Sysmon Project: Detailed Log Analysis and Incident Response Using PowerShell

Task: Analyze Sysmon logs to investigate and answer specific incident response scenarios, emphasizing PowerShell scripting for automation and Active Directory insights.

Steps and Experience Gained

1. Filtering Event Logs with PowerShell

- **Objective**: Identify the number of Event ID 3 entries from a specific `.evtx` log file.
- **PowerShell Command**:

```
Get-WinEvent -Path
"C:\Users\THM-Analyst\Desktop\Scenarios\Practice\Filtering.evtx"
-FilterXPath '*/System/EventID=3' | Measure-Object -Line
```

- **Process**:
    - Used the `-Path` parameter to specify the location of the event log file.
    - Applied `-FilterXPath` to isolate entries with `EventID=3` (network events).
    - Leveraged `Measure-Object -Line` to count the filtered entries efficiently, avoiding overwhelming output.
- **Outcome**: Counted 73,591 relevant log entries, showcasing precise log analysis using PowerShell.

---

2. Identifying Specific Event Details

- **Objective**: Find the UTC creation time of the first network event.
- **Steps**:
    - Parsed logs in Event Viewer to locate network-related entries.
    - Switched to XML View to identify the `UTC Time` tag for the first relevant log.
- **PowerShell Insight**: While the Event Viewer GUI was used here due to query complexity, this step highlighted areas where future XPath filtering in PowerShell can be optimized.

---

3. Investigating Malicious Registry Activity

- **Objective**: Locate a malicious registry key tied to `svchost.exe`.
- **PowerShell Command**:

```
Get-WinEvent -Path
"C:\Users\THM-Analyst\Desktop\Scenarios\Practice\Filtering.evtx" |
Where-Object { $_.Message -like '*svchost.exe*' }
```

- **Process**:
    - Scanned logs for entries containing references to `svchost.exe`.

Examined the detailed properties of relevant logs to extract the full registry path:

```
HKLM\System\CurrentControlSet\Enum\WpdBusEnumRoot\UMB\...
```

- **Outcome**: Identified the exact registry key, demonstrating the ability to parse and investigate registry-related logs programmatically.

---

4. Decoding Malicious PowerShell Commands

- **Objective**: Analyze and decode PowerShell commands used to launch a payload.
- **PowerShell Analysis**:

```
$encodedCommand = "<Base64EncodedCommand>"
[System.Text.Encoding]::Unicode.GetString([Convert]::FromBase64String(
$encodedCommand))
```

- **Process**:
    - Extracted Base64-encoded strings from logs.

Decoded the strings to reveal the full PowerShell command:

```
“C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NonI -W
hidden -c ..."
```

- **Outcome**: Deciphered the adversary's command used to create persistence via a scheduled task.

---

5. Tracing Scheduled Tasks for Persistence

- **Objective**: Identify malicious scheduled tasks executed via `schtasks.exe`.
- **PowerShell Command**:

```
Get-WinEvent -Path
"C:\Users\THM-Analyst\Desktop\Scenarios\Practice\Filtering.evtx" |
Where-Object { $_.Message -like '*schtasks.exe*' }
```

- **Process**:
    - Focused on events containing `schtasks.exe` references.

Extracted command-line arguments to understand how tasks were created, revealing:

```
“C:\WINDOWS\system32\schtasks.exe” /Create /F /SC DAILY /ST 09:00 /TN
Updater /TR “C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
...”
```

- **Outcome**: Identified the exact scheduled task used by the adversary for persistence.

---

6. Network Activity and C2 Connections

- **Objective**: Uncover adversary IP addresses and C2 server activity.
- **PowerShell Command**:

```
Get-WinEvent -Path
"C:\Users\THM-Analyst\Desktop\Scenarios\Practice\Filtering.evtx" |
Where-Object { $_.Message -like '*Network connections detected*' }
```

- **Process**:
    - Isolated logs referencing network activity.
    - Analyzed source and destination IPs, correlating them to C2 infrastructure.
- **Outcome**: Discovered adversary IPs (`172.30.1.253`, `172.168.103.188`) and connection details (port `80`, `Empire` C2 framework).

Conclusion

This project provided hands-on experience in:

1. **PowerShell Scripting**: Filtering and analyzing large datasets, decoding Base64 strings, and tracing adversarial activity through logs.
2. **Active Directory Insight**: Identifying and analyzing registry modifications and process activity tied to malicious actors.
3. **Incident Response**: Automating log investigation tasks and uncovering critical details such as payloads, registry keys, scheduled tasks, and C2 connections.

This showcases proficiency in log analysis and practical incident response techniques in cybersecurity.