**Post-Compromise Memory Forensics Report**

**Project Title**: Memory Forensics and Post-Compromise Analysis  
**Analyst**: Sumit Kumar  
**Date of Analysis**: June 8, 2025  
**Platform**: Windows 10 (Target VM), Kali Linux (Attacker)  
**Tooling**: Volatility 3, Sysmon, PowerShell, RAM Capturer

**1. Executive Summary**

This investigation documents a simulated post-compromise intrusion scenario performed in a lab setting using a Windows 10 virtual machine. The goal was to emulate real-world adversary behavior and perform memory forensics using Volatility 3 to detect persistence mechanisms, malicious in-memory activity, and command & control (C2) traffic. Findings confirm scheduled task-based persistence, PowerShell-based execution, outbound HTTP traffic, and injected RWX memory in the PowerShell process — all traceable through memory artifacts.

**2. Scope of Investigation**

**Image Analyzed**: 20250608.mem (2 GB)  
**Hash (SHA256)**: [screenshot in appendix]  
**Source System**: Windows 10 VM (hosted on VirtualBox)  
**Objective**:

* Capture a live compromised system's memory
* Detect persistence mechanisms
* Analyze network and memory-level artifacts left by the simulated attack

**3. Tools and Data Integrity**

* **Memory Acquisition Tool**: Belkasoft RAM Capturer
* **Hashing**: SHA256 generated using PowerShell (Get-FileHash)
* **Analysis Engine**: Volatility 3 (v2.26.2), Python 3.11
* **Artifact Location**: Plugin logs, screenshots, PowerShell script payload
* **Integrity Verified**: Captured image hash saved alongside .mem file (see has\_output and run.png)

**4. Forensic Analysis Findings**

**4.1 Persistence via Scheduled Task**

* Scheduled task BackupSyncRunner discovered.
* Executes: powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -File C:\Users\Public\backdoor.ps1
* Persistence trigger: **At logon**
* Plugin: windows.scheduled\_tasks
* Screenshot: Task arguments visible in Process-tree.png

**4.2 Process Execution Chain**

* powershell.exe (PID 396) observed with parent explorer.exe (PID 3244)
* Child process: cmd.exe → indicative of LOLBAS technique
* Confirmed via windows.pslist and windows.cmdline
* Screenshot: Start-Process usage in PowerShell (see mm.png)

**4.3 In-Memory Execution (malfind)**

* powershell.exe contains 5+ RWX pages
* Disassembly shows jmp rax, movabs → indicative of shellcode or PE injection
* Plugins used: windows.malfind
* Screenshot: Disassembly pattern and PID evidence (see mm.png)

**4.4 Network Communication**

* Outbound HTTP connection detected from powershell.exe (PID 396)
* To: 23.192.228.80 over TCP port 80
* Plugin: windows.netscan
* Matches manual Invoke-WebRequest test
* Screenshot: WebRequest output (see has\_output and run.png)

**5. Indicators of Compromise (IOCs)**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Value | Source Plugin | Description |
| Task Name | BackupSyncRunner | scheduled\_tasks | Persistence via scheduled task |
| File | C:\Users\Public\backdoor.ps1 | cmdline | Simulated malware PowerShell script |
| Process | powershell.exe (PID 396) | pslist, cmdline | Suspicious scripting interpreter |
| IP Addr | 23.192.228.80 | netscan | Simulated C2 server (HTTP) |
| Memory | RWX pages at 0x15369c10000+ | malfind | In-memory payloads in powershell.exe |
| CmdLine | -ExecutionPolicy Bypass ... | cmdline | Hidden script execution bypassing policy |

**6. MITRE ATT&CK Technique Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| Tactic | Technique ID | Name | Evidence |
| Persistence | T1053.005 | Scheduled Task | Task: BackupSyncRunner |
| Execution | T1059 | PowerShell | Process + script + cmdline |
| Defense Evasion | T1027.005 | Obfuscated Files or Information | -WindowStyle Hidden + Bypass flag |
| Privilege Escalation | T1055.002 | Process Injection | RWX memory in PowerShell |
| C2 | T1071.001 | Application Layer Protocol: HTTP | WebRequest to external IP |

**7. Strategic Recommendations**

* Enforce script block and transcription logging for PowerShell
* Detect and alert on powershell.exe with hidden or encoded command-line args
* Monitor scheduled tasks created outside admin-approved contexts
* Use DNS and proxy-level rules to restrict outbound traffic from scripting engines
* Periodically scan memory for RWX segments in user processes

**8. Appendix (Screenshots and Evidence)**

| **Description** | **Filename** |
| --- | --- |
| VM clean snapshot | Preattackclean state.png |
| Scheduled task persistence | Process-tree.png |
| PowerShell execution & beaconing | execution.png |
| RAM capture success | ram\_capture\_done.png |
| Hash confirmation output | has\_output and run.png |
| Volatility setup & validation | execution.png |