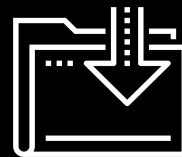




# PowerShell Scripting

Cybersecurity  
Windows Administration and Hardening Day  
2



# Class Objectives

---

By the end of today's class, you will be able to:



Use basic PowerShell cmdlets to navigate Windows and manage directories and files.



Use PowerShell pipelines to retrieve Windows system event logs.



Combine various shell-scripting concepts such as cmdlets, parameters, piping, conditions, and importing files with data structures.



# Recap

---

Last class, we learned how to use CMD to execute many Windows sysadmin tasks:



How to audit processes with Task Manager.



Using CMD to create files.



Creating a report with `wmic` in the command line.



Auditing unwanted startup applications and services.



Enumerating local users, groups, and current local password policies.



Creating new regular and administrative users and setting local password policies.

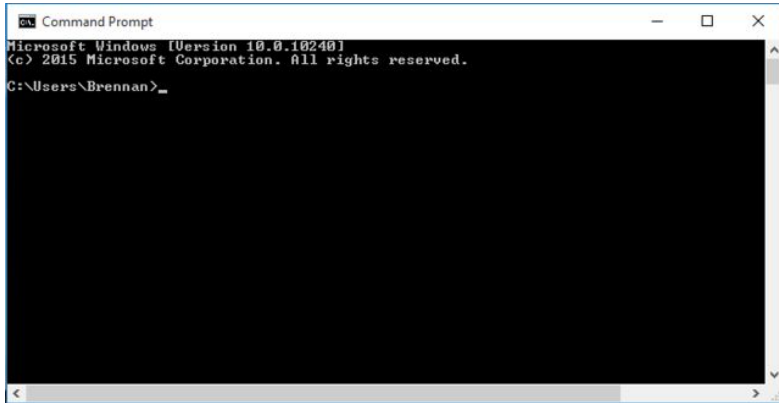


Scheduling tasks using Task Scheduler.

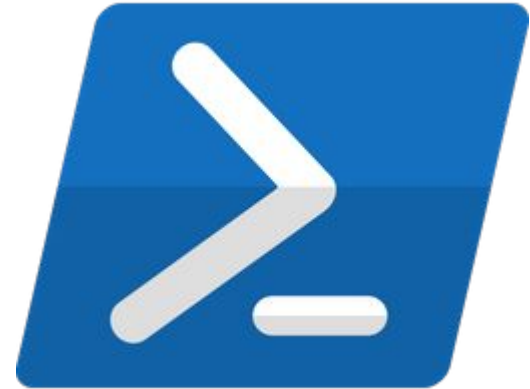
# PowerShell

---

While CMD is short, simple, and easy to learn, it isn't designed for complex operations and procedures.



On the other hand, PowerShell was designed as a powerful language used to execute, automate, and customize the most demanding and difficult tasks.



# PowerShell in System Administration

---

PowerShell can be used to manage everything in a Microsoft enterprise environment, including:



**Windows Server:** Microsoft's central server for domain and networking services.



**Windows 10:** Microsoft's personal and professional computer operating system. The operating system used by most people and within many large enterprise environments.



**Office365:** Microsoft's cloud services, offering a wide variety of cloud services, such as virtual private clouds and cloud-based VMs.



**Azure:** Microsoft's cloud services, offering a wide variety of cloud services, such as virtual private clouds and cloud-based VMs.

# PowerShell in Cybersecurity

---

## Defensive security:



PowerShell can be used to manage and audit logs.



There are many commands for interacting with Windows Event logs. We will be looking at these later.



PowerShell can also be used to harden the security on Windows hosts and servers.



There are many modules and scripts that extend PowerShell's powerful functionality to enforce cybersecurity policy.

# PowerShell in Cybersecurity

---

## Offensive security:



PowerShell is often used as a “living off the land” tool, meaning it exists on the target’s computer and can be leveraged by attackers.



Once a system is breached, PowerShell is often used to retrieve a range of information within a network, such as user and server names.

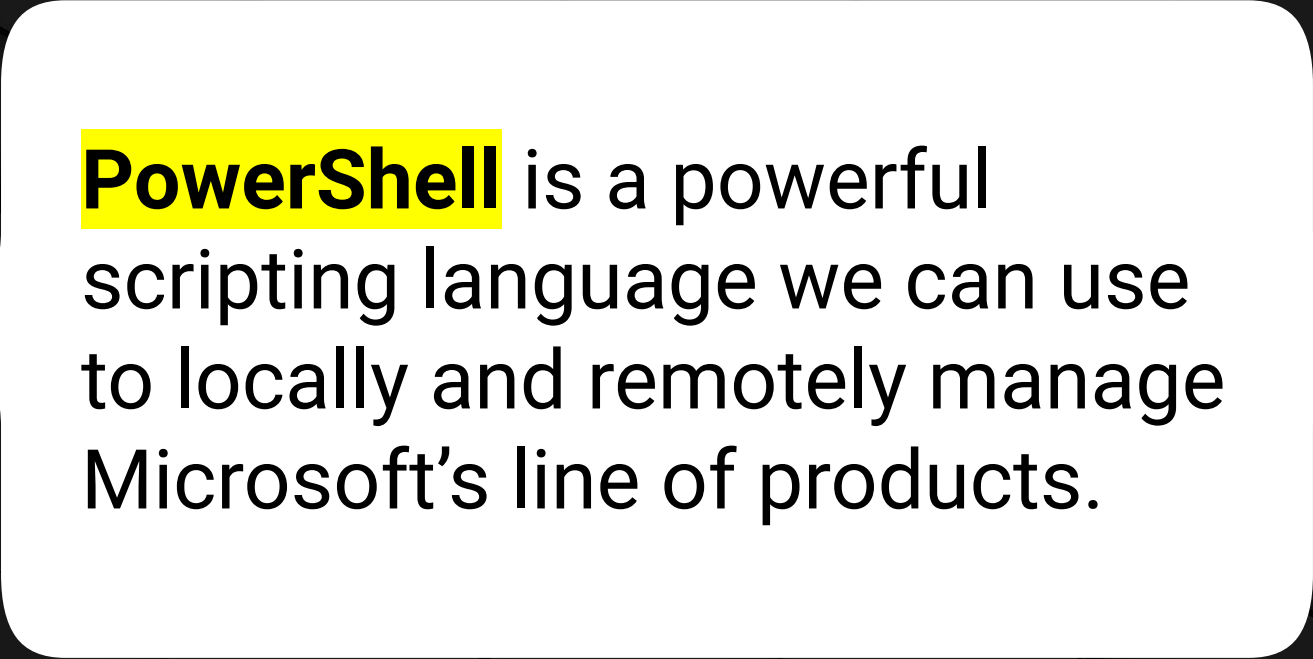


It can also be used to access and maintain access to other networked machines, if they are not properly secured.





So, what is PowerShell?



**PowerShell** is a powerful scripting language we can use to locally and remotely manage Microsoft's line of products.

Microsoft enterprise products are the most widely used by organizations.

- It is critical that system administrators and security professionals know PowerShell.
- PowerShell can be used to lock down and harden enterprise networks, leveraged by offensive security professionals, and exploited by malicious actors.



# PowerShell vs. CMD

# PowerShell vs. CMD

---

CMD's functionality is limited.

**CMD output is only available in simple text format.**

For unsupported file formats, we need to edit the output with meticulous character replacing.

**CMD command flags can be ambiguous, confusing, and specific to each command:**

Examples of different **/s** flags:

- **shutdown /s** shuts down a computer.
- **freedisk /s** specifies the name of an IP or remote computer to check disk space.

Examples of different **/d** flags.

- **shutdown /d** specifies a reason for shut down.
- **freedisk /d** specifies which disk drive to check.



**PowerShell provides clearly defined,  
universal parameters for commands.**

# PowerShell vs. CMD

---

Based on the language used in the following commands, what do you think they do?

```
Stop-Computer -Confirm
```

```
Stop-Computer -Force
```

# PowerShell vs. CMD

---

Based on the language used in the following commands, what do you think they do?

```
Stop-Computer -Confirm
```

Shuts down the machine with a confirmation prompt verifying that you want to shut it down.

```
Stop-Computer -Force
```

This command will immediately shut down the computer regardless of what is running.





We can use PowerShell piping for operations that CMD doesn't support.

# What are Objects?

---

To find the file sizes of subdirectories in C:\Windows\System in CMD, we must use a complex batch file consisting of the following:

```
@echo off
set size=0
for /r %%x in (System\*) do set /a size+=%%~zx
echo %size% Bytes
```

PowerShell can do this with a simple pipe ( | ):

```
dir C:\Windows\System -Recurse | Measure-Object -Sum Length
```

Grabs all the current directory and subdirectory contents.

The output is piped into this command, which measures files.

# What are Objects?

---

To find the file sizes of subdirectories in `C:\Windows\System` in CMD, we must use a complex batch file consisting of the following:

```
@echo off
set size=0
for /r %%x in (System\*) do set /a size+=%%~zx
echo %size% Bytes
```

PowerShell can do this with a simple pipe ( | ):

```
dir C:\Windows\System -Recurse | Measure-Object -Sum Length
```

Grabs all the current directory and subdirectory contents.

The output is piped into this command, which measures files.

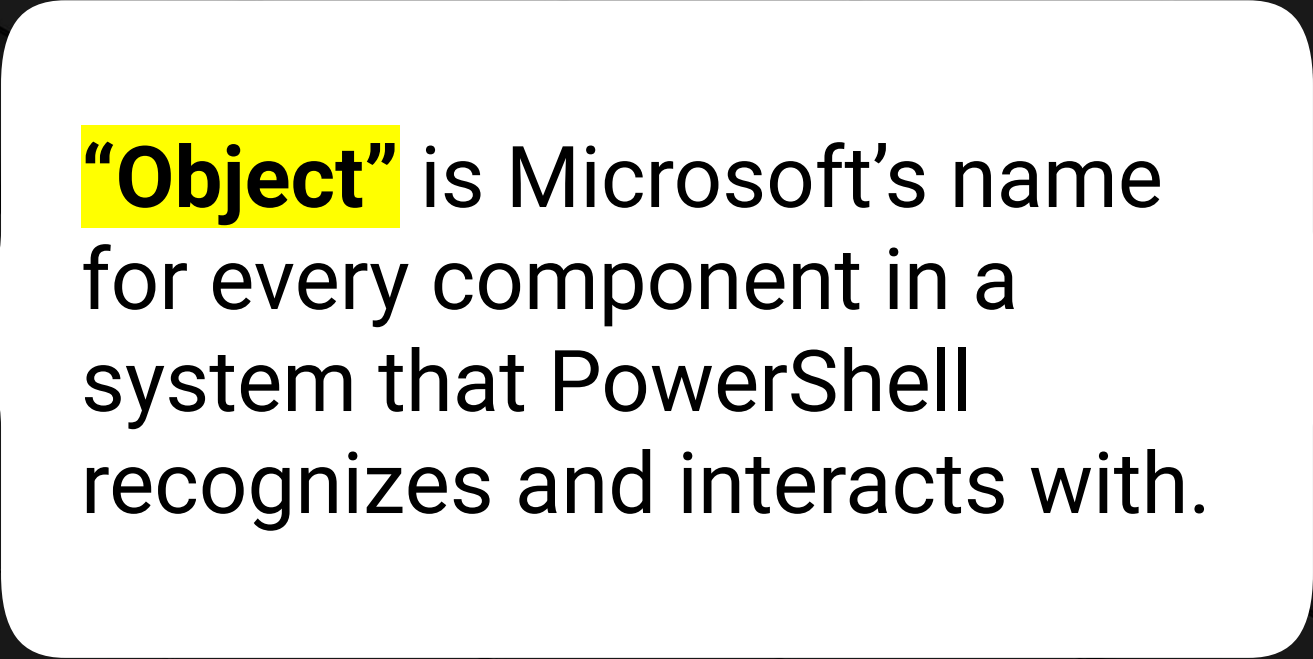
**Objects** are a very important PowerShell concept. We'll look at them next.



# Instructor Demonstration

---

## Powershell Objects?



**“Object”** is Microsoft’s name for every component in a system that PowerShell recognizes and interacts with.

# What are Objects?

---

If we run:

```
ls C:\Windows
```

All the files and directories in `C:\Windows` are processed by PowerShell.  
Each is an object, with its own properties.

```
C:\Documents\Recipes\Guacamole.doc
```

`Guacamole.doc` is the `file.name` property of the file (the object).



**Understanding everything as  
objects with properties allows us  
to use more specific commands  
to target the results we want.**

# What are Objects?

---

For example:

We can use a pipe to retrieve only objects containing the word “system”:

```
ls C:\Windows | Where-Object {$_.name -like "*system*"}
```

Lists contents of `C:\Windows`  
in which the `.name` property  
contains “system.”

`$_` refers to the current pipeline  
object, in this example `C:\Windows`.

We’ll cover syntax in more depth later.



# More PowerShell Benefits

---



We can confirm we're using the right commands with PowerShell's extensive internal documentation system.



PowerShell uses **aliases** to mimic Unix commands, like `ls` and `cat`.



Despite being a Microsoft product, PowerShell is open source and available on GitHub.



By contrast, there's no source code available for CMD, and writing tools are limited to batch scripts.

# PowerShell Commands:

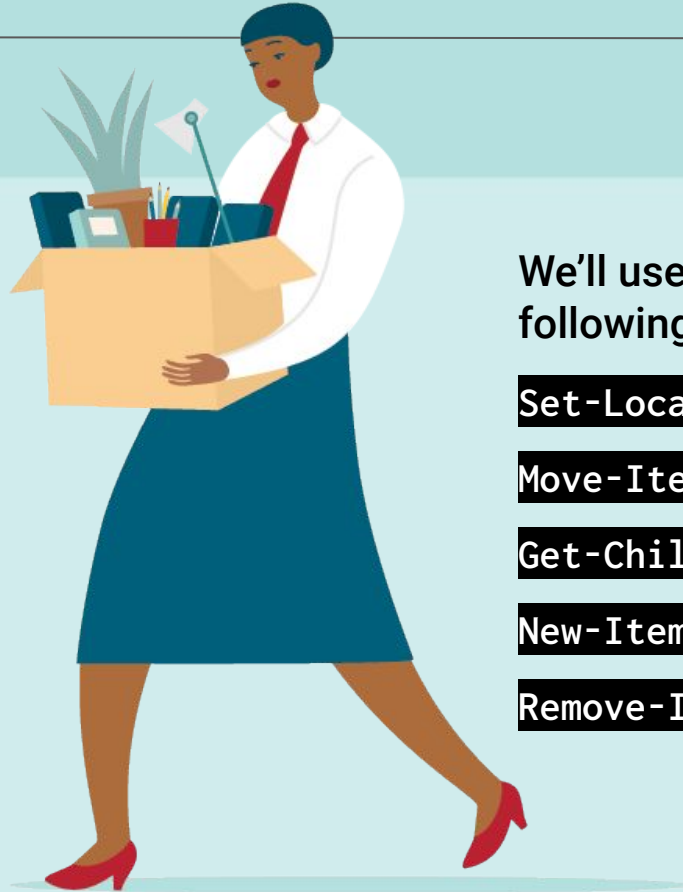
<b>cmdlet</b>	<b>Function</b>	<b>Equivalent Command</b>
<b>Set-Location</b>	Changes to specified directory	<b>cd</b>
<b>Get-ChildItem</b>	Returns current directory contents	<b>ls, dir</b>
<b>New-Item</b>	Makes new file or directory	<b>mkdir, touch</b>
<b>Remove-Item</b>	Deletes file or directory	<b>rm, rmdir</b>
<b>Get-Location</b>	Retrieves path to current directory	<b>pwd</b>
<b>Get-Content</b>	Returns file content	<b>cat, type</b>
<b>Copy-Item</b>	Copies a file from one location to another	<b>cp</b>
<b>Move-Item</b>	Moves item from one location to another	<b>mv</b>
<b>Write-Output</b>	Prints output	<b>echo</b>
<b>Get-Alias</b>	Shows aliases from the current session	<b>alias</b>
<b>Get-Help</b>	Retrieves information about command	<b>man</b>
<b>Get-Process</b>	Retrieves processes running on machine	<b>ps</b>
<b>Stop-Process</b>	Stops specific process	<b>kill</b>
<b>Get-Service</b>	Retrieves list of services	<b>service --status-all</b>

# Verbs-Nouns

---

Consider the following scenario:

- User **Alex** left the company.
- We want to remove their user account from the system, but we want to keep the reports they were working on.
- We need to move the reports files from their user desktop directory to a directory outside of the user.
- Along the way, we'll explore some other useful PowerShell commands.



We'll use the following commands:

**Set-Location**

**Move-Item**

**Get-ChildItem**

**New-Item**

**Remove-Item**

# Questions?

Any Questions?





# Instructor Demonstration

---

## Verbs-Nouns



**Now, we'll use parameters to  
customize the commands.**

# Parameters

---

Instead of using `New-Item` to create another file, we can add parameters to the `New-Item` command and create a directory, with a specific name and location:

```
New-Item -Path "C:\\" -Name "Logs" -ItemType "Directory"
```

## Path

Parameter specifying the location of this new directory.

## -Name

Parameter specifying the directory's name.

## -ItemType

Parameter specifying the type of item we want to create.

If we don't specify `"Directory"`, it will default to a file.



**Let's take a look at some  
more parameters...**



# Questions?

Any Questions?





# Instructor Demonstration

---

## Parameter Examples



## Activity: Moving and Creating Directories

In this activity, you will work as a junior sysadmin to vet a process to create Windows Event logs.

First we need to create directories to store our information. We'll run PowerShell commands to create, rename, and move items.

Suggested Time:

10 Minutes



Time's Up! Let's Review.

# Questions?

Any Questions?



# Generating Windows Event Log Files with Parameters and Pipelines



**Now we'll continue building on  
parameters by chaining commands  
with pipelines.**





## Pipeline Demo Setup

---

In the following demo, we're continuing our role as a junior sysadmin.

Our CIO asked us to retrieve multiple types of logs from our Windows 10 machine and save them as JSON files in our newly created **C:\Logs** directory. They will later be imported to a Splunk SIEM for analysis.

Use your cheatsheet to follow along.





# Instructor Demonstration

---

## PowerShell Parameters

# Piping Logs to JavaScript Object Notation with ConvertTo-Json

---

Now that we've used parameters to get the logs we need, we will output them to a file that can be later used by log analysis applications. This is where **pipelines** come in.

## PowerShell

```
Get-WinEvent -LogName System -MaxEvents 10 | ConvertTo-Json
```



# Instructor Demonstration

---

## Piping Logs to JSON



# Activity: Generating Windows Event Log Files

In this activity, you will create and save log files to C: /Logs.

Suggested Time:

15 Minutes



Break



Time's Up! Let's Review.

# Questions?

Any Questions?



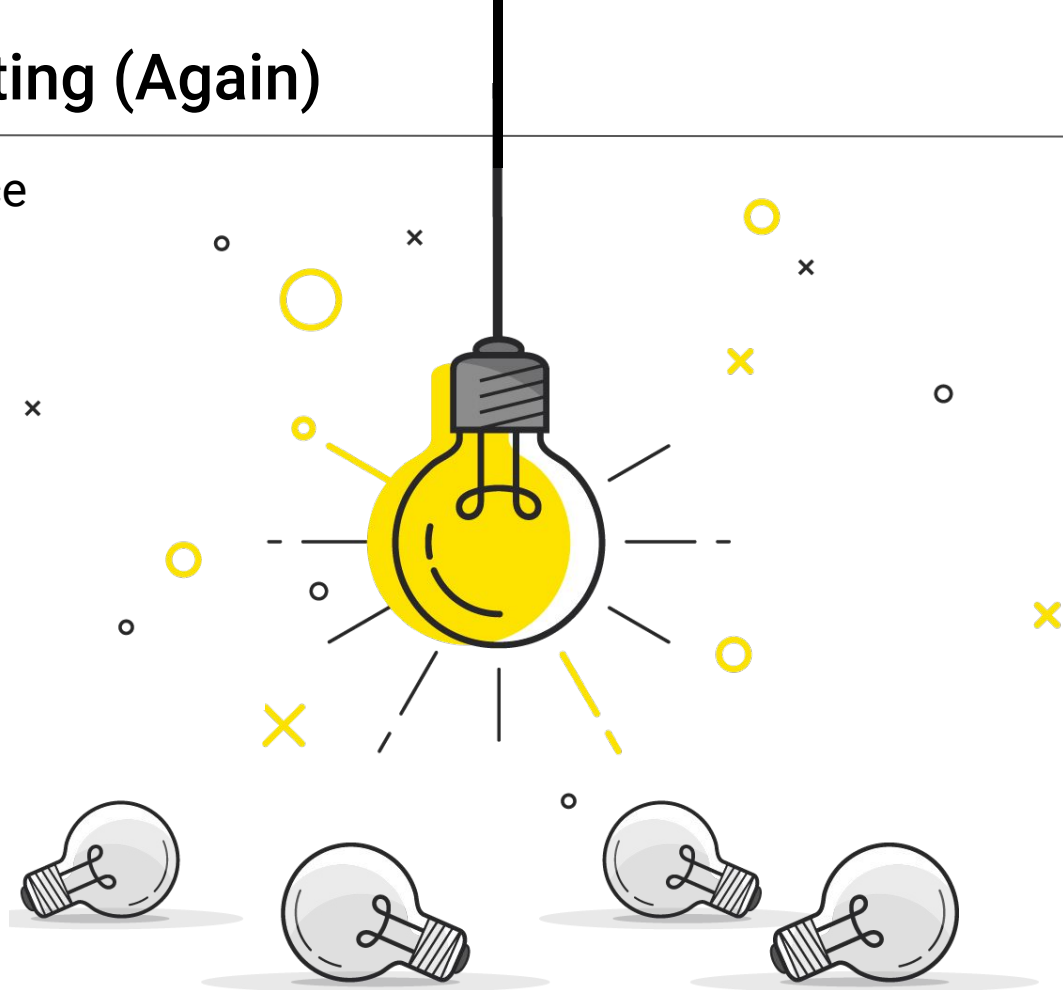
# Scripting with PowerShell



# The Importance of Scripting (Again)

We've emphasized the importance and convenience of scripting in our past sysadmin units.

- Scripts allow sysadmins and security professionals to automate and execute basic to advanced procedures and operations.
- Scripts can be used for everything from setting basic firewall rules to standing up entire cloud virtual machine environments with networking, storage, and users.



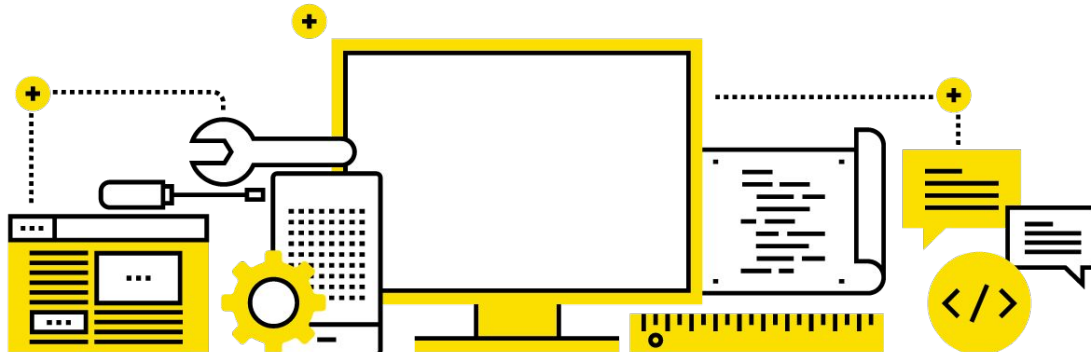


**Like Linux, PowerShell allows us to script many commands in sequence:.**

# Scripting with PowerShell

For example, suppose you need to set up Windows workstations for users in the accounting department. You could create a script that does the following, in order:

- 01 Pulls sensitive accounting data and files from a file server to a specified directory.
- 02 Downloads AppLocker, a program to create rules that allow or block certain executables or scripts from running on a system.
- 03 Deploys application control policies for AppLocker to restrict executables in a directory so only people in the accounting group can run them.



# Scripting Demo Scenario

In this demonstration:

We will create a script in VS Code to remove Skype.

**This Windows workstation has been created to simulate a previous user having installed applications that are considered largely unnecessary and potential vectors of attack.**

- Some of these settings include telemetry tracking and advertising IDs. The default installed applications include Skype.
- We want to remove these applications to reduce the attack surface area for this workstation. Instead of trusting our users to not use these apps, we're going to remove the possibility.



# Scripting Demo Scenario

---

In this demonstration, we will complete the following steps:

01

Create a PowerShell script file and execute it.

02

Use a cmdlet to import various items to interact with in a PowerShell script.

03

Use a foreach loop to go through each item in an imported CSV file.



## Instructor Demonstration

---

# PowerShell Scripts and Removing Skype

# PowerShell Scripts and Removing Skype

---

We've just deleted a useless app from our machine.

While we could remove the Windows Store apps one-by-one, it would be more efficient to create a **script that loops through a list** of the apps and uninstalls them all at once.





More specifically, we can loop through **comma-separated values (CSV)** files.



# CSV Files

---

CSVs are plain text files that contain simply structured data (fields) separated by commas. The **top line** of a CSV file contains the header — the row that describes each field.

```
appxpkg,name,description
```

```
"Microsoft.ZuneMusic","Zune","Microsoft's Zune Music Player"
```

```
"Microsoft.Music.Preview","Music Preview", "Microsoft's Music Preview"
```

```
"Microsoft.XboxGameCallableUI","Xbox Gaming GUI", "Microsoft's Xbox Overlay"
```

```
[CSV contents truncated]
```

# CSV Files

---

Sysadmins and security professionals use CSV files containing lists of items they need to parse through.

## System administrator

May use a CSV file to maintain a list of employee email addresses

## Penetration tester

Might have a list of IP addresses and corresponding domain and subdomain names to use in a test.

# CSV Files

```
appxpkg,name,description
"Microsoft.ZuneMusic","Zune","Microsoft's Zune Music Player"
"Microsoft.Music.Preview","Music Preview", "Microsoft's Music Preview"
"Microsoft.XboxGameCallableUI","Xbox Gaming GUI", "Microsoft's Xbox Overlay"
[CSV contents truncated]
```




appxpkg	name	description
Microsoft.ZuneMusic	Zune	Microsoft's Zune Music Player
Microsoft.Music.Preview	Music Preview	Microsoft's Music Preview
Microsoft.XboxIdentityProvider	Xbox ID Provider	Microsoft's Xbox Live Account Management

# foreach Loops

---

We can use foreach loops to iterate through these files' items.

The foreach loop in PowerShell is similar to the for loop in Linux, but it is mainly used for looping through files or read-only structured data that you need to loop through.



appxpkg	name	description
Microsoft.ZuneMusic	Zune	Microsoft's Zune Music Player
Microsoft.Music.Preview	Music Preview	Microsoft's Music Preview
Microsoft.XboxIdentityProvider	Xbox ID Provider	Microsoft's Xbox Live Account Management

# foreach Loops and CSV Files

---

Sysadmins and security professionals will use CSV and **foreach** loops together:

A system administrator may use a CSV file to maintain a list of employee email addresses and usernames.



They can use a **foreach** loop to loop through each item and change the passwords.

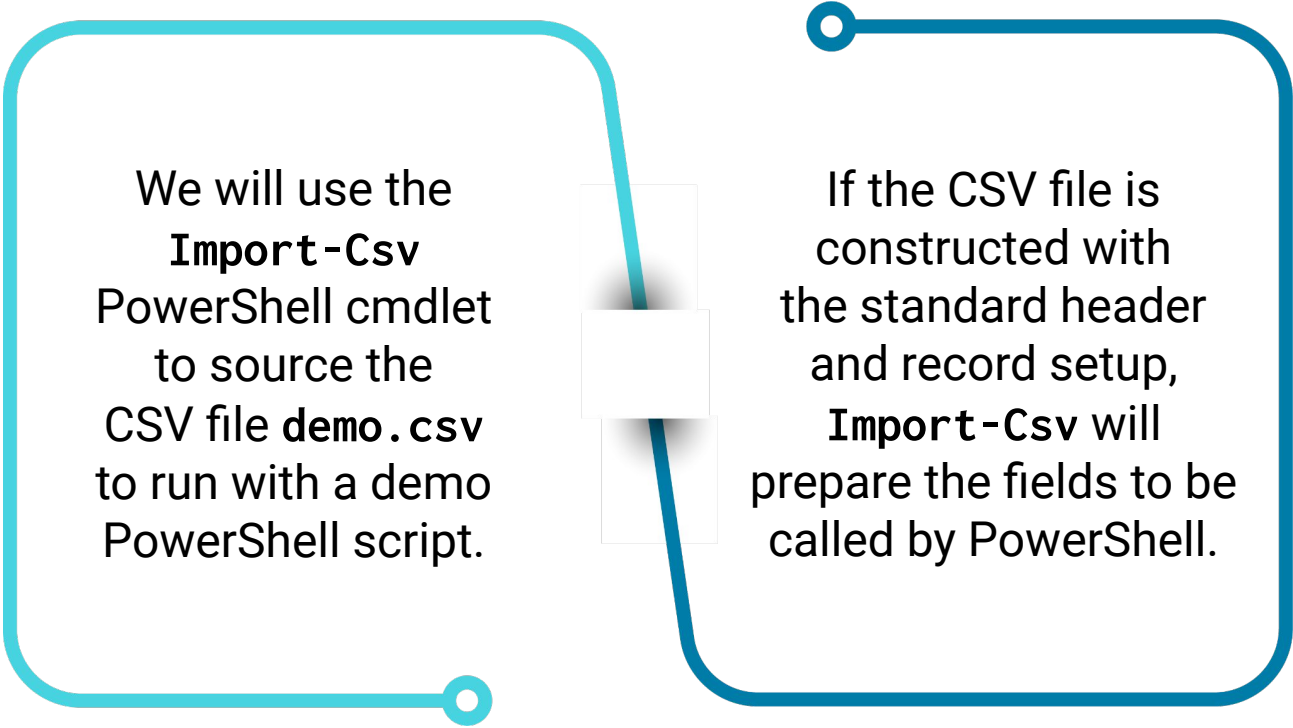
A penetration tester might have a list of IP addresses and corresponding domain and subdomain names to use in a test.



They can use a **foreach** loop to try out each password with a known username.

## Now, We Will Continue our Demo Scenario.

---



We will use the **Import-Csv** PowerShell cmdlet to source the CSV file **demo.csv** to run with a demo PowerShell script.

If the CSV file is constructed with the standard header and record setup, **Import-Csv** will prepare the fields to be called by PowerShell.



## Instructor Demonstration

---

Using `Import-Csv` with a `foreach` Loop



# Activity: Removing Unnecessary Packages with PowerShell Scripts

In this activity, you'll use PowerShell and a CSV file to create a script that removes unwanted applications.

Suggested Time:

15 Minutes





Time's Up! Let's Review.

# Questions?





## Shut Down Your Machines



Everyone must shut down their Windows RDP Host Machine.

You will need the remaining hours to complete your homework.

# Questions?

Any Questions?



*The  
End*