# Blocking default opening of “tar” and “gz” files in windows 11

## Summary

On Windows 11, Microsoft have enabled Windows Explorer to open a lot more “archive” formats to increase compatibility with other operating systems (e.g. Linux)

2 of these file types are common enough that they could be used by attackers to transfer malware to a Windows computer, but because they are not considered “Windows archive formats” they are not blocked by any of the usual tools or firewalls.

Anti-virus software should stop any bad software from being executed from within these files, but this would not stop any software that is not considered a virus from being executed. (e.g. remote access software)

To this end, the simplest solution is to add in an additional “handler” for these 2 file types, so that they either don’t run automatically or bring up a message to say that they are not going to open.

## Code for Executable

In order to make a simple message appear on Windows, the easiest way to do this is with a simple C# program.

The code for the BlockMessage executable that I will be using is as follows:

using System.Windows.Forms;

public class BlockMessage

{

    public static void Main(string[] args)

    {

        // Message to display

        string message = "This file is blocked from opening in Windows!";

        // Title for the message box

        string title = "File Blocked";

        // Display the message box with OK button

        MessageBox.Show(message, title, MessageBoxButtons.OK, MessageBoxIcon.Warning);

    }

}

## Compiling Executable

I chose to cross-compile the C# executable on Linux, using Mono, to reduce the possibility of significant library overhead. The single command line needed, to produce the “BlockMessage.exe”, for compilation is:

$ mcs -target:winexe -r:System.Windows.Forms.dll -out:BlockMessage.exe BlockMessage.cs

## Code for SetFileAssocations

We also need some scripting code to run on the Windows computer in order to set up the new file “associations” for this handler. I created a file called SetFileAssociations.ps1 with the following contents:

# Define the file extensions and associated executable

$fileExtensions = @(".tar", ".gz")

$exePath = "C:\SGS\BlockMessage.exe"

$progIDPrefix = "blockmessagefile"

# Function to remove existing registry keys for a specific user hive

function Remove-UserRegistryKeys {

    param (

        [string]$userHiveRoot,

        [string]$extension

    )

    $fileExtKey = "$userHiveRoot\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\$extension"

    # Remove the extension key if it exists

    if (Test-Path $fileExtKey) {

        Remove-Item -Path $fileExtKey -Recurse -Force

    }

}

# Function to set registry keys for a specific user hive

function Set-UserRegistryKeys {

    param (

        [string]$userHiveRoot,

        [string]$extension,

        [string]$progID,

        [string]$exePath

    )

    $fileExtKey = "$userHiveRoot\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\$extension"

    # Create the necessary registry keys if they don't exist

    if (-Not (Test-Path "$fileExtKey\OpenWithProgids")) {

        New-Item -Path "$fileExtKey\OpenWithProgids" -Force

    }

    Set-ItemProperty -Path "$fileExtKey\OpenWithProgids" -Name $progID -Value 0

    # Create and set UserChoice key

    $userChoiceKey = "$fileExtKey\UserChoice"

    if (-Not (Test-Path $userChoiceKey)) {

        New-Item -Path $userChoiceKey -Force

    }

    Set-ItemProperty -Path $userChoiceKey -Name "ProgId" -Value $progID

    Set-ItemProperty -Path $userChoiceKey -Name "Hash" -Value ""

}

# Function to remove existing registry keys in HKLM

function Remove-MachineRegistryKeys {

    param (

        [string]$extension

    )

    $rootPath = "HKLM:\Software\Classes"

    $extensionKey = "$rootPath\$extension"

    $progIDKey = "$rootPath\$extension\shell\open\command"

    $openWithProgidsKey = "$rootPath\$extension\OpenWithProgids"

    # Remove the extension key if it exists

    if (Test-Path $extensionKey) {

        Remove-Item -Path $extensionKey -Recurse -Force

    }

    # Remove the ProgID key if it exists

    if (Test-Path $progIDKey) {

        Remove-Item -Path $progIDKey -Recurse -Force

    }

    # Remove the OpenWithProgids key if it exists

    if (Test-Path $openWithProgidsKey) {

        Remove-Item -Path $openWithProgidsKey -Recurse -Force

    }

}

# Function to set registry keys in HKLM

function Set-MachineRegistryKeys {

    param (

        [string]$extension,

        [string]$progID,

        [string]$exePath

    )

    $rootPath = "HKLM:\Software\Classes"

    # Create the necessary registry keys if they don't exist

    if (-Not (Test-Path "$rootPath\$extension")) {

        New-Item -Path "$rootPath\$extension" -Force

    }

    Set-ItemProperty -Path "$rootPath\$extension" -Name "(default)" -Value $progID

    if (-Not (Test-Path "$rootPath\$progID\shell\open\command")) {

        New-Item -Path "$rootPath\$progID\shell\open\command" -Force

    }

    Set-ItemProperty -Path "$rootPath\$progID\shell\open\command" -Name "(default)" -Value "`"$exePath`" `"%1`""

    # Add ProgID to OpenWithProgids

    if (-Not (Test-Path "$rootPath\$extension\OpenWithProgids")) {

        New-Item -Path "$rootPath\$extension\OpenWithProgids" -Force

    }

    if (-Not (Get-ItemProperty -Path "$rootPath\$extension\OpenWithProgids" -Name $progID -ErrorAction SilentlyContinue)) {

        New-ItemProperty -Path "$rootPath\$extension\OpenWithProgids" -Name $progID -PropertyType String -Value ""

    }

}

foreach ($extension in $fileExtensions) {

    $progID = "${progIDPrefix}\_$($extension.TrimStart('.'))"

    # Remove existing keys in HKLM

    Remove-MachineRegistryKeys -extension $extension

    # Set new associations in HKLM

    Set-MachineRegistryKeys -extension $extension -progID $progID -exePath $exePath

}

Write-Host "File associations updated successfully for all users."

## Code for SetUserAssocations

We also need some scripting code to run on the Windows computer in order to set up the new file “associations” for this handler. I created a file called SetUserAssociations.ps1 with the following contents:

# Define the file extensions and associated executable

$fileExtensions = @(".tar", ".gz")

$exePath = "C:\SGS\BlockMessage.exe"

$progIDPrefix = "blockmessagefile"

# Function to remove existing registry keys for a specific user hive

function Remove-UserRegistryKeys {

    param (

        [string]$userHiveRoot,

        [string]$extension

    )

    $fileExtKey = Join-Path "$userHiveRoot\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts" $extension

    try {

        Write-Output "Attempting to remove keys at $fileExtKey..."

        if (Test-Path $fileExtKey) {

            Remove-Item -Path $fileExtKey -Recurse -Force -ErrorAction Stop

            Write-Output "Extension key $fileExtKey removed successfully."

        } else {

            Write-Output "Extension key $fileExtKey does not exist."

        }

    } catch {

        Write-Output "ERROR: Failed to remove keys at $fileExtKey. $\_"

    }

}

# Function to set registry keys for a specific user hive

function Set-UserRegistryKeys {

    param (

        [string]$userHiveRoot,

        [string]$extension,

        [string]$progID,

        [string]$exePath

    )

    $fileExtKey = Join-Path "$userHiveRoot\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts" $extension

    try {

        Write-Output "Attempting to set keys at $fileExtKey..."

        # Create or access the FileExts key

        if (-not (Test-Path $fileExtKey)) {

            Write-Output "creating $fileExtKey..."

            New-Item -Path $fileExtKey -Force | Out-Null

        }

        # Create or access the OpenWithProgids key

        $openWithProgidsKey = Join-Path $fileExtKey "OpenWithProgids"

        if (-not (Test-Path $openWithProgidsKey)) {

            Write-Output "creating $openWithProgidsKey..."

            New-Item -Path $openWithProgidsKey -Force | Out-Null

        }

        # Set the ProgID value under OpenWithProgids key

        Set-ItemProperty -Path $openWithProgidsKey -Name $progID -Value 0 -ErrorAction Stop

        # Create or access the UserChoice key

        $userChoiceKey = Join-Path $fileExtKey "UserChoice"

        if (-not (Test-Path $userChoiceKey)) {

            Write-Output "creating $userChoiceKey..."

            New-Item -Path $userChoiceKey -Force | Out-Null

        }

        # Set the ProgId and Hash values for UserChoice key

        Set-ItemProperty -Path $userChoiceKey -Name "ProgId" -Value $progID -ErrorAction Stop

        Set-ItemProperty -Path $userChoiceKey -Name "Hash" -Value ([byte[]]@(0)) -ErrorAction Stop

        Write-Output "Keys set successfully at $fileExtKey."

    } catch {

        Write-Output "ERROR: Failed to set keys at $fileExtKey. $\_"

    }

}

# Function to update registry keys for a user

function Update-UserRegistry {

    param (

        [array]$fileExtensions,

        [string]$progIDPrefix,

        [string]$exePath,

        [string]$profilePath,

        [string]$userName

    )

    # Flag to indicate if the profile is already loaded

    $profileLoaded = $false

    try {

        # Check if the profile is already loaded

        $loadedProfiles = Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\hivelist" | ForEach-Object { $\_.PSObject.Properties }

        $loadedProfile = $loadedProfiles | Where-Object { $\_.Value -match "$userName\\NTUSER\.DAT" }

        if ($loadedProfile) {

            Write-Output "Profile for user $userName is already loaded. Using existing hive."

            $hiveRoot = "Registry::HKU$($loadedProfile.Name -replace '^\\REGISTRY\\USER', '')"

            $profileLoaded = $true

        } else {

            # Load the user's registry hive

            $userHive = "HKU\TempHive"

        $hiveRoot = "Registry::HKU\TempHive"

            $regFile = "$profilePath\NTUSER.DAT"

            reg load $userHive $regFile

        }

        Write-Output "Updating registry for user: $userName"

        foreach ($extension in $fileExtensions) {

            $progID = "${progIDPrefix}\_$($extension.TrimStart('.'))"

            Remove-UserRegistryKeys -userHiveRoot $hiveRoot -extension $extension

            Set-UserRegistryKeys -userHiveRoot $hiveRoot -extension $extension -progID $progID -exePath $exePath

        }

        # Unload the registry hive if it was loaded in this function and not previously loaded

        if (!$profileLoaded) {

            [gc]::Collect()

        [gc]::WaitForPendingFinalizers()

        reg unload $userHive

        }

    } catch {

        Write-Output "ERROR: Failed to update registry for user $userName : $\_"

        if (!$profileLoaded -and $userHive -eq "HKU\TempHive") {

            reg unload $userHive

        }

    }

}

# Get all user profile paths

$userProfiles = Get-WmiObject Win32\_UserProfile | Where-Object { $\_.Special -eq $false } | Select-Object -ExpandProperty LocalPath

foreach ($profilePath in $userProfiles) {

    $userName = Split-Path $profilePath -Leaf

    Update-UserRegistry -profilePath $profilePath -userName $userName -fileExtensions $fileExtensions -progIDPrefix $progIDPrefix -exePath $exePath

}

Write-Host "File associations updated successfully for all users."

## Zip + Install

The assumption in the above script is that the “BlockMessage.exe” is in a folder on the C: drive called “SGS”, so you need to create a folder in C:\SGS and put the EXE in that folder. I put both the script and the EXE in that folder, for easy potential maintenance.

You will need an elevated/Administrator PowerShell to run the SetFileAssociations script, and you might find it easier to run all the commands in the same shell:

First make the SGS folder:

# Make the C:\SGS folder

C:

mkdir C:\SGS

cd C:\SGS

Then unzip the zip file:

# expand the zip file

Expand-Archive 'C:\Folder\To\Zip\BlockMessage.zip'

# move the files from the BlockMessage subdirectory to the current one

mv BlockMessage/\* .

rmdir BlockMessage

If PowerShell doesn’t want to run the SetFileAssociations script, you will need to run the following command:

# might need the following command running first:

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

After that you will be able to run the C:\SGS\SetFileAssociations.ps1 script:

PS C:\SGS> .\SetFileAssociations.ps1

File associations updated successfully for all users.