Pentesting Lab

Privilege Escalation - Windows

Ostermayer, Possegger, Pongratz, Schauklies, Schwarzl

13.04.2026

Summer 2026, www.isec.tugraz.at/ptl

Table of contents

Introduction

What Is Privilege Escalation?

- "Privilege Escalation consists of techniques that adversaries use to **gain higher-level permissions** on a system or network." - *MITRE ATT&CK*
- "Privilege Escalation is the act of exploiting a bug, a design flaw, or a configuration oversight in an operating system or software application to **gain elevated access** to resources that are normally protected from an application or user." - *Wikipedia*
- "Privilege Escalation is the process of **gaining** unauthorized access to **higher-level permissions** or privileges within a system or network." - *ChatGPT*

What Is Privilege Escalation?

- "Privilege Escalation consists of techniques that adversaries use to **gain higher-level permissions** on a system or network." - *MITRE ATT&CK*
- "Privilege Escalation is the act of exploiting a bug, a design flaw, or a configuration oversight in an operating system or software application to **gain elevated access** to resources that are normally protected from an application or user." - *Wikipedia*
- "Privilege Escalation is the process of **gaining** unauthorized access to **higher-level permissions** or privileges within a system or network." - *ChatGPT*

What Is Privilege Escalation?

- "Privilege Escalation consists of techniques that adversaries use to **gain higher-level permissions** on a system or network." - *MITRE ATT&CK*
- "Privilege Escalation is the act of exploiting a bug, a design flaw, or a configuration oversight in an operating system or software application to **gain elevated access** to resources that are normally protected from an application or user." - *Wikipedia*
- "Privilege Escalation is the process of **gaining** unauthorized access to **higher-level permissions** or privileges within a system or network." - *ChatGPT*

Why Do We Want to Escalate Privileges?

Core Objectives:

- **Persistence:** Maintaining access even after reboots or password resets.
- **Credential Dumping:** Extracting hashes (LSASS, SAM) for further attacks.
- **Lateral Movement:** Pivoting to other machines in the network.
- **Impact:** Proving the full extent of the vulnerability to the client.

Why Do We Want to Escalate Privileges?

Core Objectives:

- **Persistence:** Maintaining access even after reboots or password resets.
- **Credential Dumping:** Extracting hashes (LSASS, SAM) for further attacks.
- **Lateral Movement:** Pivoting to other machines in the network.
- **Impact:** Proving the full extent of the vulnerability to the client.

Why Do We Want to Escalate Privileges?

Core Objectives:

- **Persistence:** Maintaining access even after reboots or password resets.
- **Credential Dumping:** Extracting hashes (LSASS, SAM) for further attacks.
- **Lateral Movement:** Pivoting to other machines in the network.
- **Impact:** Proving the full extent of the vulnerability to the client.

Why Do We Want to Escalate Privileges?

Core Objectives:

- **Persistence:** Maintaining access even after reboots or password resets.
- **Credential Dumping:** Extracting hashes (LSASS, SAM) for further attacks.
- **Lateral Movement:** Pivoting to other machines in the network.
- **Impact:** Proving the full extent of the vulnerability to the client.

Basics of the Windows Security Model

Access Token: Describes the security context of a user.

- **Owner Identity:** Contains User SID and Group SIDs. (Security Identifiers)
- **Privileges:** A list of specific rights.
- **Storage**: Each process holds a copy of the access token of owner in memory.
- **Inheritance:** Every child process inherits a copy of the creator's token.

Access Token: Describes the security context of a user.

- **Owner Identity:** Contains User SID and Group SIDs. (Security Identifiers)
- **Privileges:** A list of specific rights.
- **Storage**: Each process holds a copy of the access token of owner in memory.
- **Inheritance:** Every child process inherits a copy of the creator's token.

Access Token: Describes the security context of a user.

- **Owner Identity:** Contains User SID and Group SIDs. (Security Identifiers)
- **Privileges:** A list of specific rights.
- **Storage**: Each process holds a copy of the access token of owner in memory.
- **Inheritance:** Every child process inherits a copy of the creator's token.

Access Token: Describes the security context of a user.

- **Owner Identity:** Contains User SID and Group SIDs. (Security Identifiers)
- **Privileges:** A list of specific rights.
- **Storage**: Each process holds a copy of the access token of owner in memory.
- **Inheritance:** Every child process inherits a copy of the creator's token.

Administrator Split-Token Model

Administrator Tokens: Windows assigns **two** different access tokens to users in the Administrators group upon login.

- **Filtered Token (Standard):** Normal user tasks with normal privileges (Medium Integrity).
- **Elevated Token (Admin):** Administrative tasks with high privileges (High Integrity).
- **UAC:** Lets admin explicitly approve the shift from Filtered to Elevated.

Administrator Tokens: Windows assigns **two** different access tokens to users in the Administrators group upon login.

- **Filtered Token (Standard):** Normal user tasks with normal privileges (Medium Integrity).
- **Elevated Token (Admin):** Administrative tasks with high privileges (High Integrity).
- **UAC:** Lets admin explicitly approve the shift from Filtered to Elevated.

Administrator Tokens: Windows assigns **two** different access tokens to users in the Administrators group upon login.

- **Filtered Token (Standard):** Normal user tasks with normal privileges (Medium Integrity).
- **Elevated Token (Admin):** Administrative tasks with high privileges (High Integrity).
- **UAC:** Lets admin explicitly approve the shift from Filtered to Elevated.

Mechanism:

- Prompts Administrator to allow elevated activities.
- Configurable at different security levels.
- Processes run with **User** rights by default until elevated.

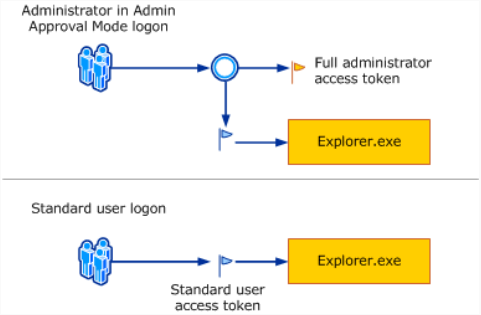


Image-Source

Mechanism:

- Prompts Administrator to allow elevated activities.
- Configurable at different security levels.
- Processes run with **User** rights by default until elevated.

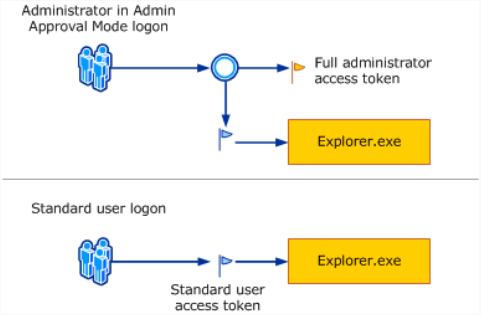


Image-Source

Mechanism:

- Prompts Administrator to allow elevated activities.
- Configurable at different security levels.
- Processes run with **User** rights by default until elevated.

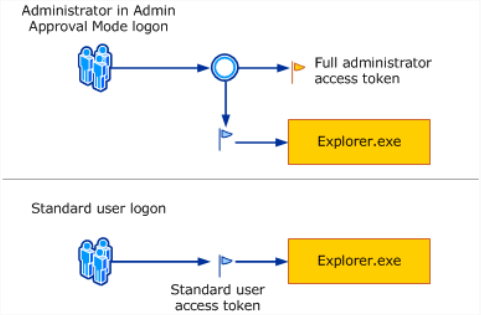


Image-Source

UAC - Consent Prompt

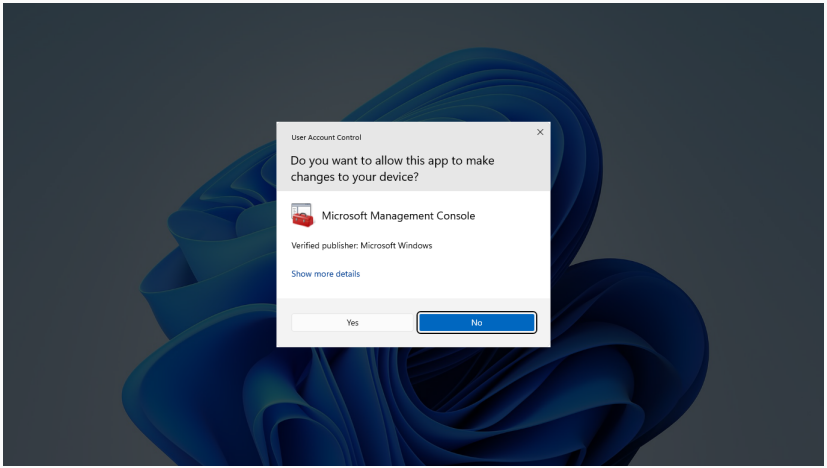


Image-Source

UAC - Credential prompt

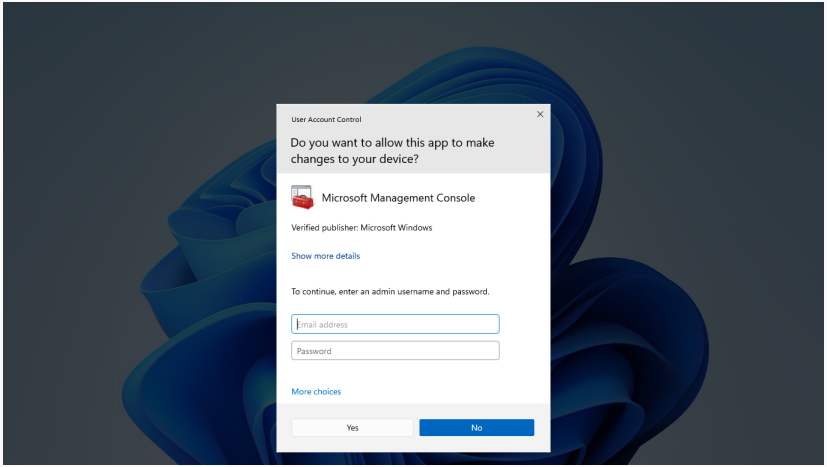


Image-Source

Scenario: If user is in the **Administrators group** but has a Medium Integrity token.

- **Auto-Elevating Binaries:** Some Windows binaries (e.g., `fodhelper.exe`) bypass UAC automatically if signed by Microsoft.
- **COM Hijacking:** Manipulating Component Object Model registry keys to intercept a high-integrity process call.
- **Tooling: UAC-ME** is the definitive collection of 70+ bypass methods.

Scenario: If user is in the **Administrators group** but has a Medium Integrity token.

- **Auto-Elevating Binaries:** Some Windows binaries (e.g., `fodhelper.exe`) bypass UAC automatically if signed by Microsoft.
- **COM Hijacking:** Manipulating Component Object Model registry keys to intercept a high-integrity process call.
- **Tooling: UAC-ME** is the definitive collection of 70+ bypass methods.

Scenario: If user is in the **Administrators group** but has a Medium Integrity token.

- **Auto-Elevating Binaries:** Some Windows binaries (e.g., `fodhelper.exe`) bypass UAC automatically if signed by Microsoft.
- **COM Hijacking:** Manipulating Component Object Model registry keys to intercept a high-integrity process call.
- **Tooling: UAC-ME** is the definitive collection of 70+ bypass methods.

High-Level Accounts:

- `Administrator`
  - You can do **almost** everything on the OS.
- `NT AUTHORITY\SYSTEM`
  - The highest operational level for the Windows kernel.
  - Equivalent to root on Unix.

High-Level Accounts:

- `Administrator`
  - You can do **almost** everything on the OS.
- `NT AUTHORITY\SYSTEM`
  - The highest operational level for the Windows kernel.
  - Equivalent to root on Unix.

High-Level Accounts:

- `Administrator`
  - You can do **almost** everything on the OS.
- `NT AUTHORITY\SYSTEM`
  - The highest operational level for the Windows kernel.
  - Equivalent to root on Unix.

High-Level Accounts:

- `Administrator`
  - You can do **almost** everything on the OS.
- `NT AUTHORITY\SYSTEM`
  - The highest operational level for the Windows kernel.
  - Equivalent to root on Unix.

High-Level Accounts:

- `Administrator`
  - You can do **almost** everything on the OS.
- `NT AUTHORITY\SYSTEM`
  - The highest operational level for the Windows kernel.
  - Equivalent to root on Unix.

Built-in Services: Beyond the standard Administrator, Windows uses specific service accounts:

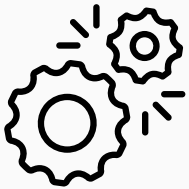
- `LocalService`: Minimum privileges, acts as anonymous on the network.
- `NetworkService`: Limited privileges, but uses the **computer's credentials** on the network.
- Why we care: `LocalService` and `NetworkService` often possess **SeImpersonatePrivilege**, making them prime targets for Potato exploits (see later).

Built-in Services: Beyond the standard Administrator, Windows uses specific service accounts:

- `LocalService`: Minimum privileges, acts as anonymous on the network.
- `NetworkService`: Limited privileges, but uses the **computer's credentials** on the network.
- Why we care: `LocalService` and `NetworkService` often possess **SeImpersonatePrivilege**, making them prime targets for Potato exploits (see later).

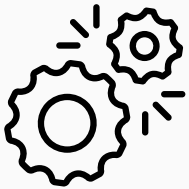
Built-in Services: Beyond the standard Administrator, Windows uses specific service accounts:

- `LocalService`: Minimum privileges, acts as anonymous on the network.
- `NetworkService`: Limited privileges, but uses the **computer's credentials** on the network.
- Why we care: `LocalService` and `NetworkService` often possess **SeImpersonatePrivilege**, making them prime targets for Potato exploits (see later).



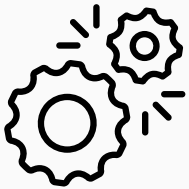
Integrity Level: Define the trust of process and prevent modification of higher trust objects.

- **Untrusted:** Anonymous logins. Highly restricted.
- **Low:** Internet Applications (e.g., Browsers). No registry write access.
- **Medium:** Default level for standard users and unelevated Admins.
- **High:** Elevated Administrators. Can modify lower/same levels.
- **System:** Kernel and core services. Our goal.
- **Installer:** Can uninstall any object (highest level).



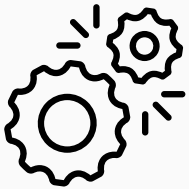
Integrity Level: Define the trust of process and prevent modification of higher trust objects.

- **Untrusted:** Anonymous logins. Highly restricted.
- **Low:** Internet Applications (e.g., Browsers). No registry write access.
- **Medium:** Default level for standard users and unelevated Admins.
- **High:** Elevated Administrators. Can modify lower/same levels.
- **System:** Kernel and core services. Our goal.
- **Installer:** Can uninstall any object (highest level).



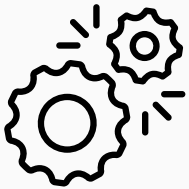
Integrity Level: Define the trust of process and prevent modification of higher trust objects.

- **Untrusted:** Anonymous logins. Highly restricted.
- **Low:** Internet Applications (e.g., Browsers). No registry write access.
- **Medium:** Default level for standard users and unelevated Admins.
- **High:** Elevated Administrators. Can modify lower/same levels.
- **System:** Kernel and core services. Our goal.
- **Installer:** Can uninstall any object (highest level).



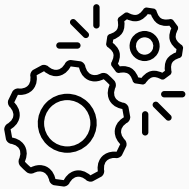
Integrity Level: Define the trust of process and prevent modification of higher trust objects.

- **Untrusted:** Anonymous logins. Highly restricted.
- **Low:** Internet Applications (e.g., Browsers). No registry write access.
- **Medium:** Default level for standard users and unelevated Admins.
- **High:** Elevated Administrators. Can modify lower/same levels.
- **System:** Kernel and core services. Our goal.
- **Installer:** Can uninstall any object (highest level).



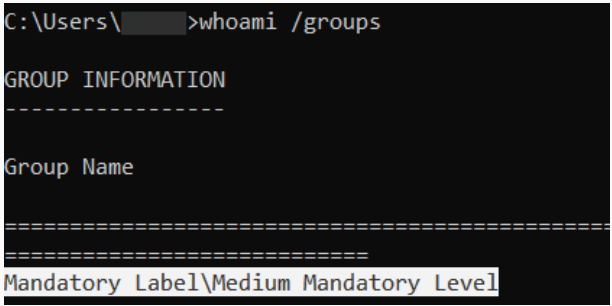
Integrity Level: Define the trust of process and prevent modification of higher trust objects.

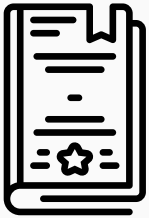
- **Untrusted:** Anonymous logins. Highly restricted.
- **Low:** Internet Applications (e.g., Browsers). No registry write access.
- **Medium:** Default level for standard users and unelevated Admins.
- **High:** Elevated Administrators. Can modify lower/same levels.
- **System:** Kernel and core services. Our goal.
- **Installer:** Can uninstall any object (highest level).



Integrity Level: Define the trust of process and prevent modification of higher trust objects.

- **Untrusted:** Anonymous logins. Highly restricted.
- **Low:** Internet Applications (e.g., Browsers). No registry write access.
- **Medium:** Default level for standard users and unelevated Admins.
- **High:** Elevated Administrators. Can modify lower/same levels.
- **System:** Kernel and core services. Our goal.
- **Installer:** Can uninstall any object (highest level).


C:\Users\ >whoami /groups
GROUP INFORMATION
Group Name
Mandatory Label\Medium Mandatory Level



ACL: A list of ACEs (Access Control Entities) dictating file/folder permissions.

- **Tooling:** View ACEs using `icacls.exe`
- **Key Attributes:**
  - **(I)** Inherited — **(F)** Full Control
  - **(M)** Modify — **(D)** Delete
  - **(RX)** Read & Execute — **(W)** Write

```
PS> icacls C:\Users\User\Desktop\file.txt
C:\Users\User\Desktop\file.txt BUILTIN\Administrators:(I)(F)
                              NT AUTHORITY\SYSTEM:(I)(F)
                              BUILTIN\Users:(I)(RX)
                              Mandatory Label\High Mandatory Level:(NW)
```

Two Types of ACLs:

- **DACL (Discretionary Access Control List)**
  - Specifies which users/groups **have or do not have** access to an object.
  - Contains ACEs that grant or deny access permissions.
- **SACL (System Access Control List)**
  - Used for **auditing** and logging.
  - ACEs define the type of access attempts to be logged in the Security Event Log.

Two Types of ACLs:

- **DACL (Discretionary Access Control List)**
  - Specifies which users/groups **have or do not have** access to an object.
  - Contains ACEs that grant or deny access permissions.
- **SACL (System Access Control List)**
  - Used for **auditing** and logging.
  - ACEs define the type of access attempts to be logged in the Security Event Log.

Two Types of ACLs:

- **DACL (Discretionary Access Control List)**
  - Specifies which users/groups **have or do not have** access to an object.
  - Contains ACEs that grant or deny access permissions.
- **SACL (System Access Control List)**
  - Used for **auditing** and logging.
  - ACEs define the type of access attempts to be logged in the Security Event Log.

Two Types of ACLs:

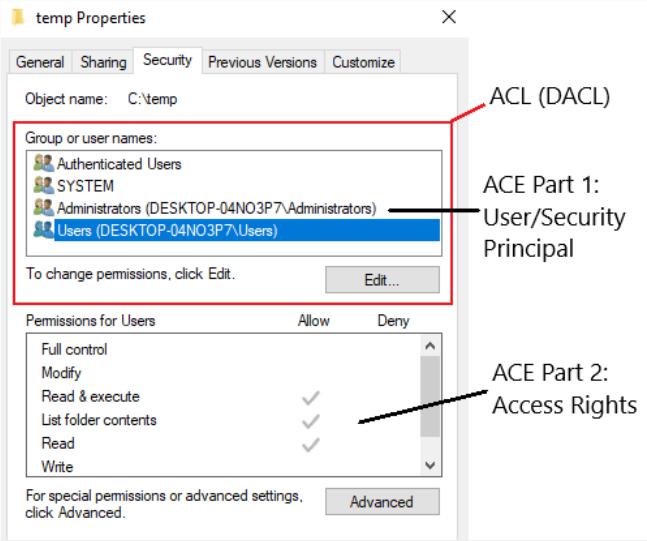
- **DACL (Discretionary Access Control List)**
  - Specifies which users/groups **have or do not have** access to an object.
  - Contains ACEs that grant or deny access permissions.
- **SACL (System Access Control List)**
  - Used for **auditing** and logging.
  - ACEs define the type of access attempts to be logged in the Security Event Log.

Two Types of ACLs:

- **DACL (Discretionary Access Control List)**
  - Specifies which users/groups **have or do not have** access to an object.
  - Contains ACEs that grant or deny access permissions.
- **SACL (System Access Control List)**
  - Used for **auditing** and logging.
  - ACEs define the type of access attempts to be logged in the Security Event Log.

Two Types of ACLs:

- **DACL (Discretionary Access Control List)**
  - Specifies which users/groups **have or do not have** access to an object.
  - Contains ACEs that grant or deny access permissions.
- **SACL (System Access Control List)**
  - Used for **auditing** and logging.
  - ACEs define the type of access attempts to be logged in the Security Event Log.


temp Properties
General
Sharing
Security
Previous Versions
Customize
Object name: C:\temp
Group or user names:
Authenticated Users
SYSTEM
Administrators (DESKTOP-04NO3P7\Administrators)
Users (DESKTOP-04NO3P7\Users)
To change permissions, click Edit.
Edit...
Permissions for Users
Allow
Deny
Full control
Modify
Read & execute
List folder contents
Read
Write
For special permissions or advanced settings, click Advanced.
Advanced
ACL (DACL)
ACE Part 1: User/Security Principal
ACE Part 2: Access Rights

Where to find the keys to the kingdom:

- **SAM & SYSTEM:** Stores cached hashed credentials on disk/registry: `C:\Windows\System32\config\sam` & `HKLM\SAM`.
- **LSA (Local Security Authority):** Stores credentials in memory `lsass.exe`.
- **NTDS.dit:** Database of Active Directory (Domain Controllers only - See AD Lecture).
- **LAPS:** Solution to manage local admin passwords in a domain.

Where to find the keys to the kingdom:

- **SAM & SYSTEM:** Stores cached hashed credentials on disk/registry: `C:\Windows\System32\config\sam` & `HKLM\SAM`.
- **LSA (Local Security Authority):** Stores credentials in memory `lsass.exe`.
- **NTDS.dit:** Database of Active Directory (Domain Controllers only - See AD Lecture).
- **LAPS:** Solution to manage local admin passwords in a domain.

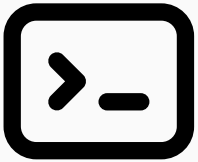
Where to find the keys to the kingdom:

- **SAM & SYSTEM:** Stores cached hashed credentials on disk/registry: `C:\Windows\System32\config\sam` & `HKLM\SAM`.
- **LSA (Local Security Authority):** Stores credentials in memory `lsass.exe`.
- **NTDS.dit:** Database of Active Directory (Domain Controllers only - See AD Lecture).
- **LAPS:** Solution to manage local admin passwords in a domain.

Where to find the keys to the kingdom:

- **SAM & SYSTEM:** Stores cached hashed credentials on disk/registry: `C:\Windows\System32\config\sam` & `HKLM\SAM`.
- **LSA (Local Security Authority):** Stores credentials in memory `lsass.exe`.
- **NTDS.dit:** Database of Active Directory (Domain Controllers only - See AD Lecture).
- **LAPS:** Solution to manage local admin passwords in a domain.

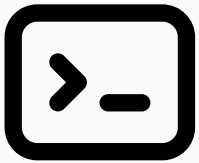
DLL (Dynamically Linked Library)



Concept: Equivalent to Shared Libraries (.so) on Unix systems.

- **Strong Reference:** Application calls the exact path (e.g., `C:\path\lib.dll`).
- **Weak Reference:** Application only calls the filename (e.g., `lib.dll`).
  - Vulnerability: Windows attempts to find the weakly referenced DLL through a **pre-defined search order**.

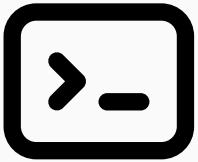
DLL (Dynamically Linked Library)



Concept: Equivalent to Shared Libraries (.so) on Unix systems.

- **Strong Reference:** Application calls the exact path (e.g., `C:\path\lib.dll`).
- **Weak Reference:** Application only calls the filename (e.g., `lib.dll`).
  - Vulnerability: Windows attempts to find the weakly referenced DLL through a **pre-defined search order**.

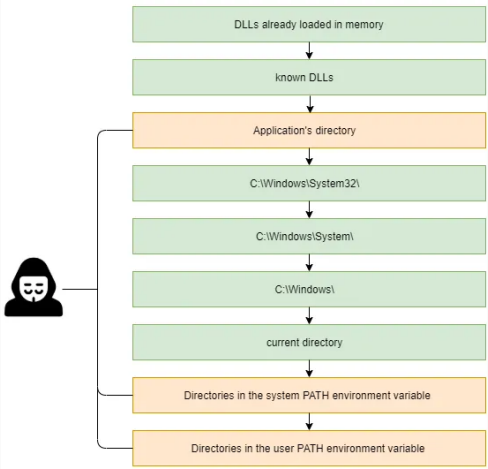
DLL (Dynamically Linked Library)



Concept: Equivalent to Shared Libraries (.so) on Unix systems.

- **Strong Reference:** Application calls the exact path (e.g., `C:\path\lib.dll`).
- **Weak Reference:** Application only calls the filename (e.g., `lib.dll`).
  - Vulnerability: Windows attempts to find the weakly referenced DLL through a **pre-defined search order**.

DLL Search Order



https://sid4hack.medium.com/malware-development-part-6-dll-hijacking-70bd4611a105

Common Vulnerabilities

Vulnerability: Passwords accessible when they shouldn't

- Plaintext credentials saved in the Registry (e.g., AutoLogon).
- Saved credentials in text files (`passwords.txt`, `config.ini`).
- Readable SAM & SYSTEM database backups (`Repair` folder).
- PowerShell history files (`ConsoleHost_history.txt`).
- Unattended installation files (`unattend.xml`, `sysprep.xml`).

Vulnerability: Passwords accessible when they shouldn't

- Plaintext credentials saved in the Registry (e.g., AutoLogon).
- Saved credentials in text files (`passwords.txt`, `config.ini`).
- Readable SAM & SYSTEM database backups (`Repair` folder).
- PowerShell history files (`ConsoleHost_history.txt`).
- Unattended installation files (`unattend.xml`, `sysprep.xml`).

Vulnerability: Passwords accessible when they shouldn't

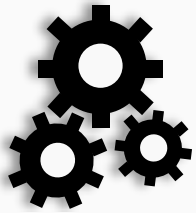
- Plaintext credentials saved in the Registry (e.g., AutoLogon).
- Saved credentials in text files (`passwords.txt`, `config.ini`).
- Readable SAM & SYSTEM database backups (`Repair` folder).
- PowerShell history files (`ConsoleHost_history.txt`).
- Unattended installation files (`unattend.xml`, `sysprep.xml`).

Vulnerability: Passwords accessible when they shouldn't

- Plaintext credentials saved in the Registry (e.g., AutoLogon).
- Saved credentials in text files (`passwords.txt`, `config.ini`).
- Readable SAM & SYSTEM database backups (`Repair` folder).
- PowerShell history files (`ConsoleHost_history.txt`).
- Unattended installation files (`unattend.xml`, `sysprep.xml`).

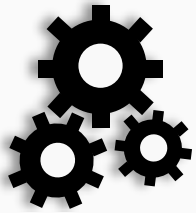
Vulnerability: Passwords accessible when they shouldn't

- Plaintext credentials saved in the Registry (e.g., AutoLogon).
- Saved credentials in text files (`passwords.txt`, `config.ini`).
- Readable SAM & SYSTEM database backups (`Repair` folder).
- PowerShell history files (`ConsoleHost_history.txt`).
- Unattended installation files (`unattend.xml`, `sysprep.xml`).



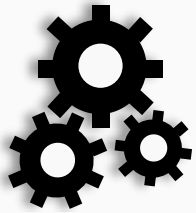
Vulnerability: Weak DACLs on the Service configuration.

- **Concept:** If our user has `SERVICE_CHANGE_CONFIG` privileges, we can modify the service properties.
- **Attack:** Change the `binPath` to point to our malicious payload, then restart the service.
- **Tooling:** Check privileges using `accesschk.exe` and modify using `sc.exe`.



Vulnerability: Weak DACLs on the Service configuration.

- **Concept:** If our user has `SERVICE_CHANGE_CONFIG` privileges, we can modify the service properties.
- **Attack:** Change the `binPath` to point to our malicious payload, then restart the service.
- **Tooling:** Check privileges using `accesschk.exe` and modify using `sc.exe`.



Vulnerability: Weak DACLs on the Service configuration.

- **Concept:** If our user has `SERVICE_CHANGE_CONFIG` privileges, we can modify the service properties.
- **Attack:** Change the `binPath` to point to our malicious payload, then restart the service.
- **Tooling:** Check privileges using `accesschk.exe` and modify using `sc.exe`.

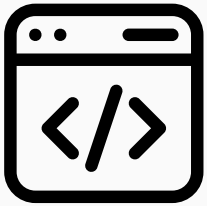
Insecure Service Permissions II

```
C:\Users\pentestlab>accesschk.exe -uwcqv "pentestlab" * -accepteula
accesschk.exe -uwcqv "pentestlab" * -accepteula

Accesschk v6.10 - Reports effective permissions for securable objects
Copyright (C) 2006-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

RW Apache
        SERVICE_ALL_ACCESS

C:\Users\pentestlab>
```

https://pentestlab.blog/2017/03/30/weak-service-permissions/

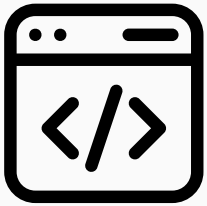
Insecure Service Permissions III

```
C:\Users\pentestlab>sc qc Apache
sc qc Apache
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Apache
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\xampp\apache\bin\httpd.exe" -k runservice
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Apache
        DEPENDENCIES       : Tcpip
                           : Afd
        SERVICE_START_NAME : LocalSystem

C:\Users\pentestlab>sc config "Apache" binPath= "net localgroup administrators p
entestlab /add"
sc config "Apache" binPath= "net localgroup administrators pentestlab /add"
[SC] ChangeServiceConfig SUCCESS
```



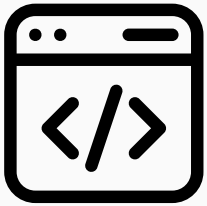
Vulnerability: Weak permissions on the actual executable file.

- **Concept:** The service config is locked down, but the binary itself (`*.exe`) is writable by us.
- **Attack:** Overwrite the original executable with a malicious one. Wait for the service to start/restart.
- **Tooling:** Use `icacls.exe` to check file permissions.



Vulnerability: Weak permissions on the actual executable file.

- **Concept:** The service config is locked down, but the binary itself (`*.exe`) is writable by us.
- **Attack:** Overwrite the original executable with a malicious one. Wait for the service to start/restart.
- **Tooling:** Use `icacls.exe` to check file permissions.



Vulnerability: Weak permissions on the actual executable file.

- **Concept:** The service config is locked down, but the binary itself (`*.exe`) is writable by us.
- **Attack:** Overwrite the original executable with a malicious one. Wait for the service to start/restart.
- **Tooling:** Use `icacls.exe` to check file permissions.

Insecure Service Executables II

```
C:\Users\pentestlab>sc qc Apache
sc qc Apache
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Apache
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : "C:\xampp\apache\bin\httpd.exe" -k runservice
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Apache
        DEPENDENCIES       : Tcpip
                           : Afd
        SERVICE_START_NAME : LocalSystem
```

https://pentestlab.blog/2017/03/30/weak-service-permissions/

Vulnerability: Windows interprets spaces in unquoted paths as delimiters.

- **Example Path:** `C:\Program Files\Inventory Service\inventory svc.exe`
- **Attack:** Windows executes binaries in this specific order:
  1. `C:\Program.exe`
  2. `C:\Program Files\Inventory.exe`
  3. `C:\Program Files\Inventory Service\inventory.exe`

Vulnerability: Windows interprets spaces in unquoted paths as delimiters.

- **Example Path:** `C:\Program Files\Inventory Service\inventory svc.exe`
- **Attack:** Windows executes binaries in this specific order:
  1. `C:\Program.exe`
  2. `C:\Program Files\Inventory.exe`
  3. `C:\Program Files\Inventory Service\inventory.exe`

Vulnerability: Windows interprets spaces in unquoted paths as delimiters.

- **Example Path:** `C:\Program Files\Inventory Service\inventory svc.exe`
- **Attack:** Windows executes binaries in this specific order:
  1. `C:\Program.exe`
  2. `C:\Program Files\Inventory.exe`
  3. `C:\Program Files\Inventory Service\inventory.exe`

—

Requirement: *You must have* **Write** *access to one of the parent directories to plant the fake binary.*

Vulnerability: Windows interprets spaces in unquoted paths as delimiters.

- **Example Path:** `C:\Program Files\Inventory Service\inventory svc.exe`
- **Attack:** Windows executes binaries in this specific order:
  1. `C:\Program.exe`
  2. `C:\Program Files\Inventory.exe`
  3. `C:\Program Files\Inventory Service\inventory.exe`

—

Requirement: *You must have* **Write** *access to one of the parent directories to plant the fake binary.*

Vulnerability: Windows interprets spaces in unquoted paths as delimiters.

- **Example Path:** `C:\Program Files\Inventory Service\inventory svc.exe`
- **Attack:** Windows executes binaries in this specific order:
  1. `C:\Program.exe`
  2. `C:\Program Files\Inventory.exe`
  3. `C:\Program Files\Inventory Service\inventory.exe`

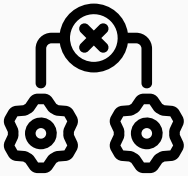
—

Requirement: *You must have* **Write** *access to one of the parent directories to plant the fake binary.*

Unquoted Service Paths II

```
PS C:\Users\User> Get-ServiceUnquoted


ServiceName    : GDCAgent
Path           : C:\Program Files (x86)\Lenovo\GDCAgent.exe
ModifiablePath : @{Permissions=System.Object[]; ModifiablePath=C:\; IdentityReference=BUILTIN\Administrators}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'GDCAgent' -Path <HijackPath>
CanRestart     : True

ServiceName    : GDCAgent
Path           : C:\Program Files (x86)\Lenovo\GDCAgent.exe
ModifiablePath : @{Permissions=System.Object[]; ModifiablePath=C:\; IdentityReference=BUILTIN\Users}
StartName      : LocalSystem
AbuseFunction  : Write-ServiceBinary -Name 'GDCAgent' -Path <HijackPath>
CanRestart     : True
```

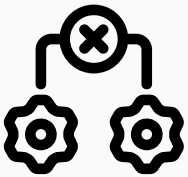
https://pentestlab.blog/2017/03/09/unquoted-service-path/



Vulnerability: Same as with Services.

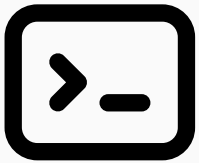
- Tasks often run as `SYSTEM` or high-privileged users.
- Search for tasks triggering modifiable binaries or scripts (.bat, .ps1).

```
# List all scheduled tasks
PS> schtasks /query /fo LIST /v
```



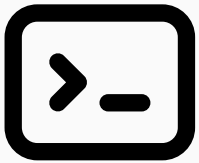
Vulnerability: High-privilege process calls a binary without an absolute path

- Program calls binary without absolute path (e.g., `net.exe`)...
- And a folder in the `%PATH%` is writable...
- A malicious binary can be placed earlier in the search order to hijack the call.



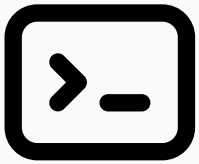
Vulnerability: Exploiting the Windows DLL Search Order.

- **Attack:** If an application is missing a DLL (or calls it weakly), and we have write access to a folder in the search path, we can supply a malicious DLL.
- **Automated Discovery:** Crassus or DLLHijackHunter.
- **Manual Discovery:** Use Process Monitor (ProcMon).
  - Requires Administrative Privileges in most cases.
  - - Download target executable to a local Windows VM.
  - - Analyze calls and filter for `NAME NOT FOUND`.



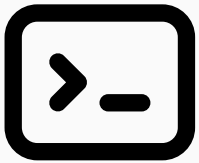
Vulnerability: Exploiting the Windows DLL Search Order.

- **Attack:** If an application is missing a DLL (or calls it weakly), and we have write access to a folder in the search path, we can supply a malicious DLL.
- **Automated Discovery:** Crassus or DLLHijackHunter.
- **Manual Discovery:** Use Process Monitor (ProcMon).
  - Requires Administrative Privileges in most cases.
  - - Download target executable to a local Windows VM.
  - - Analyze calls and filter for `NAME NOT FOUND`.



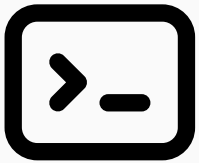
Vulnerability: Exploiting the Windows DLL Search Order.

- **Attack:** If an application is missing a DLL (or calls it weakly), and we have write access to a folder in the search path, we can supply a malicious DLL.
- **Automated Discovery:** Crassus or DLLHijackHunter.
- **Manual Discovery:** Use Process Monitor (ProcMon).
  - Requires Administrative Privileges in most cases.
  - - Download target executable to a local Windows VM.
  - - Analyze calls and filter for `NAME NOT FOUND`.



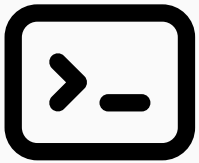
Vulnerability: Exploiting the Windows DLL Search Order.

- **Attack:** If an application is missing a DLL (or calls it weakly), and we have write access to a folder in the search path, we can supply a malicious DLL.
- **Automated Discovery:** Crassus or DLLHijackHunter.
- **Manual Discovery:** Use Process Monitor (ProcMon).
  - Requires Administrative Privileges in most cases.
  - - Download target executable to a local Windows VM.
  - - Analyze calls and filter for `NAME NOT FOUND`.



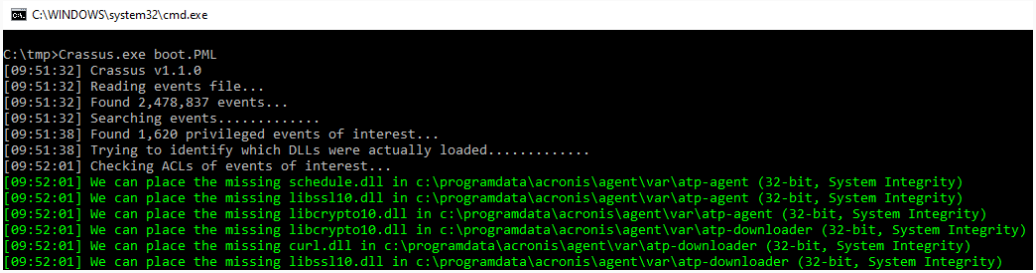
Vulnerability: Exploiting the Windows DLL Search Order.

- **Attack:** If an application is missing a DLL (or calls it weakly), and we have write access to a folder in the search path, we can supply a malicious DLL.
- **Automated Discovery:** Crassus or DLLHijackHunter.
- **Manual Discovery:** Use Process Monitor (ProcMon).
  - Requires Administrative Privileges in most cases.
  - - Download target executable to a local Windows VM.
  - - Analyze calls and filter for `NAME NOT FOUND`.

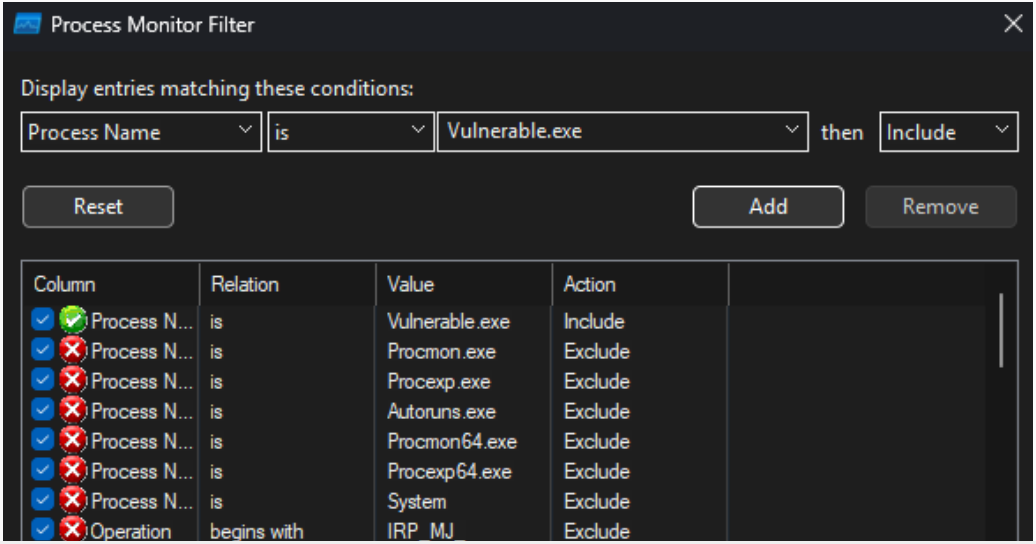


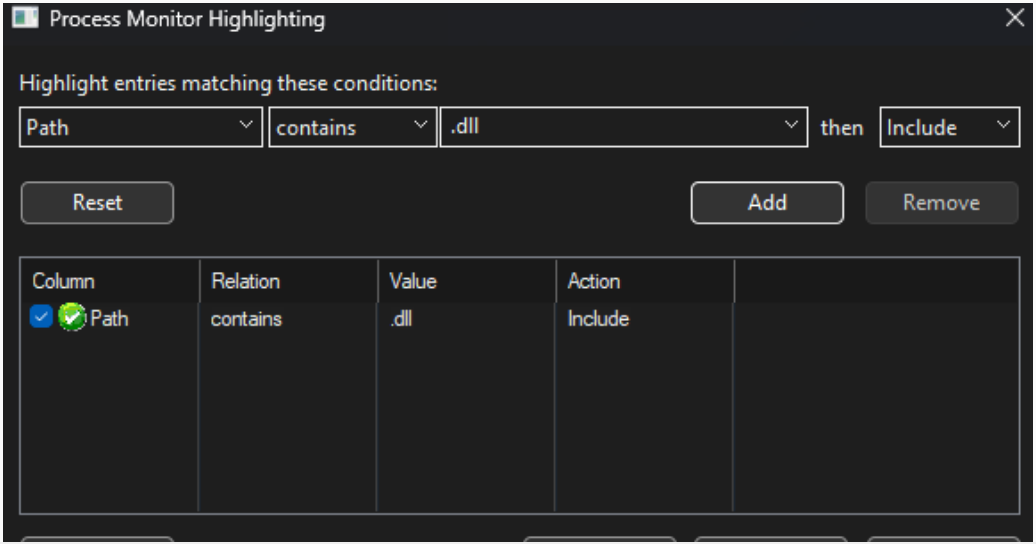
Vulnerability: Exploiting the Windows DLL Search Order.

- **Attack:** If an application is missing a DLL (or calls it weakly), and we have write access to a folder in the search path, we can supply a malicious DLL.
- **Automated Discovery:** Crassus or DLLHijackHunter.
- **Manual Discovery:** Use Process Monitor (ProcMon).
  - Requires Administrative Privileges in most cases.
  - - Download target executable to a local Windows VM.
  - - Analyze calls and filter for `NAME NOT FOUND`.



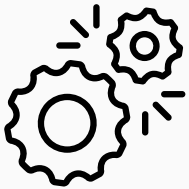
https://github.com/vu-ls/Crassus


Process Monitor Filter
Display entries matching these conditions:
Process Name
is
Vulnerable.exe
then
Include
Reset
Add
Remove
Column
Relation
Value
Action
Process N...
is
Vulnerable.exe
Include
Process N...
is
Procmon.exe
Exclude
Process N...
is
Procexp.exe
Exclude
Process N...
is
Autoruns.exe
Exclude
Process N...
is
Procmon64.exe
Exclude
Process N...
is
Procexp64.exe
Exclude
Process N...
is
System
Exclude
Operation
begins with
IRP_MJ_
Exclude


Process Monitor Highlighting
Highlight entries matching these conditions:
Path
contains
.dll
then
Include
Reset
Add
Remove
Column
Relation
Value
Action
Path
contains
.dll
Include

DLL Hijacking V - Missing Dlls

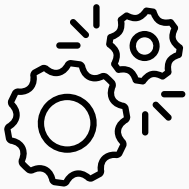
Time ...	Process Name	PID	Operation	Path	Result
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\System32\mscoree.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\System32\mscoree.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFileMapp...	C:\Windows\System32\mscoree.dll	FILE LOCKED WITH ONLY READERS
11:56:...	Vulnerable.exe	24032	CreateFileMapp...	C:\Windows\System32\mscoree.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\System32\mscoree.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\System32\conhost.exe	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFileMapp...	C:\Windows\System32\conhost.exe	FILE LOCKED WITH ONLY READERS
11:56:...	Vulnerable.exe	24032	CreateFileMapp...	C:\Windows\System32\conhost.exe	SUCCESS
11:56:...	Vulnerable.exe	24032	QuerySecurityFile	C:\Windows\System32\conhost.exe	SUCCESS
11:56:...	Vulnerable.exe	24032	QueryNameInfo...	C:\Windows\System32\conhost.exe	SUCCESS
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\System32\conhost.exe	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\System32\MSCOREE.DLL.local	NAME NOT FOUND
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319	SUCCESS
11:56:...	Vulnerable.exe	24032	QueryDirectory	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319	SUCCESS
11:56:...	Vulnerable.exe	24032	QueryDirectory	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	QueryBasicInfor...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFileMapp...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	FILE LOCKED WITH ONLY READERS
11:56:...	Vulnerable.exe	24032	CreateFileMapp...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CloseFile	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.dll	SUCCESS
11:56:...	Vulnerable.exe	24032	CreateFile	C:\Windows\System32\MSCOREE.DLL.local	NAME NOT FOUND



Vulnerability: "Bring Your Own Vulnerable Driver"

- Exploit officially **signed** but vulnerable drivers to execute code in kernel space.
- Reference List: https://www.loldrivers.io

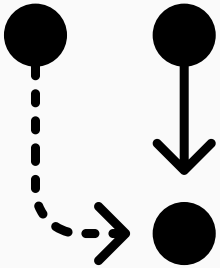
```
PS C:\Users\User> driverquery.exe /fo table /si
Module Name  Display Name              Driver Type  Link Date
===========  ========================  ===========  ======================
1394ohci     1394 OHCI Compliant Ho    Kernel       12/10/2006 4:44:38 PM
3ware        3ware                     Kernel       5/18/2015 6:28:03 PM
ACPI         Microsoft ACPI Driver     Kernel       12/9/1975 6:17:08 AM
<SNIP>
```



Vulnerability: "Bring Your Own Vulnerable Driver"

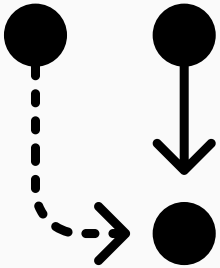
- Exploit officially **signed** but vulnerable drivers to execute code in kernel space.
- Reference List: https://www.loldrivers.io

```
PS C:\Users\User> driverquery.exe /fo table /si
Module Name  Display Name              Driver Type  Link Date
===========  ========================  ===========  ======================
1394ohci     1394 OHCI Compliant Ho    Kernel       12/10/2006 4:44:38 PM
3ware        3ware                     Kernel       5/18/2015 6:28:03 PM
ACPI         Microsoft ACPI Driver     Kernel       12/9/1975 6:17:08 AM
<SNIP>
```



Vulnerability: Blurring the lines between OS boundaries.

- If you have **root privileges in WSL**, you effectively have Administrator access on the Windows host (via mounting `C:\`).
- If you don't have root in WSL, perform standard Unix Privesc (see next lecture) techniques first.



Vulnerability: Blurring the lines between OS boundaries.

- If you have **root privileges in WSL**, you effectively have Administrator access on the Windows host (via mounting `C:\`).
- If you don't have root in WSL, perform standard Unix Privesc (see next lecture) techniques first.

Vulnerability: Abusing assigned privileges to get better tokens.

- **SeImpersonatePrivilege:** Impersonate (but not create) any token. (See Potatoes).
- **SeCreateTokenPrivilege :** Create token for a process.
- **SeAssignPrimaryTokenPrivilege:** Replace token of a process.
- **SeTakeOwnershipPrivilege:** Take Ownership of on object.

Vulnerability: Abusing assigned privileges to get better tokens.

- **SeImpersonatePrivilege:** Impersonate (but not create) any token. (See Potatoes).
- **SeCreateTokenPrivilege :** Create token for a process.
- **SeAssignPrimaryTokenPrivilege:** Replace token of a process.
- **SeTakeOwnershipPrivilege:** Take Ownership of on object.

Vulnerability: Abusing assigned privileges to get better tokens.

- **SeImpersonatePrivilege:** Impersonate (but not create) any token. (See Potatoes).
- **SeCreateTokenPrivilege :** Create token for a process.
- **SeAssignPrimaryTokenPrivilege:** Replace token of a process.
- **SeTakeOwnershipPrivilege:** Take Ownership of on object.

Vulnerability: Abusing assigned privileges to get better tokens.

- **SeImpersonatePrivilege:** Impersonate (but not create) any token. (See Potatoes).
- **SeCreateTokenPrivilege :** Create token for a process.
- **SeAssignPrimaryTokenPrivilege:** Replace token of a process.
- **SeTakeOwnershipPrivilege:** Take Ownership of on object.

Vulnerability: Abusing assigned privileges to access file system.

- **SeBackupPrivilege:** Grants read access to the **entire** filesystem (ignores NTFS rules).
- **SeRestorePrivilege:** Grants write access to the **entire** filesystem.

Vulnerability: Abusing assigned privileges to access file system.

- **SeBackupPrivilege:** Grants read access to the **entire** filesystem (ignores NTFS rules).
- **SeRestorePrivilege:** Grants write access to the **entire** filesystem.

Vulnerability: Abusing assigned token privileges.

- **SeLoadDriverPrivilege:** Allows loading drivers (useful for BYOVD).
- **SeDebugPrivilege:** Allows debugging/injecting into other processes.
- **SeTcbPrivilege** Act as part of the OS, allows impersonation.

```
PS C:\Users\User> whoami /priv
```

Vulnerability: Abusing assigned token privileges.

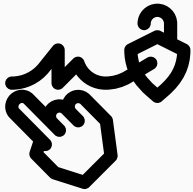
- **SeLoadDriverPrivilege:** Allows loading drivers (useful for BYOVD).
- **SeDebugPrivilege:** Allows debugging/injecting into other processes.
- **SeTcbPrivilege** Act as part of the OS, allows impersonation.

```
PS C:\Users\User> whoami /priv
```

Vulnerability: Abusing assigned token privileges.

- **SeLoadDriverPrivilege:** Allows loading drivers (useful for BYOVD).
- **SeDebugPrivilege:** Allows debugging/injecting into other processes.
- **SeTcbPrivilege** Act as part of the OS, allows impersonation.

```
PS C:\Users\User> whoami /priv
```



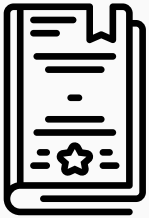
Vulnerability: GUI applications spawned in high-integrity contexts.

- Software centers launch installers as **Admin** or **SYSTEM**.
- **Attack:** If an installer provides a "Help" or "Browse" menu, it opens an `explorer.exe` or `browser` dialog in that same context.
- From that File Explorer, you can right-click and spawn a `cmd.exe` running as SYSTEM.

Vulnerability: Dangerous Group Policy configuration.

- If **both** of these registry keys are set to `1`, any `.msi` file run by a standard user executes as `NT AUTHORITY\SYSTEM`.

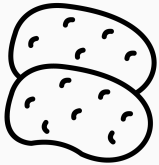
```
# Check both HKCU and HKLM
PS> reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v
    AlwaysInstallElevated
PS> reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v
    AlwaysInstallElevated
```



Vulnerability: Unpatched known CVEs.

- Check OS versions, build numbers, and installed software.
- Look for public exploits on `https://www.exploit-db.com` or GitHub.

```
# Get patch history
PS> Get-HotFix
# Or system info
PS> systeminfo
```



Vulnerability: A series of privilege escalation techniques abusing Windows authentication (NTLM), tokens, and named pipes.

- **Target:** Service accounts with `SeImpersonatePrivilege`.
- **All-in-one:** When in doubt, try Sweet Potato.

—

Read the Blog Post *summarizing the history and mechanics of all potatoes.*

Enumeration

Search the system, look for something standing out

- User files (C:\Users\, %APPDATA%)
- Custom services and executables
- Version enumeration
- Scheduled tasks and services
- Processes running on the system

Essential Enumeration Scripts:

- **WinPEAS:** The industry standard for automated enumeration.
  - `https://github.com/carlospolop/PEASS-ng`
- **Seatbelt:** Highly modular C# enumeration tool.
  - `https://github.com/GhostPack/Seatbelt`
- **SharpUp:** Updated C# port of the classic PowerUp script.
  - `https://github.com/GhostPack/SharpUp`
- **LOLBAS:** Directory of built-in executables for post-exploitation.
  - `https://lolbas-project.github.io`

Checklists and Methodologies:

- **HackTricks Book:**
  - `https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation`
- **Internal All The Things:**
  - `https://swisskyrepo.github.io/InternalAllTheThings/redteam/escalation/windows-privilege-escalation`

Try it yourself

We have prepared a vulnerable server that contains many of the previously explained vulnerabilities. For each vulnerability, you will receive a flag to submit to CTFd. For more information, refer to the Challenge Descriptions.

To get started, look at the **Privesc Lecture Challenges Intro**.

Any Questions?