Threat Hunt Report: Common Windows Post-Exploitation Tool Used

Detection of script/executable matching name of common Post-Exploitation tool ran on `peter-windows-vm-soc` (`peter-windows-v` in Defender) from command line.

Scenario:

Management is requesting the Security Team harden the environment against Post-Exploitation tools used by threat actors, as there has been a recent rash of account compromises providing initial footholds as evidenced by SIEM alerts. As an initial measure, Management has requested alerts and basic remediations against well-known Post-Exploitation/Privilege Escalation scripts and executables run by attackers trying to get a 'quick win.' Often attackers will gain a foothold on a machine, and use automated Post Exploitation tools/scripts to quickly enumerate potential vectors for full/further compromise of the box - these tools can come in the form of scripts, executables, or other command-line commands. Additionally, attackers could attempt to use the new access as a 'pivot' to reach further into our environment, move laterally, or even setup Command & Control (C2)..

Less-stealthy attackers may not rename the tools they use on our infrastructure, this is where we hope to gain a basic detection against this kind of activity - we want to detect Post-Exploitation tools based on their names as they are run. If a threat actor has command-line access to the box, and is able to download or transfer tools/scripts to the target, they may not rename the tool, which could give us a chance to detect and mitigate.

Because our enterprise is "Windows-heavy", some examples of popular Windows Post-Exploitation or pivoting tools include Mimikatz.exe, PowerUp.ps1, Chisel.exe, plink.exe, WinPEAS.exe, Sherlock.ps1, Meterpreter.exe, Powersploit.ps1, Windows-Exploit-Suggester.exe, Seatbelt.exe, but there are many more. If any files matching a list (including the files above) of popular tools is run, an alert will be generated, and the device which the command ran on will be isolated, with an Investigation Package populated.

The overall goal of this threat hunt is to detect well-known Post-Exploitation/Privesc/Pivot tools being ran on the CLI on `peter-windows-vm-soc` (`peter-windows-v` for the purposes of this document), and trigger several automatic remediations in addition to an alert. The KQL will allow for easy additions of extra scripts/tools to expand identification of known threats.

Note: During POC, all alerts apply ONLY to host '**`peter-windows-vm-soc`**'.

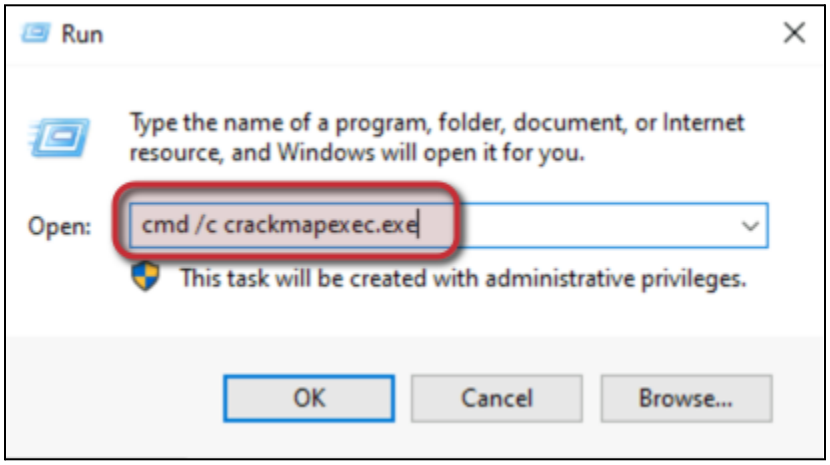
High-Level IoC Discovery Plan

- Define a list of known Post-Exploitation / Privesc tools to detect, define it as a list.
- Check `DeviceProcessEvents` for any signs of cmd.exe, Powershell, or explorer.exe usage on "`peter-windows-v`" (truncated VM name in Defender).
- From the above results, filter for results where the ProcessCommandLine contains regex matches for the list of defined tools/scripts.
- The activity is detected because the name matches a known tool/script, allowing for automatic remediations/alerting.

Steps Taken

1. Used KQL to search for cmd.exe, powershell.exe, or explorer.exe events containing several well-known Post-Exploitation tools being used, to see if one tool could be detected.

Example post-exploitation command line activity:



Example KQL query to detect crackmapexec.exe being ran on 'peter-windows-v':

```
let file = @'(?i)(crackmapexec\.exe)';
DeviceProcessEvents
```

```
| where isnotempty(ProcessCommandLine)
| where ProcessCommandLine has_any("cmd.exe", "powershell.exe", "explorer.exe")
| where ProcessCommandLine matches regex file
| where DeviceName startswith "peter-windows-v"
```

2. Generated several lists of relevant Windows Post-Exploitation and Privesc scripts/files, sorted through the relevant ones. Added them to the KQL query to catch those additional scripts/files.

Updated KQL query with additional tools:

```
let files =
@'(?i)(mimikatz\.exe|wmic\.exe|nircmd\.exe|netcat\.exe|powerup\.ps1|powershell
empire\.ps1|sharphound\.exe|seimpersonateprivilege\.exe|dirtycow\.exe|procdump\.e
xe|sharpdesk\.exe|psexec\.exe|incognito\.exe|add-localadmin\.ps1|chisel\.exe|jmsp
\.exe|tokenmanipulation\.ps1|meterpreter\.exe|winpeas\.exe|rubeus\.exe|lazagne\.e
xe|sherlock\.ps1|pstools\.exe|getadmin\.ps1|kalilinux\.exe|hackersploit\.exe|resp
onder\.exe|eternalblue\.exe|crackmapexec\.exe|netcat64\.exe|hashdump\.ps1|impacke
t\.py|powersploit\.ps1|zerologon\.exe|windows-exploit-suggester\.exe|dcomignite\.
exe|bypassuac\.exe|powershellwebshell\.ps1|kerberosbrute\.exe|beroot\.ps1)';
\\One line with all tools as a variable
DeviceProcessEvents
| where isnotempty(ProcessCommandLine)
| where ProcessCommandLine has_any("cmd.exe", "powershell.exe", "explorer.exe")
| where ProcessCommandLine matches regex files
| where DeviceName startswith "peter-windows-v"
```

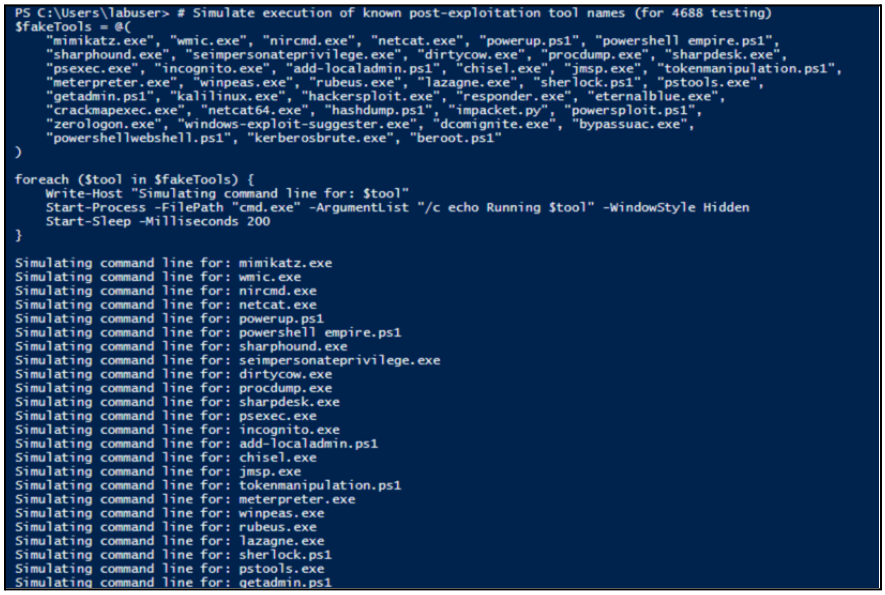
3. Used a Powershell script to simulate the listed tools being ran on the VM in question. This script effectively calls the name of each tool in the list using Powershell, with a slight wait between each call.

Powershell script to simulate tools being called:

```
# Simulate execution of known post-exploitation tool names (for 4688 testing)
$fakeTools = @(
    "mimikatz.exe", "wmic.exe", "nircmd.exe", "netcat.exe", "powerup.ps1",
"powershell empire.ps1",
```

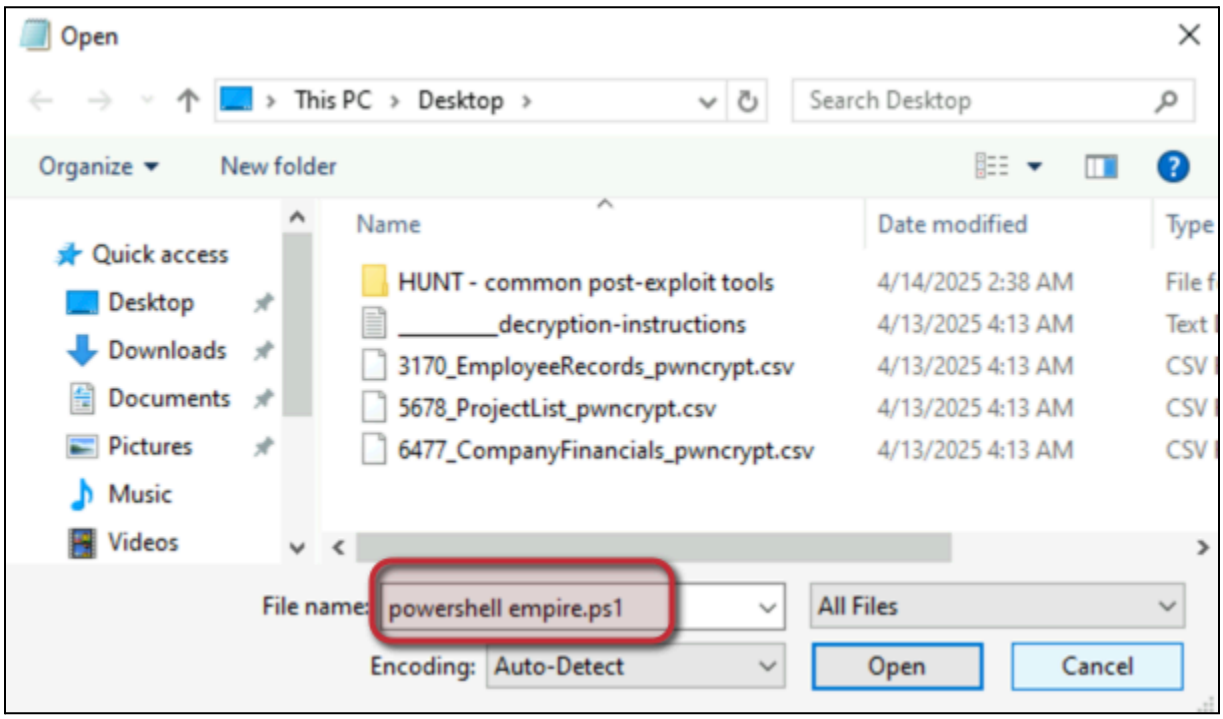
```
    "sharphound.exe", "seimpersonateprivilege.exe", "dirtycow.exe",
"procdump.exe", "sharpdesk.exe",
    "psexec.exe", "incognito.exe", "add-localadmin.ps1", "chisel.exe",
"jmsp.exe", "tokenmanipulation.ps1",
    "meterpreter.exe", "winpeas.exe", "rubeus.exe", "lazagne.exe",
"sherlock.ps1", "pstools.exe",
    "getadmin.ps1", "kalilinux.exe", "hackersploit.exe", "responder.exe",
"eternalblue.exe",
    "crackmapexec.exe", "netcat64.exe", "hashdump.ps1", "impacket.py",
"powersploit.ps1",
    "zerologon.exe", "windows-exploit-suggester.exe", "dcomignite.exe",
"bypassuac.exe",
    "powershellwebshell.ps1", "kerberosbrute.exe", "beroot.ps1"
)


foreach ($tool in $fakeTools) {
    Write-Host "Simulating command line for: $tool"
    Start-Process -FilePath "cmd.exe" -ArgumentList "/c echo Running $tool"
-WindowStyle Hidden
    Start-Sleep -Milliseconds 200
}
```

Script calling fake tools being ran:

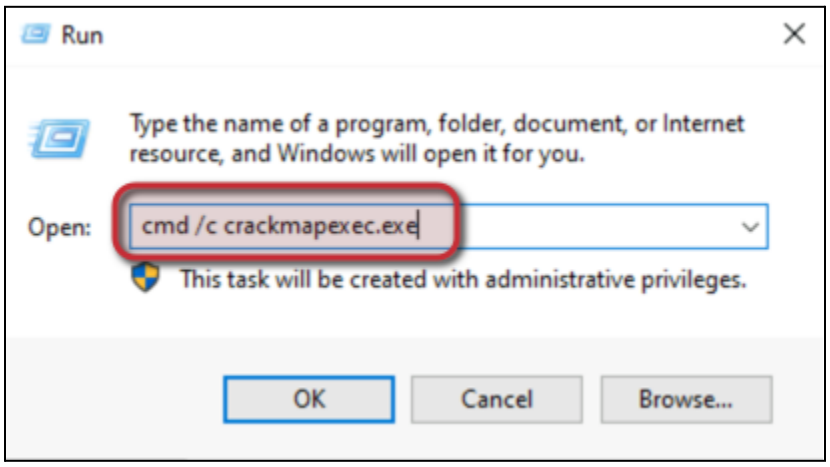


Also, cmd.exe and explorer.exe were used to call several tools just to make sure those methods are being detected as well:

Powershell Empire being called using Explorer.exe:



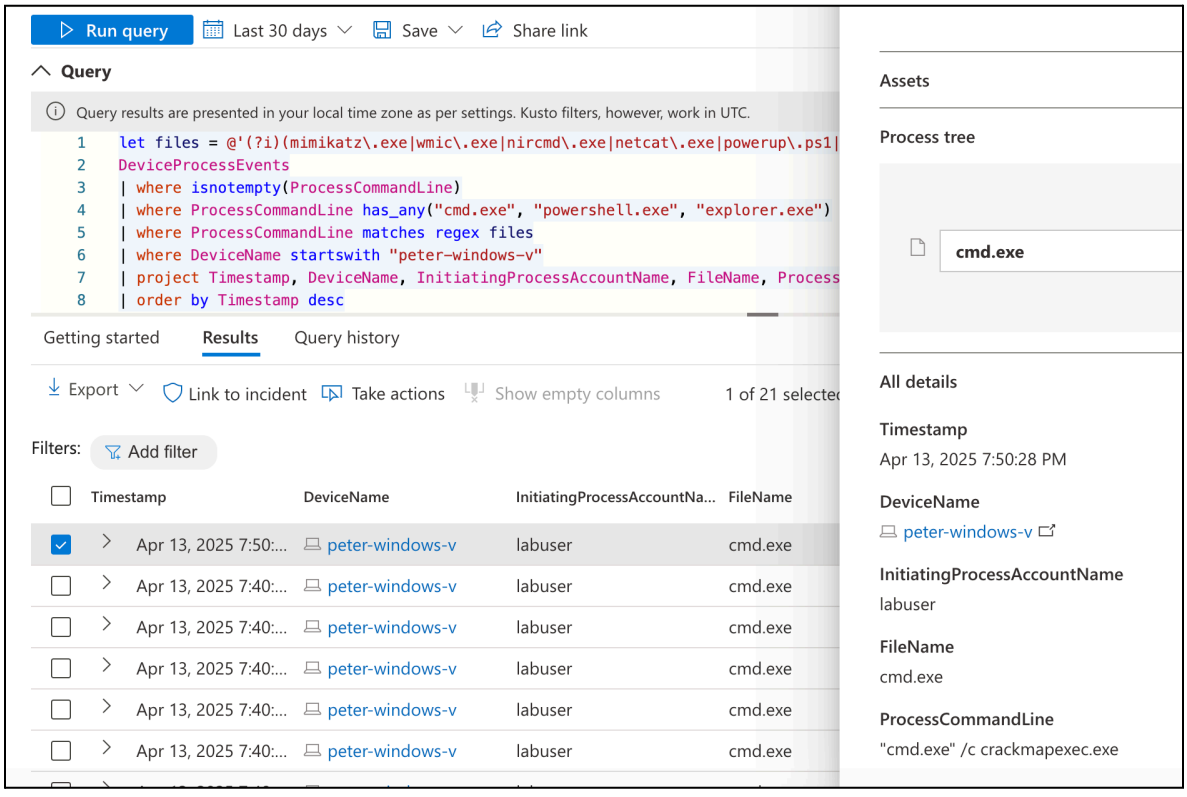
CrackMapExec being called using cmd.exe:

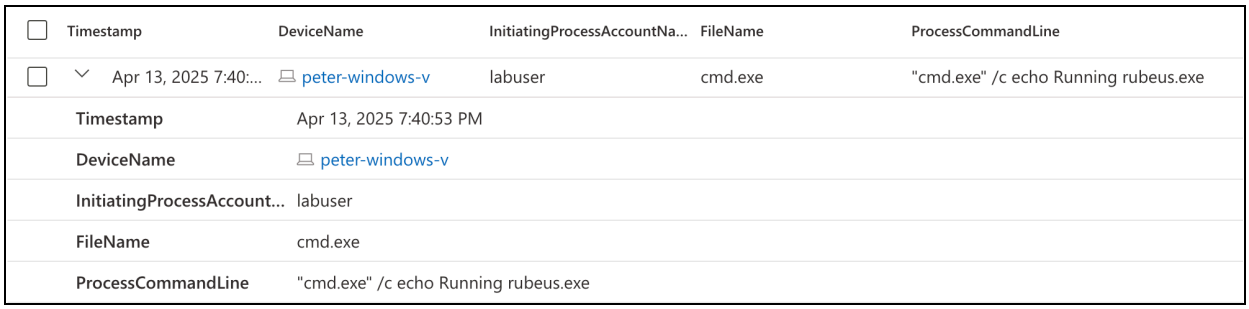


The query was successful, and a basic sorting function was added to the query:

```
let files =
@'(?i)(mimikatz\.exe|wmic\.exe|nircmd\.exe|netcat\.exe|powerup\.ps1|powershell
empire\.ps1|sharphound\.exe|seimpersonateprivilege\.exe|dirtycow\.exe|procdump\.e
xe|sharpdesk\.exe|psexec\.exe|incognito\.exe|add-localadmin\.ps1|chisel\.exe|jmsp
\.exe|tokenmanipulation\.ps1|meterpreter\.exe|winpeas\.exe|rubeus\.exe|lazagne\.e
xe|sherlock\.ps1|pstools\.exe|getadmin\.ps1|kalilinux\.exe|hackersploit\.exe|resp
onder\.exe|eternalblue\.exe|crackmapexec\.exe|netcat64\.exe|hashdump\.ps1|impacke
t\.py|powersploit\.ps1|zerologon\.exe|windows-exploit-suggester\.exe|dcomignite\.
exe|bypassuac\.exe|powershellwebshell\.ps1|kerberosbrute\.exe|beroot\.ps1)';
DeviceProcessEvents
| where isnotempty(ProcessCommandLine)
| where ProcessCommandLine has_any("cmd.exe", "powershell.exe", "explorer.exe")
| where ProcessCommandLine matches regex files
| where DeviceName startswith "peter-windows-v"
| project Timestamp, DeviceName, InitiatingProcessAccountName, FileName,
ProcessCommandLine
| order by Timestamp desc
```

Example hits for this query:





Timestamp	DeviceName	InitiatingProcessAccountNa...	FileName	ProcessCommandLine
Apr 13, 2025 7:40:...	peter-windows-v	labuser	cmd.exe	"cmd.exe" /c echo Running rubeus.exe

Timestamp	Apr 13, 2025 7:40:53 PM
DeviceName	peter-windows-v
InitiatingProcessAccount...	labuser
FileName	cmd.exe
ProcessCommandLine	"cmd.exe" /c echo Running rubeus.exe

Apr 13, 2025 7:40:...	peter-windows-v	labuser	cmd.exe	"cmd.exe" /c echo Running powershell empire.ps1

Timestamp	Apr 13, 2025 7:40:47 PM
DeviceName	peter-windows-v
InitiatingProcessAccount...	labuser
FileName	cmd.exe
ProcessCommandLine	"cmd.exe" /c echo Running powershell empire.ps1

Based on the identified KQL results, there were numerous tools being called by cmd.exe, powershell.exe, and explorer.exe on the 'peter-windows-v' VM. The above events took place on April 13, 2025, from 7:40 PM to 7:50 PM, resulting in 40 attempted tools being ran on the host. Used by a real attacker, any of these tools being ran would indicate that an attack had access to the file system/CLI, and is openly attempting Post-Exploitation or Privilege Escalation on the device.

Chronological Events

The events in this threat hunt were meant to simulate an attacker using Post-Exploitation/Privesc tools, so in a real-world scenario we might only see one or several or these tools being used. Also, the scenario relies on the attacker not renaming the script/executable (which a more savvy attacker or red-teamer would almost definitely do...). When the tool is transferred over using the command line, or downloaded from some repository, or built from Github, it could be named anything as an output. This hunt relies on it NOT being renamed, which could happen with a lazy attacker, pentester, or someone reflexively transferring and running a tool they rely on heavily.

The simulated chronology here is:

1. Attacker gains credentials or lands on `peter-windows-v`.

- With credentialed access, the attacker can log in and interact with the file system.

2. File(s) transferred to VM using any means

- Files could be copy/pasted, downloaded from a repository, coded, built, or transferred in using Powershell/CLI.

3. List of Post-Exploitation/Privesc Files called using powershell.exe, explorer.exe, or cmd.exe:

- **Timestamp:** `2025-04-14T02:40:45.5206135Z`
- **Event:** The user "labuser" executed powershell.exe using the ProcessCommandLine to call numerous tools.
- **Action:** Further investigation would be needed to determine if those tools actually ran, or if there is indeed malicious/offensive activity in the environment.
- **Scripts/Commands**:

Powershell:

```
$fakeTools = @(
    "mimikatz.exe", "wmic.exe", "nircmd.exe", "netcat.exe", "powerup.ps1",
"powershell empire.ps1",
    "sharphound.exe", "seimpersonateprivilege.exe", "dirtycow.exe",
"procdump.exe", "sharpdesk.exe",
    "psexec.exe", "incognito.exe", "add-localadmin.ps1", "chisel.exe",
"jmsp.exe", "tokenmanipulation.ps1",
    "meterpreter.exe", "winpeas.exe", "rubeus.exe", "lazagne.exe",
"sherlock.ps1", "pstools.exe",
    "getadmin.ps1", "kalilinux.exe", "hackersploit.exe", "responder.exe",
"eternalblue.exe",
    "crackmapexec.exe", "netcat64.exe", "hashdump.ps1", "impacket.py",
"powersploit.ps1",
    "zerologon.exe", "windows-exploit-suggester.exe", "dcomignite.exe",
"bypassuac.exe",
    "powershellwebshell.ps1", "kerberosbrute.exe", "beroot.ps1"
)

foreach ($tool in $fakeTools) {
    Write-Host "Simulating command line for: $tool"
    Start-Process -FilePath "cmd.exe" -ArgumentList "/c echo Running $tool"
-WindowStyle Hidden
    Start-Sleep -Milliseconds 200
}
```

cmd.exe:

```
cmd.exe /c netcat.exe
cmd.exe /c powershell empire.ps1
cmd.exe /c hacktools\winpeas.exe
```

- **Result:** Due to the commands that were ran, an attacker would potentially have enumerated the VM or even our environment using automated scripts/tools. This could be very problematic for our defensive posture if this activity is able to occur.

Summary

An Post-Exploitation/Privilege Escalation scenario was simulated, where a threat actor was able to access `peter-windows-v`, download scripts or tools, and run those tools on the VM. This hunt assumes that the tools were not renamed upon transfer to the VM, which they may not always be depending on the stealth/skill of the adversary. KQL was used to check for a list of ~40 well-known Windows Post-Exploitation/Privesc tools commonly used by penetration testers/ethical hackers, to see if any of those tools were ran using cmd.exe, powershell.exe, or explorer.exe - the main ways to run files in Windows. This KQL can be turned into an alert, so that additional tools or scripts can be detected (assuming they are not renamed), to detect lazy attackers who have already accessed our system as their target, and are working on exploiting the machine further. Seeing this activity on the device requires further analysis, password change, and possible DFIR to understand the attackers actions.

Response Taken

Known Post-Exploitation scripts/executables were found to have been called from powershell.exe, cmd.exe, and explorer.exe. The KQL used in our alert sounded the alarm to the SOC team, who isolated the VM, downloaded an analysis/investigative package, blocked the malicious IP, and changed the VM password - also enabling 2FA to prevent future incidents. The alert will be tuned to include additional Post-Exploitation tools.