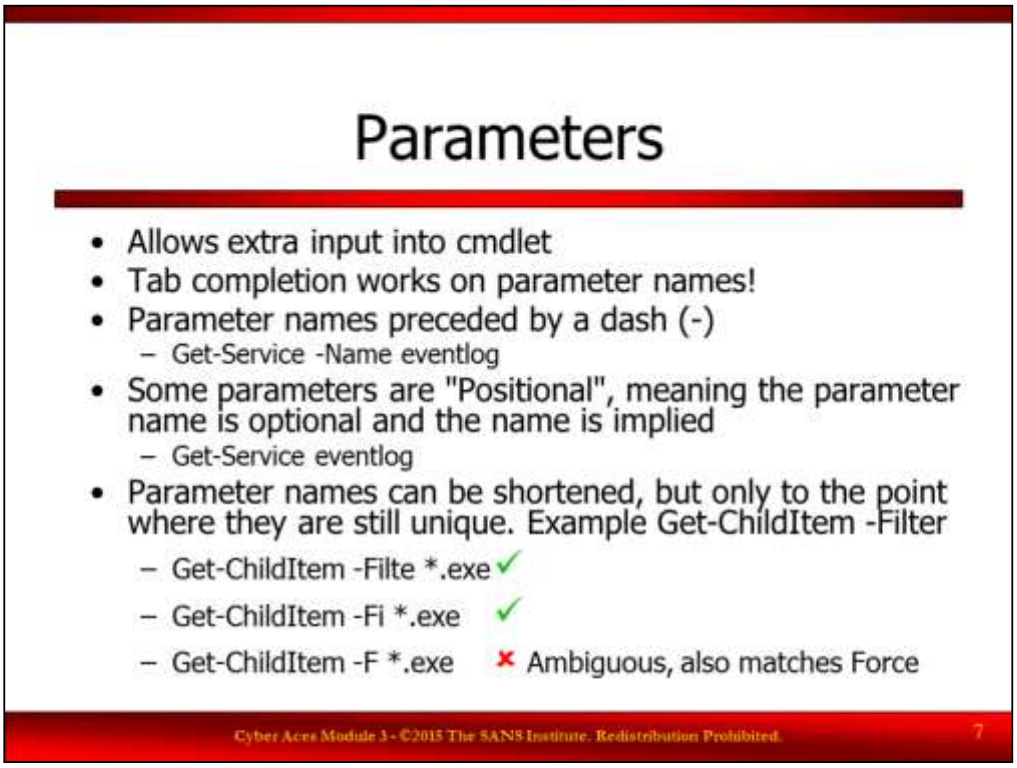
   Get-NetworkConfiguration

   Cmdlets are named Verb-Noun and the noun is the full name without underscores between words

2) Which of these is a standard PowerShell Verb?

   Clear

   Only "clear" is in the list of standard verbs

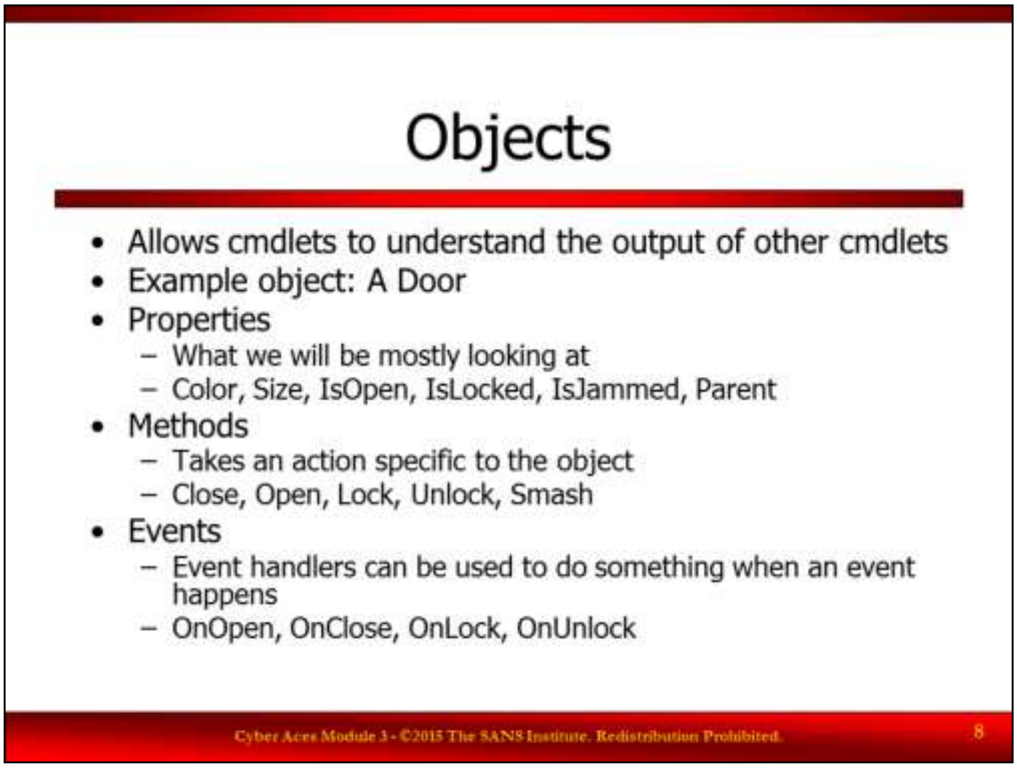

Parameters

- Allows extra input into cmdlet
- Tab completion works on parameter names!
- Parameter names preceded by a dash (-)
  - Get-Service -Name eventlog
- Some parameters are "Positional", meaning the parameter name is optional and the name is implied
  - Get-Service eventlog
- Parameter names can be shortened, but only to the point where they are still unique. Example Get-ChildItem -Filter
  - Get-ChildItem -Filte *.exe ✓
  - Get-ChildItem -Fi *.exe ✓
  - Get-ChildItem -F *.exe ✗ Ambiguous, also matches Force

7

Parameters

Most cmdlets take additional parameters (or arguments). Parameter names are preceded by a dash (-). One such parameter, used by the cmdlet "Get-Service", is "Name". The command "Get-Process -Name svchost" will "get" the objects representing each "process" with the "name" "svchost". Some cmdlets accept positional parameters, meaning a parameter name is not required since it is assumed from its position on the command line. "Get-Service's" "Name" is such a parameter, and the above command can be shortened to "Get-Process svchost". It is pretty convenient that cmdlets can be shortened and some parameter names can be dropped.



Objects

- Allows cmdlets to understand the output of other cmdlets
- Example object: A Door
- Properties
  - What we will be mostly looking at
  - Color, Size, IsOpen, IsLocked, IsJammed, Parent
- Methods
  - Takes an action specific to the object
  - Close, Open, Lock, Unlock, Smash
- Events
  - Event handlers can be used to do something when an event happens
  - OnOpen, OnClose, OnLock, OnUnlock

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited. 8

Objects

In Bash and cmd.exe scripting, you often spend a great deal of time interfacing between applications. In other words, you capture the output of one command, parse out the pieces of data that you need (such as an IP address), and then pass that information on to the next command. Wouldn't it be nice if each command automatically understood the output of the other? Besides being easier to read, it is much easier to write.

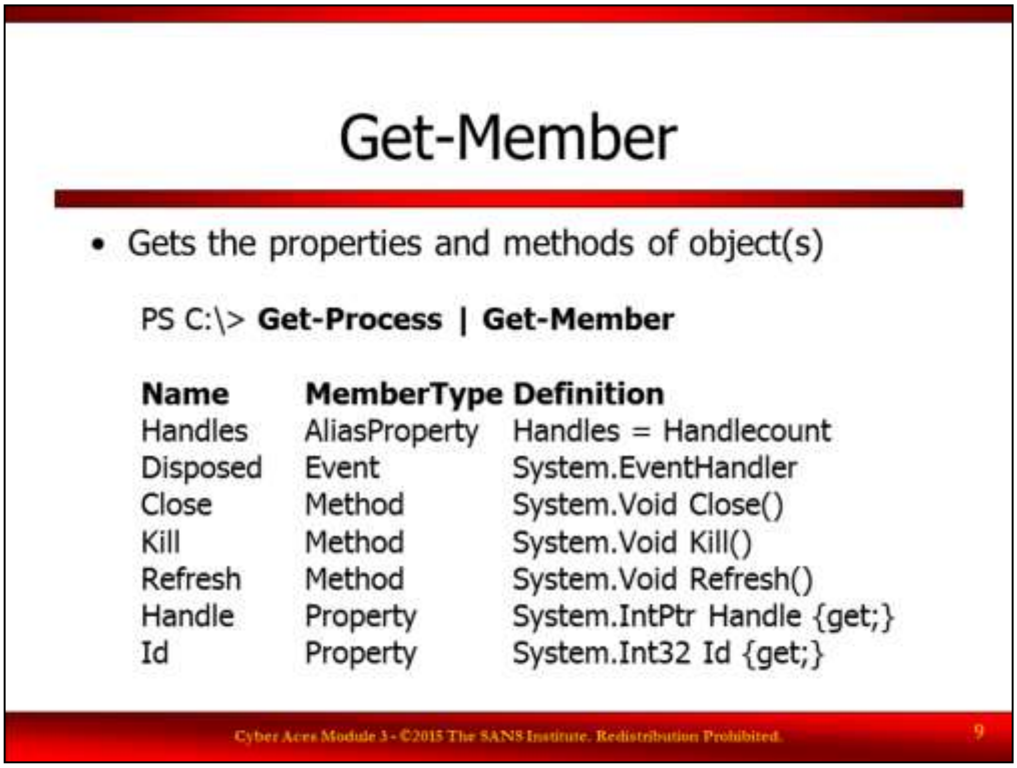
Well, as you probably guessed, that is one of the benefits of the object-oriented nature of PowerShell. In PowerShell, every "cmdlet" has an understanding of the output from other cmdlets, and they can be tied together with powerful results. The objects returned from each cmdlet are understood by other cmdlets. A simple glance at the command reveals which property is being used, and it doesn't require extra effort or intimate knowledge of the output in "field 2."

For example, it isn't immediately clear what is being done in the command below (it gets a list of the process ID's for each running process).

```
$ ps aux | cut -d' ' -f2
```

While the equivalent PowerShell command is much more readable.

```
PS C:\> Get-Process | Select ID
```



Get-Member

- Gets the properties and methods of object(s)

```
PS C:\> Get-Process | Get-Member

Name      MemberType    Definition
Handles   AliasProperty Handles = Handlecount
Disposed  Event         System.EventHandler
Close     Method        System.Void Close()
Kill      Method        System.Void Kill()
Refresh   Method        System.Void Refresh()
Handle    Property      System.IntPtr Handle {get;}
Id        Property      System.Int32 Id {get;}
```

Get-Member

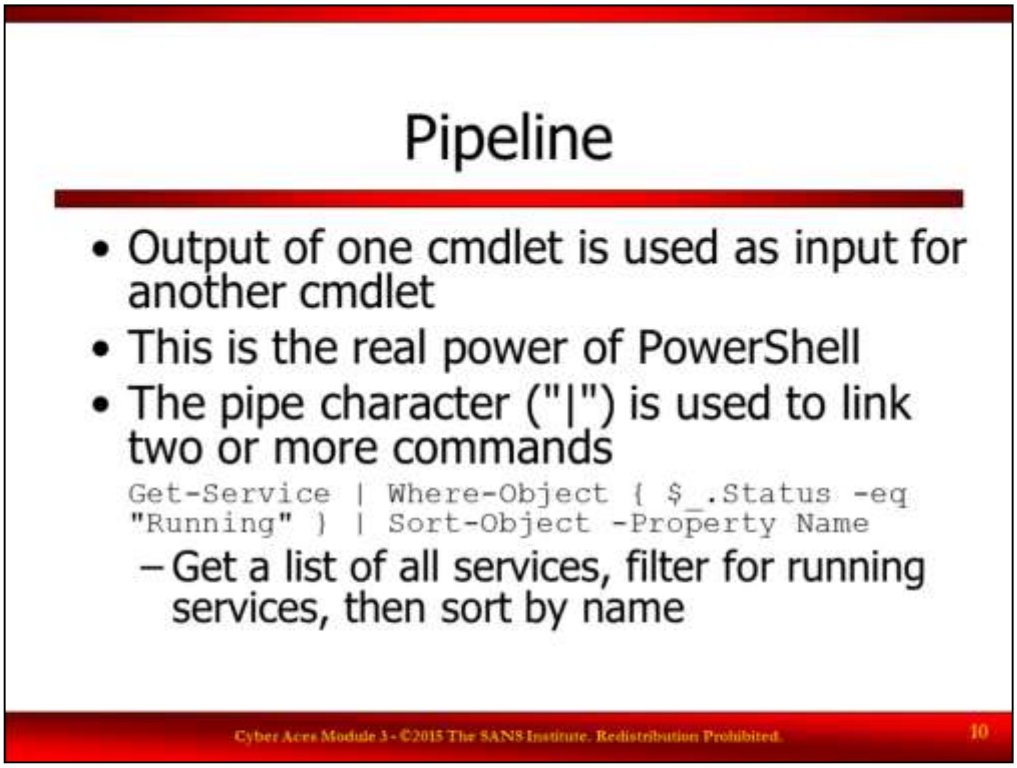
A new object type (or set of objects) may be encountered for the first time and you may not know what properties and methods are available to interact with the object. How do we know which properties and methods are available? The cmdlet "Get-Member" can be used to show available properties, methods, and events as shown above (the output has been modified for brevity).

We can kill a process by calling the Kill method.

```
PS C:\> $a = Get-Process spoolsv
PS C:\> $a.Kill()
```

Or more tersely:

```
PS C:\> (Get-Process spoolsv).Kill()
```



Pipeline

- Output of one cmdlet is used as input for another cmdlet
- This is the real power of PowerShell
- The pipe character ("|") is used to link two or more commands

  ```
  Get-Service | Where-Object { $_.Status -eq
  "Running" } | Sort-Object -Property Name
  ```

  - Get a list of all services, filter for running services, then sort by name

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited. 10

Pipeline

The real "power" in PowerShell is using the objects with the pipeline. This pipeline takes the output objects from one command and sends it as input to the next command. Simply use the pipe character ("|") to link our two commands. Here is a real-world example of the use of the pipeline:

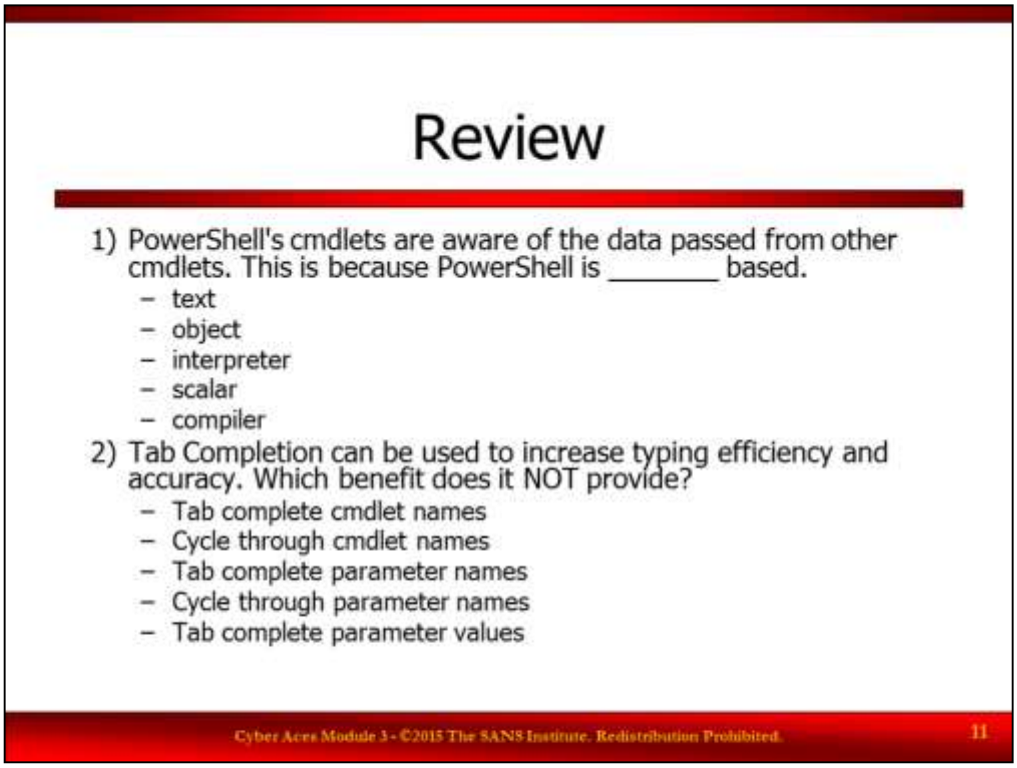
```
PS C:\> Get-Service | Where-Object { $_.Status -eq "Running" } | Sort-
        Object -Property Name
```

This command will return all the services, filter for the running ones, and sort them by name. Don't worry about the syntax of "Where-Object" for now, that will be covered in a bit.

While this is highly dangerous, we could even use this same syntax to stop all running services (don't try this at home!):

```
PS C:\> Get-Service | Where-Object { $_.Status -eq "Running" } | Stop-
        Service
```

All sorts of commands can be chained together to create some really powerful and flexible commands.



Review

1) PowerShell's cmdlets are aware of the data passed from other cmdlets. This is because PowerShell is _______ based.
   - text
   - object
   - interpreter
   - scalar
   - compiler
2) Tab Completion can be used to increase typing efficiency and accuracy. Which benefit does it NOT provide?
   - Tab complete cmdlet names
   - Cycle through cmdlet names
   - Tab complete parameter names
   - Cycle through parameter names
   - Tab complete parameter values

Review

1) PowerShell's cmdlets are aware of the data passed from other cmdlets. This is because PowerShell is _______ based.

   text

   object

   interpreter

   scalar

   compiler

2) Tab Completion can be used to increase typing efficiency and accuracy. Which benefit does it NOT provide?

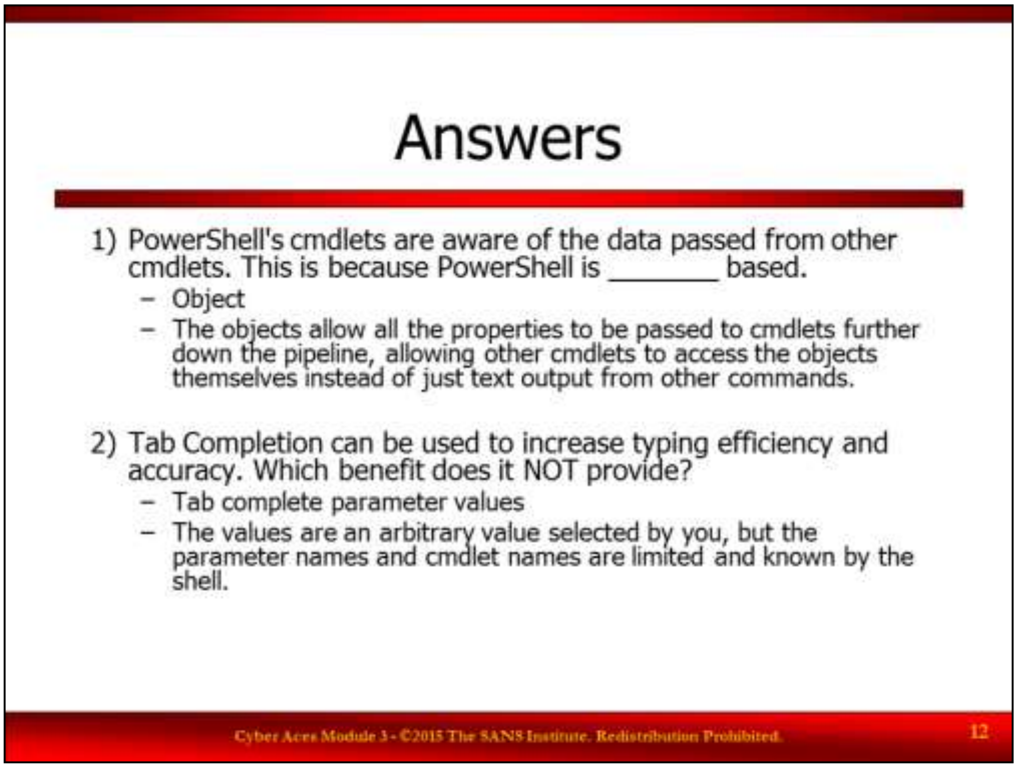
   Tab complete cmdlet names

   Cycle through cmdlet names

   Tab complete parameter names

   Cycle through parameter names

   Tab complete parameter values



Answers

1) PowerShell's cmdlets are aware of the data passed from other cmdlets. This is because PowerShell is _______ based.
   - Object
   - The objects allow all the properties to be passed to cmdlets further down the pipeline, allowing other cmdlets to access the objects themselves instead of just text output from other commands.

2) Tab Completion can be used to increase typing efficiency and accuracy. Which benefit does it NOT provide?
   - Tab complete parameter values
   - The values are an arbitrary value selected by you, but the parameter names and cmdlet names are limited and known by the shell.

12

Answers

1) PowerShell's cmdlets are aware of the data passed from other cmdlets. This is because PowerShell is _______ based.

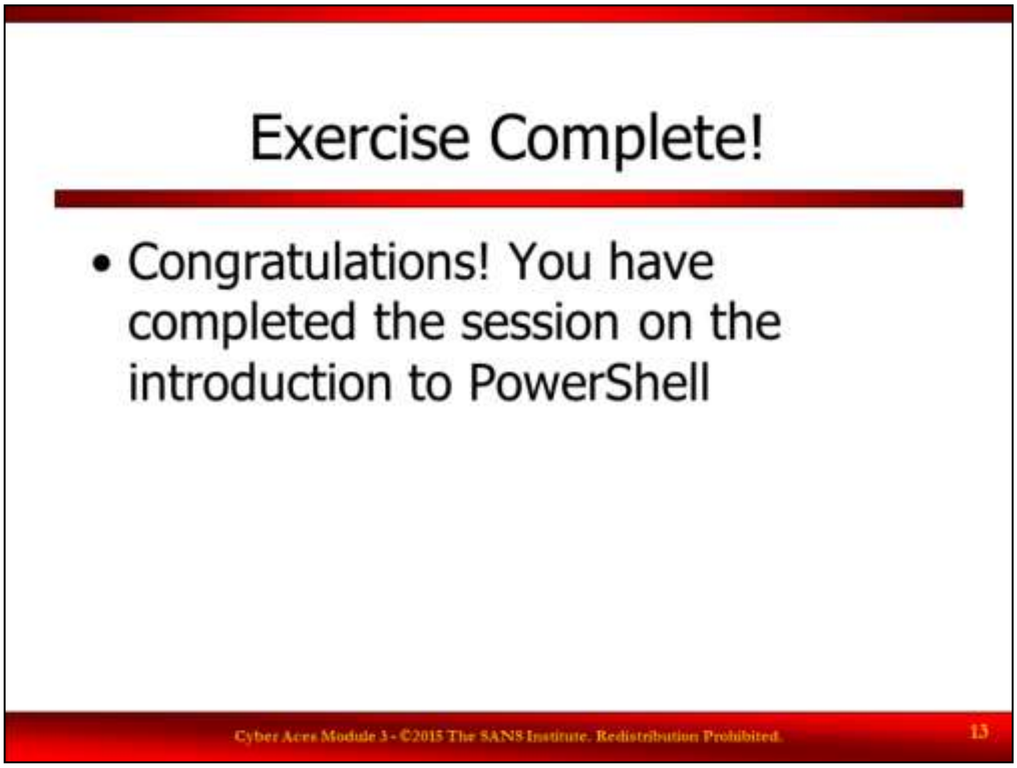
   Object

   The objects allow all the properties to be passed to cmdlets further down the pipeline, allowing other cmdlets to access the objects themselves instead of just text output from other commands.

2) Tab Completion can be used to increase typing efficiency and accuracy. Which benefit does it NOT provide?

   Tab complete parameter values

   The values are an arbitrary value selected by you, but the parameter names and cmdlet names are limited and known by the shell.



Exercise Complete!

- Congratulations! You have completed the session on the introduction to PowerShell

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited. 13

Exercise Complete