Create a VM and onboard it to MDE if you haven't already

Run this PowerShell command on your VM after onboarding it to MDE

```
Invoke-WebRequest -Uri
'https://raw.githubusercontent.com/joshmadakor1/lognpacific-publ
ic/refs/heads/main/cyber-range/entropy-gorilla/portscan.ps1'
-OutFile 'C:\programdata\portscan.ps1';cmd /c powershell.exe
-ExecutionPolicy Bypass -File C:\programdata\portscan.ps1
```

1. Preparation

- **Goal**: Set up the hunt by defining what you're looking for.
    - The server team has noticed a significant network performance degradation on some of their older devices attached to the network in the 10.0.0.0/16 network. After ruling out external DDoS attacks, the security team suspects something might be going on internally.
- **Activity**: Develop a hypothesis based on threat intelligence and security gaps (e.g., “Could there be lateral movement in the network?”).
    - All traffic originating from within the local network is by default allowed by all hosts. There is also unrestricted use of PowerShell and other applications in the environment. It’s possible someone is either downloading large files or doing some kind of port scanning against hosts in the local network.

2. Data Collection

- **Goal**: Gather relevant data from logs, network traffic, and endpoints.
    - Consider inspecting the logs for excessive successful/failed connections from any devices. If discovered, pivot and inspect those devices for any suspicious file or process events.
- **Activity**: Ensure data is available from all key sources for analysis.
    - Ensure the relevant tables contain recent logs:
        - DeviceNetworkEvents
        - DeviceFileEvents
        - DeviceProcessEvents

3. Data Analysis

- **Goal**: Analyze data to test your hypothesis.
- **Activity**: Look for anomalies, patterns, or indicators of compromise (IOCs) using various tools and techniques.

    - Anything going on that you can notice in terms of excessive network connections to/from any hosts? Take note of the the query/logs/time

4. Investigation

- **Goal**: Investigate any suspicious findings.
- **Activity**: Dig deeper into detected threats, determine their scope, and escalate if necessary. See if anything you find matches TTPs within the MITRE ATT&CK Framework.
    - Search the DeviceFileEvents and DeviceProcessEvents tables around the same time based on your findings in the DeviceNetworkEvents tables to see if you can find more evidence for the cause of network slowdowns.
    - You can use ChatGPT to figure this out by pasting/uploading the logs: Scenario 2 - TTPs

5. Response

- **Goal**: Mitigate any confirmed threats.
- **Activity**: Work with security teams to contain, remove, and recover from the threat.
    - Can anything be done?

6. Documentation

- **Goal**: Record your findings and learn from them.
- **Activity**: Document what you found and use it to improve future hunts and defenses.
    - Document what you did

7. Improvement

- **Goal**: Improve your security posture or refine your methods for the next hunt.
- **Activity**: Adjust strategies and tools based on what worked or didn't.
    - Anything we could have done to prevent the thing we hunted for? Any way we could have improved our hunting process?

Notes / Findings:

```
// Count up failed connections, take note of any IPs with excessive connections
DeviceNetworkEvents
| where ActionType == "ConnectionFailed"
| summarize FailedConnectionsAttempts = count() by DeviceName, ActionType, LocalIP, RemoteIP
| order by FailedConnectionsAttempts desc
```

```
// Observe total failed connections for a specific IP Address against other IPs
let IPInQuestion = "10.0.0.5";
DeviceNetworkEvents
| where ActionType == "ConnectionFailed"
| where LocalIP == IPInQuestion
| summarize FailedConnectionsAttempts = count() by DeviceName, ActionType, LocalIP
| order by FailedConnectionsAttempts desc

// Observe all failed connections for the IP in question.
let IPInQuestion = "10.0.0.5";
DeviceNetworkEventspo
| where ActionType == "ConnectionFailed"
| where LocalIP == IPInQuestion
| order by Timestamp desc

// Observe DeviceProcessEvents for the past 10 minutes of the unusual activity found
let VMName = "windows-target-1";
let specificTime = datetime(2024-10-18T04:09:37.5180794Z);
DeviceProcessEvents
| where Timestamp between ((specificTime - 10m) .. (specificTime + 10m))
| where DeviceName == VMName
| order by Timestamp desc
| project Timestamp, FileName, InitiatingProcessCommandLine
```

Timeline Summary and Findings:

After observing failed connection requests from a suspected host (10.0.0.5) in chronological order, I observed and noticed a port scan that was taking places due to the sequential order of the ports. There were several ports scans being conducted:

```
// Observe all failed connections for the IP in question. Notice anything?

let IPInQuestion = "10.0.0.5";

DeviceNetworkEvents

| where ActionType == "ConnectionFailed"

| where LocalIP == IPInQuestion
```

```
| order by Timestamp desc
```

We changed over to the DeviceProcessEvents table to see if we could see anything that was suspicious around the time the port scan started. We noticed a PowerShell script named portscan.ps1 launch at **2025-03-02T16:37:15.126608Z**

```
// Observe DeviceProcessEvents for the past 10 minutes of the unusual activity found

let VMName = "windows-target-1";

let specificTime = datetime(2025-03-02T16:40:41.2467663Z);

DeviceProcessEvents

| where Timestamp between ((specificTime - 10m) .. (specificTime + 10m))

| where DeviceName == VMName

| where FileName == "powershell.exe"

| order by Timestamp desc

| project Timestamp, FileName, InitiatingProcessCommandLine
```

It appears that portscan.ps1 was created at 2025-03-03T00:22:18.0018711Z according to DeviceFileEvents. It seems that PoweShell was used to download the portscan.ps1 file from https://raw.githubusercontent.com/joshmadakor1/lognpacific-public/refs/heads/main/cyber-range/entropy-gorilla/portscan.ps1

I logged into the computer suspected and observed the powershell script (portscan.ps1) that was utilized to conduct the port scan:

```powershell
# Define the log file path
$logFile = "C:\ProgramData\entropygorilla.log"
$scriptName = "portscan.ps1"

# Function to log messages
function Log-Message {
    param (
        [string]$message,
        [string]$level = "INFO"
    )
    $timestamp = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    $logEntry = "$timestamp [$level] [$scriptName] $message"
    Add-Content -Path $logFile -Value $logEntry
}

# Define the range of IP addresses to scan
$startIP = 4
$endIP = 10
$baseIP = "10.0.0."

# Expanded list of common ports (well-known port numbers 0-1023 + some higher)
$commonPorts = @(21, 22, 23, 25, 53, 69, 80, 110, 123, 135, 137, 138, 139, 143, 161, 194, 443, 445, 465, 587, 993, 995, 3306, 3389, 5900, 8080, 8443)
```

We observed the port scan script was launched by the SYSTEM account. This is not considered expected behavior in our circumstances and was not something that was setup by the administrators, so I proceeded to isolate the device and run a malware scan.

The malware scan produced no results, so out of caution, we kept the device isolated and put in a ticket to have it rebuilt and reimaged.

MITRE ATT&CK Framework Related TTPS:

1. Reconnaissance (Tactic TA0043)

- **Technique: T1595 - Active Scanning**
    - **Sub-technique:** T1595.001 - Scanning IP Blocks
    - **Sub-technique:** T1595.002 -Vulnerability Scanning
    - **Justification:** The sequential failed connection attempts and the identification of `portscan.ps1` clearly indicate an active port scan. This is a classic reconnaissance technique used to discover open ports and potentially identify services running on the target system. The query explicitly looking for `ConnectionFailed` events in sequential order strongly suggests port scanning. Although no vulnerability were explicitly scanned for, the nature of port scanning for this threat hunt is closely related to Vulnerability Scanning.
-

2. Execution (Tactic TA0002)

- **Technique: T1059 - Command and Scripting Interpreter**

    - **Sub-technique:** T1059.001 - PowerShell
    - **Justification:** The `portscan.ps1` script was executed using PowerShell. This is explicitly identified in the `DeviceProcessEvents` query where `FileName == "powershell.exe"` and the script name are observed.
-

3. Resource Development (Tactic TA0042)

- **Technique: T1588 - Obtain Capabilities**
    - **Sub-technique:** T1588.002 - Tool
    - **Justification:** The powershell script was observed to have been downloaded from `https://raw.githubusercontent.com/joshmadakor1/lognpacific-public/refs/heads/main/cyber-range/entropy-gorilla/portscan.ps1`, a tool to conduct a port scan.