

# about\_Token\_Privileges

## by Thorsten Butz



Jane Street



IRONMAN SOFTWARE

KNOWLEDGE  
FACTORY



SynEdgy



@thorsten.butz.io

```
# > whoami.exe /priv
```

# PRIVILEGES INFORMATION

-----

Privilege Name	Description	State
=====	=====	=====
SeShutdownPrivilege	Shut down the system	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeUndockPrivilege	Remove computer from docking station	Disabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled
SeTimeZonePrivilege	Change the time zone	Disabled



User Account Control



Do you want to allow this app to make changes to your device?



Terminal

Verified publisher: Microsoft Corporation

File origin: Hard drive on this computer

[Show more details](#)

To continue, enter an admin username and password.

Terminal will also be installed for the administrator.

User name

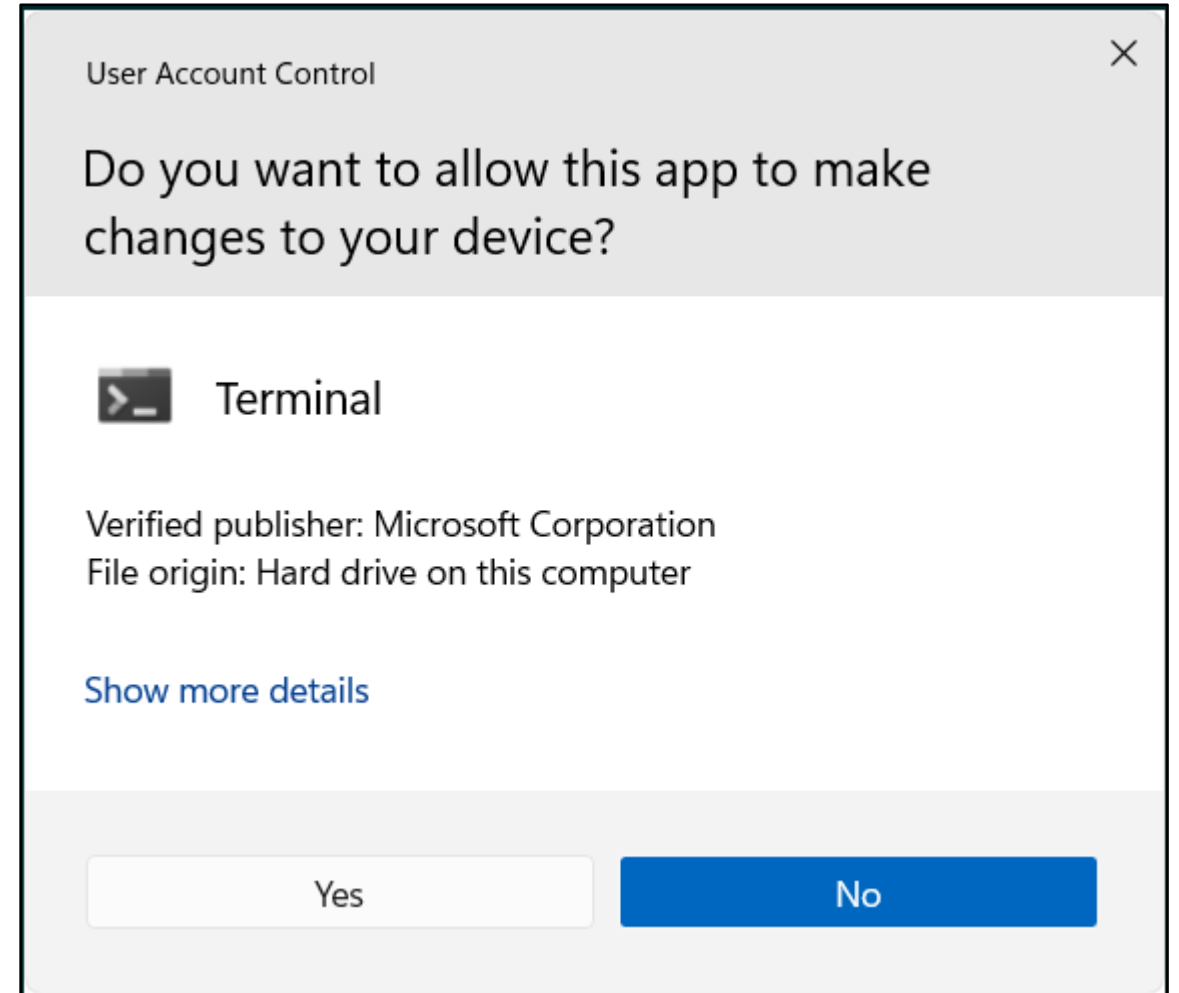
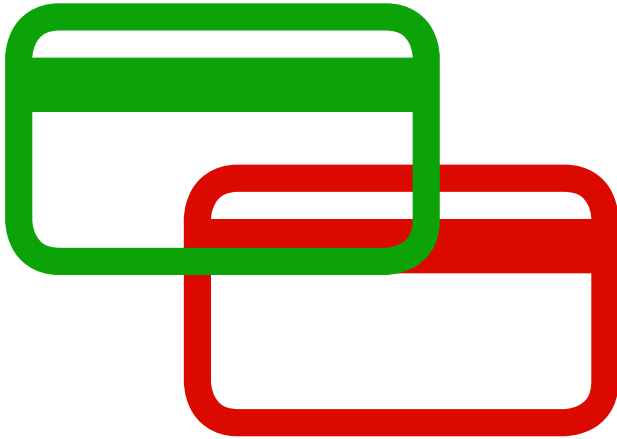
Password

Domain: CONTOSO

Yes

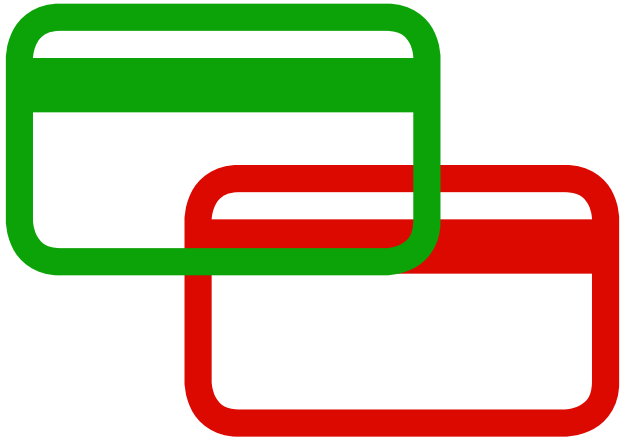
No

# Windows Vista: User Account Control



# "Protected Admin"

Vista: Split token login



## PRIVILEGES INFORMATION

Privilege Name	Description	State
SeShutdownPrivilege	Shut down the system	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeUndockPrivilege	Remove computer from docking station	Disabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled
SeTimeZonePrivilege	Change the time zone	Disabled

## PRIVILEGES INFORMATION

Privilege Name	Description	State
SeIncreaseQuotaPrivilege	Adjust memory quotas for a process	Disabled
SeSecurityPrivilege	Manage auditing and security log	Disabled
SeTakeOwnershipPrivilege	Take ownership of files or other objects	Disabled
SeLoadDriverPrivilege	Load and unload device drivers	Disabled
SeSystemProfilePrivilege	Profile system performance	Disabled
SeSystemtimePrivilege	Change the system time	Disabled
SeProfileSingleProcessPrivilege	Profile single process	Disabled
SeIncreaseBasePriorityPrivilege	Increase scheduling priority	Disabled
SeCreatePagefilePrivilege	Create a pagefile	Disabled
SeBackupPrivilege	Back up files and directories	Disabled
SeRestorePrivilege	Restore files and directories	Disabled
SeShutdownPrivilege	Shut down the system	Disabled
SeDebugPrivilege	Debug programs	Enabled
SeSystemEnvironmentPrivilege	Modify firmware environment values	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeRemoteShutdownPrivilege	Force shutdown from a remote system	Disabled
SeUndockPrivilege	Remove computer from docking station	Disabled
SeManageVolumePrivilege	Perform volume maintenance tasks	Disabled
SeImpersonatePrivilege	Impersonate a client after authentication	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled
SeTimeZonePrivilege	Change the time zone	Disabled
SeCreateSymbolicLinkPrivilege	Create symbolic links	Disabled
SeDelegateSessionUserImpersonatePrivilege	Obtain an impersonation token for another user in the same session	Disabled

## PRIVILEGES INFORMATION

Privilege Name	Description	State
SeIncreaseQuotaPrivilege	Adjust memory quotas for a process	Disabled
SeSecurityPrivilege	Manage auditing and security log	Disabled
SeTakeOwnershipPrivilege	Take ownership of files or other objects	Disabled
SeLoadDriverPrivilege	Load and unload device drivers	Disabled
SeSystemProfilePrivilege	Profile system performance	Disabled
SeSystemtimePrivilege	Change the system time	Disabled
SeProfileSingleProcessPrivilege	Profile single process	Disabled
SeIncreaseBasePriorityPrivilege	Increase scheduling priority	Disabled
SeCreatePagefilePrivilege	Create a pagefile	Disabled
SeBackupPrivilege	Back up files and directories	Disabled
SeRestorePrivilege	Restore files and directories	Disabled
SeShutdownPrivilege	Shut down the system	Disabled
SeDebugPrivilege	Debug programs	Enabled
SeSystemEnvironmentPrivilege	Modify firmware environment values	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeRemoteShutdownPrivilege	Force shutdown from a remote system	Disabled
SeUndockPrivilege	Remove computer from docking station	Disabled
SeManageVolumePrivilege	Perform volume maintenance tasks	Disabled
SeImpersonatePrivilege	Impersonate a client after authentication	Enabled
SeCreateGlobalPrivilege	Create global objects	Enabled
SeIncreaseWorkingSetPrivilege	Increase a process working set	Disabled
SeTimeZonePrivilege	Change the time zone	Disabled
SeCreateSymbolicLinkPrivilege	Create symbolic links	Disabled
SeDelegateSessionUserImpersonatePrivilege	Obtain an impersonation token for another user in the same session	Disabled



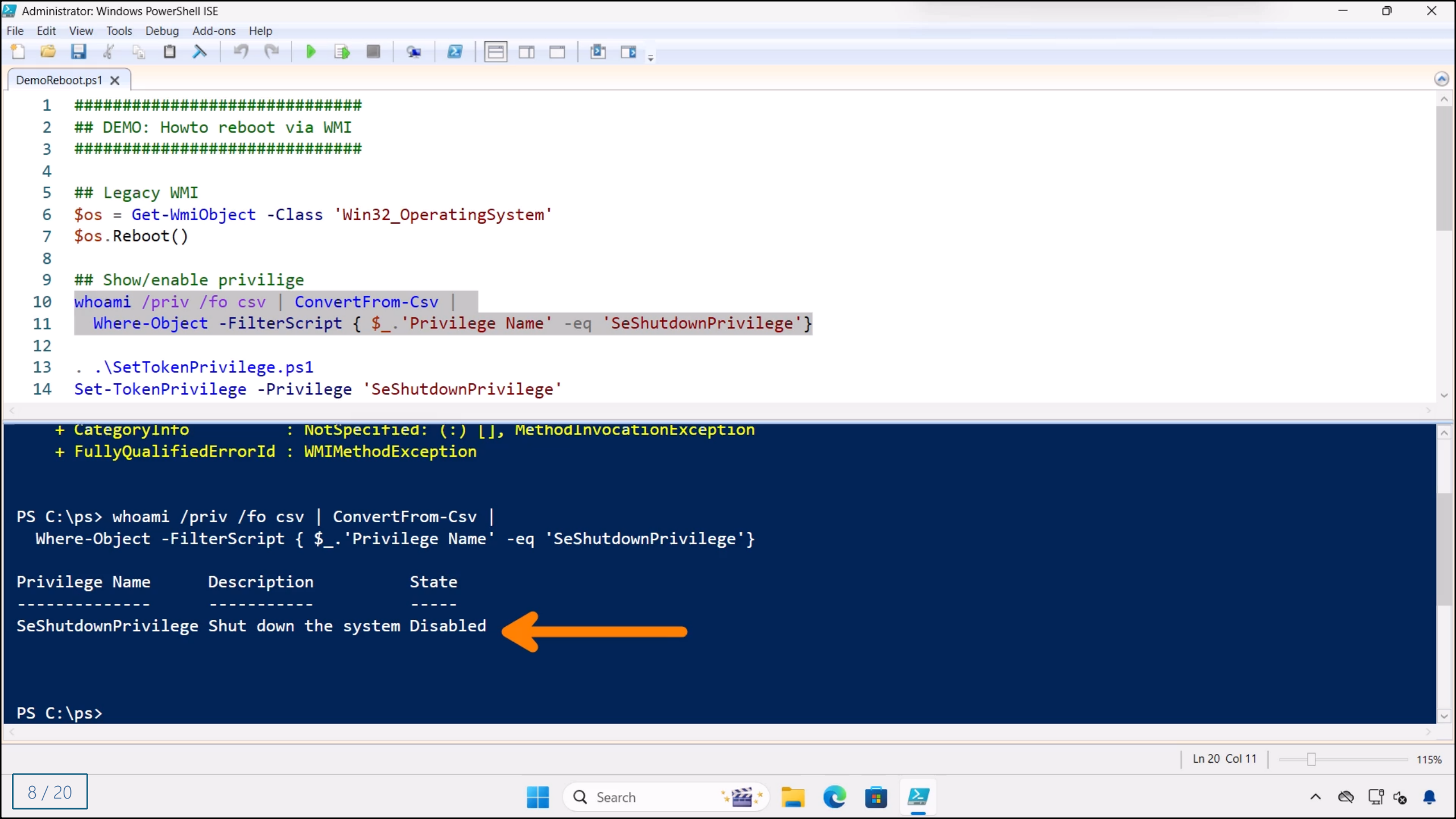
DemoReboot.ps1 X

```
1 #####
2 ## DEMO: Howto reboot via WMI
3 #####
4
5 ## Legacy WMI
6 $os = Get-WmiObject -Class 'Win32_OperatingSystem'
7 $os.Reboot()
8
9 ## Show/enable privilege
10 whoami /priv /fo csv | ConvertFrom-Csv |
11   Where-Object -FilterScript { $_.'Privilege Name' -eq 'SeShutdownPrivilege'}
12
13 . .\SetTokenPrivilege.ps1
14 Set-TokenPrivilege -Privilege 'SeShutdownPrivilege'
```

PS C:\ps&gt;







```
1 #####
2 ## DEMO: Howto reboot via WMI
3 #####
4
5 ## Legacy WMI
6 $os = Get-WmiObject -Class 'Win32_OperatingSystem'
7 $os.Reboot()
8
9 ## Show/enable privilege
10 whoami /priv /fo csv | ConvertFrom-Csv |
11   Where-Object -FilterScript { $_.'Privilege Name' -eq 'SeShutdownPrivilege'}
12
13 . .\SetTokenPrivilege.ps1
14 Set-TokenPrivilege -Privilege 'SeShutdownPrivilege'
```

```
+ CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
+ FullyQualifiedErrorId : WMIMethodException
```

```
PS C:\ps> whoami /priv /fo csv | ConvertFrom-Csv |
  Where-Object -FilterScript { $_.'Privilege Name' -eq 'SeShutdownPrivilege'}
```

Privilege Name	Description	State
-----	-----	----
SeShutdownPrivilege	Shut down the system	Disabled



```
PS C:\ps>
```



```

## Taken from P/Invoke.NET with minor adjustments.
$definition = @'
using System;
using System.Runtime.InteropServices;
public class AdjPriv {
    [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
    internal static extern bool AdjustTokenPrivileges(IntPtr htok, bool disall,
        ref TokPriv1Luid newst, int len, IntPtr prev, IntPtr relen);

    [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
    internal static extern bool OpenProcessToken(IntPtr h, int acc, ref IntPtr phtok);

    [DllImport("advapi32.dll", SetLastError = true)]
    internal static extern bool LookupPrivilegeValue(string host, string name, ref long pluid);

    [StructLayout(LayoutKind.Sequential, Pack = 1)]
    public struct TokPriv1Luid {
        public int Count;
        public long Luid;
        public int Attr;
    }

    internal const int SE_PRIVILEGE_ENABLED = 0x00000002;
    internal const int SE_PRIVILEGE_DISABLED = 0x00000000;
    internal const int TOKEN_QUERY = 0x00000008;
    internal const int TOKEN_ADJUST_PRIVILEGES = 0x00000020;

    [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
    internal static extern bool AdjustTokenPrivileges(IntPtr htok, bool disable,
        ref TokPriv1Luid newst, int len, IntPtr prev, IntPtr relen);

    [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
    internal static extern bool OpenProcessToken(IntPtr h, int acc, ref IntPtr phtok);

    ...

    if(disable) {
        tp.Attr = SE_PRIVILEGE_DISABLED;
    }
    else {
        tp.Attr = SE_PRIVILEGE_ENABLED;
    }

    retVal = LookupPrivilegeValue(null, privilege, ref tp.Luid);
    retVal = AdjustTokenPrivileges(htok, false, ref tp, 0, IntPtr.Zero, IntPtr.Zero);
    return retVal;
}
'@

```

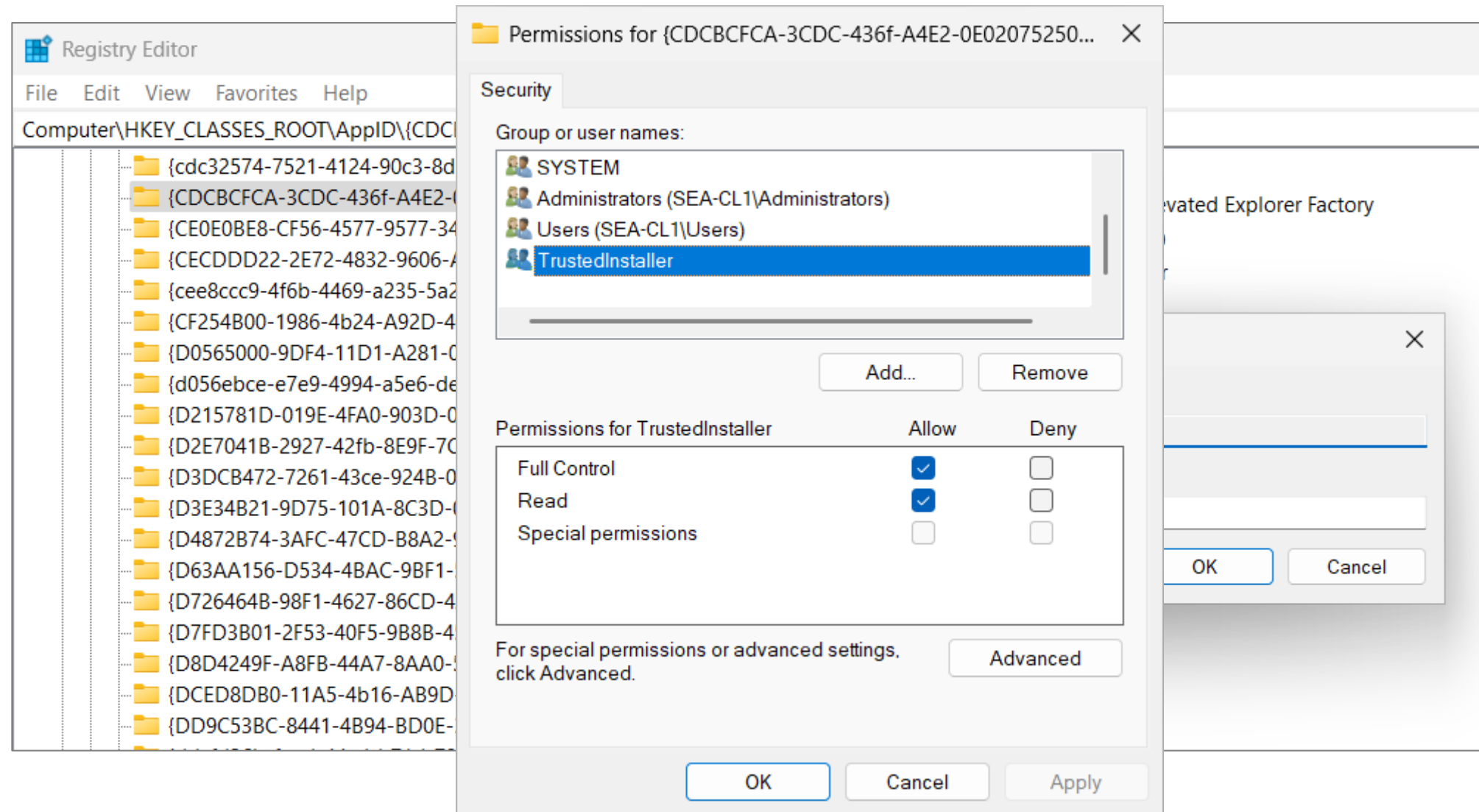
# Token privileges: a 2<sup>nd</sup> step authorization

"A privilege is the right of an account, such as a user or group account, to perform various system-related operations on the local computer, such as shutting down the system, loading device drivers, or changing the system time."



<https://learn.microsoft.com/en-us/windows/win32/secauthz/privileges>

# Run explorer.exe as administrator



```
## Enable SeTakeOwnership privilege
. .\SetTokenPrivilege.ps1
Set-TokenPrivilege -Privilege 'SeTakeOwnershipPrivilege' -ProcessId $pid

## Define RegKeys (Since Set-ACL does not work here, we use the .NET API)
$baseKey = [Microsoft.Win32.Registry]::ClassesRoot ## 'HKEY_CLASSES_ROOT'
$subKey = 'AppID\{CDCBCFCA-3CDC-436f-A4E2-0E02075250C2}'
$me = [System.Security.Principal.NTAccount]::new("$env:USERDOMAIN\$env:USERNAME")

## Creating the required RegistryKey object (Microsoft.Win32.RegistryKey)
$regKey = $baseKey.OpenSubKey(
    $subKey,
    [Microsoft.Win32.RegistryKeyPermissionCheck]::ReadWriteSubTree,
    [System.Security.AccessControl.RegistryRights]::TakeOwnership
)

## Getting the current ACL
$acl = $regKey.GetAccessControl()
$acl.owner
$acl.Sddl
```

```
## Let the calling account become the owner
```

```
$acl.SetOwner($me)
```

```
$regKey.SetAccessControl($acl)
```

```
## Define new rule: Allow "me" to have full control
```

```
$rule = [System.Security.AccessControl.RegistryAccessRule]::new(
```

```
    $me,
```

```
    [System.Security.AccessControl.RegistryRights]::FullControl ,
```

```
    [System.Security.AccessControl.InheritanceFlags]'ObjectInherit,ContainerInherit',
```

```
    [System.Security.AccessControl.PropagationFlags]::None ,
```

```
    [System.Security.AccessControl.AccessControlType]::Allow
```

```
)
```

```
$acl.AddAccessRule($rule)
```

```
$regKey.SetAccessControl( $acl )
```

```
## "Remove" the blocker
```

```
Rename-ItemProperty -Path "Registry::$($regKey.Name)" -Name 'RunAs' -NewName 'RunA_'
```



explorer.exe:7708 Properties

Image	Performance	Performance Graph	Disk and Network	GPU Graph
Threads	TCP/IP	Security	Environment	Strings

User: CONTOSO\MossAdmin  
 SID: S-1-5-21-1276007578-2914169427-449847108-1108  
 Session: 4 Logon Session: 21f8f29  
 Virtualized: No Protected: No

Group	Flags
Authentication authority asserted identity	Mandatory
BUILTIN\Administrators	Deny
BUILTIN\Remote Desktop Users	Mandatory
BUILTIN\Users	Mandatory
CONSOLE LOGON	Mandatory
CONTOSO\Domain Users	Mandatory
Everyone	Mandatory
LOCAL	Mandatory
Mandatory Label\Medium Mandatory Level	Integrity
NT AUTHORITY\Authenticated Users	Mandatory
NT AUTHORITY\INTERACTIVE	Mandatory
NT AUTHORITY\LogonSessionId_0_35622507	Mandatory
NT AUTHORITY\This Organization	Mandatory

Group SID: n/a

Privilege	Flags
SeChangeNotifyPrivilege	Default Enabled
SeIncreaseWorkingSetPrivilege	Disabled
SeShutdownPrivilege	Disabled
SeTimeZonePrivilege	Disabled
SeUndockPrivilege	Disabled

explorer.exe:6592 Properties

Image	Performance	Performance Graph	Disk and Network	GPU Graph
Threads	TCP/IP	Security	Environment	Strings

User: CONTOSO\MossAdmin  
 SID: S-1-5-21-1276007578-2914169427-449847108-1108  
 Session: 4 Logon Session: 21f8ec9  
 Virtualized: No Protected: No

Group	Flags
Authentication authority asserted identity	Mandatory
BUILTIN\Administrators	Owner
BUILTIN\Remote Desktop Users	Mandatory
BUILTIN\Users	Mandatory
CONSOLE LOGON	Mandatory
CONTOSO\Domain Users	Mandatory
Everyone	Mandatory
LOCAL	Mandatory
Mandatory Label\High Mandatory Level	Integrity
NT AUTHORITY\Authenticated Users	Mandatory
NT AUTHORITY\INTERACTIVE	Mandatory
NT AUTHORITY\LogonSessionId_0_35622507	Mandatory
NT AUTHORITY\This Organization	Mandatory

Group SID: n/a

Privilege	Flags
SeBackupPrivilege	Enabled
SeRestorePrivilege	Enabled
SeCreateGlobalPrivilege	Default Enabled
SeCreatePagefilePrivilege	Disabled
SeCreateSymbolicLinkPrivilege	Disabled
SeDebugPrivilege	Enabled
SeDelegateSessionUserImpersonatePrivilege	Disabled
SeImpersonatePrivilege	Default Enabled

\\SEA-CL1  
CONTOSO\MossAdmin



Recycle Bin

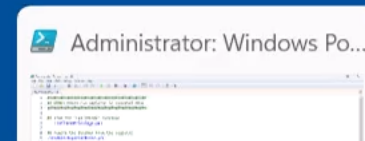




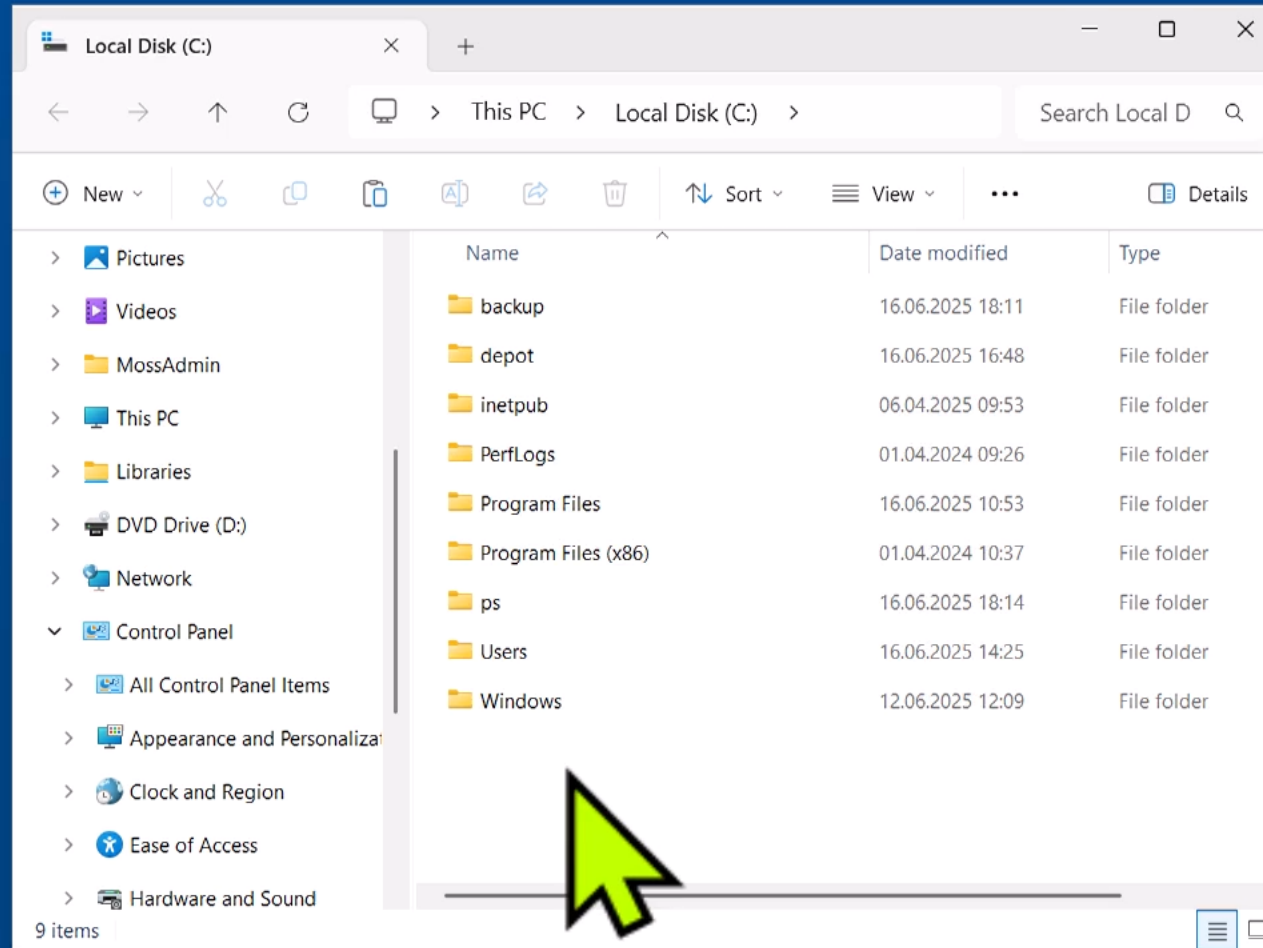
\\SEA-CL1  
CONTOSO\MossAdmin



Recycle Bin



\\SEA-CL1  
CONTOSO\MossAdmin



Recycle Bin



# Holding out for a hero

```
# > robocopy /mir C:\Users\BarberJ\Desktop\ C:\backup\ /b
```



```
Add-Type -TypeDefinition @"
using System;
using System.IO;
using System.Runtime.InteropServices;
using Microsoft.Win32.SafeHandles;
```

```
public class BackupCopy {
    const uint GENERIC_READ = 0x80000000;
    const uint GENERIC_WRITE = 0x40000000;
    const uint FILE_FLAG_BACKUP_SEMANTICS = 0x02000000;
    const uint OPEN_EXISTING = 3;
    const uint CREATE_ALWAYS = 1;
    const uint CREATE_NEW = 2;
    const uint FILE_ATTRIBUTE_NORMAL = 0x80;

    [DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Unicode)]
    public static extern SafeFileHandle CreateFile(
        string lpFileName,
        uint dwDesiredAccess,
        uint dwShareMode,
        IntPtr lpSecurityAttributes,
        uint dwCreationDisposition,
        uint dwFlagsAndAttributes,
        IntPtr hTemplate);

    public class BackupCopy {
        const uint GENERIC_READ = 0x80000000;
        const uint GENERIC_WRITE = 0x40000000;
        const uint FILE_FLAG_BACKUP_SEMANTICS = 0x02000000;
        const uint OPEN_EXISTING = 3;

        public static void CopyFileWithBackupSemantics(
            string src, string dst, bool overwrite = true) {
            uint creationMode = overwrite ? CREATE_ALWAYS : CREATE_NEW;
            ...
            var srcHandle = CreateFile(source, GENERIC_READ, 0, IntPtr.Zero, OPEN_EXISTING,
                FILE_FLAG_BACKUP_SEMANTICS, IntPtr.Zero);
            if (srcHandle.IsInvalid)
                throw new IOException("Failed to open source file: " + source, Marshal.GetLastWin32Error());
            var dstHandle = CreateFile(dst, GENERIC_WRITE, 0, IntPtr.Zero, creationMode,
                FILE_ATTRIBUTE_NORMAL, IntPtr.Zero);
            if (dstHandle.IsInvalid)
                throw new IOException("Failed to create destination file: " + dst, Marshal.GetLastWin32Error());

            using (var src = new FileStream(srcHandle, FileAccess.Read))
            using (var dst = new FileStream(dstHandle, FileAccess.Write)) {
                src.CopyTo(dst);
            }
        }
    }
}
"@
```



P/INVOKE

Add-Type -TypeDefinition @"

using System;

using System.IO;

using System.Runtime.InteropServices;

using Microsoft.Win32.SafeHandles;

public class BackupCopy {

const uint GENERIC\_READ = 0x80000000;

const uint GENERIC\_WRITE = 0x40000000;

const uint FILE\_FLAG\_BACKUP\_SEMANTICS = 0x02000000;

const uint OPEN\_EXISTING = 3;

...

var srcHandle = CreateFile(source, GENERIC\_READ, 0, IntPtr.Zero, OPEN\_EXISTING, FILE\_FLAG\_BACKUP\_SEMANTICS, IntPtr.Zero);

if (srcHandle.IsInvalid)

throw new IOException("Failed to open source file: " + source, Marshal.GetLastWin32Error());

## Copy a single file (set overwrite to true or false)

var dstHandle = CreateFile(dst, GENERIC\_WRITE, 0, IntPtr.Zero, creationMode, FILE\_ATTRIBUTE\_NORMAL, IntPtr.Zero);

if (dstHandle.IsInvalid)

throw new IOException("Failed to create destination file: " + dst, Marshal.GetLastWin32Error());

[BackupCopy]::CopyFileWithBackupSemantics(\$src,\$dst,\$false)

using (var src = new FileStream(srcHandle, FileAccess.Read))

using (var dst = new FileStream(dstHandle, FileAccess.Write)) {

src.CopyTo(dst);

}

}

}

@

# In a nutshell

- 💡 **Token privileges** must be granted (during logon) and activated (at runtime).
- 💡 **UAC** works like a shim to enable elevated permissions while NOT activating privileges.
- 💡 You have to deal with the Win32 API and P/Invoke to utilize **NTFS backup semantics** combined with privileges like SeBackupPrivilege or SeRestorePrivilege to let processes bypass file system ACLs for reading, writing or restoring.
- 🦹 **Robocopy** has built-in superpowers for everyday working life.