Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

Kevin-Robertson / Inveigh Public

Notifications Fork 444 Star 2.5k

Code Issues 19 Pull requests 1 Actions Projects Wiki Security Insights

Files

29d9e3c

Go to file

- .github
- Inveigh
  - Listeners
  - Protocols
  - Sniffer
  - Sockets
  - Support
    - Arguments.cs
    - Control.cs
    - Output.cs
    - Shell.cs
  - FodyWeavers.xml
  - FodyWeavers.xsd
  - Inveigh.csproj
  - Program.cs
- .gitattributes
- .gitignore
- Inveigh-Relay.ps1
- Inveigh.ps1
- Inveigh.psd1
- Inveigh.psm1
- Inveigh.sln
- LICENSE
- README.md

Inveigh / Inveigh / Support / Output.cs

Kevin-Robertson interval fix, DNS AAAA 28ffe89 · 2 years ago History

Code Blame 1570 lines (1287 loc) · 57.9 KB Raw

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using System.Linq;
using System.Diagnostics;
using Quiddity.Support;

namespace Inveigh
{
    class Output
    {

        public static void OutputLoop()
        {
            bool keyDetect = true;
            bool keyPressed = false;

            do
            {

                while (Program.enabledConsoleOutput && !keyPressed)
                {

                    try
                    {

                        if (keyDetect && Console.KeyAvailable)
                        {
                            keyPressed = true;
                        }

                    }
                    catch { keyDetect = false; }

                    while (Program.consoleList.Count > 0)
                    {
                        ConsoleOutputFormat(Program.consoleList[0]);
                        Program.consoleList.RemoveAt(0);
                    }

                    if (!Program.isRunning)
                    {
                        break;
                    }

                    Thread.Sleep(5);
                }

            } while (Program.isRunning && Program.enabledConsoleOutput && Console.ReadK

        }

        public static void Queue(string Output)
        {

            lock (Program.outputList)
            lock (Program.outputList)
            {
                Program.outputList.Add(Output);
            }

        }

        public static void OutputColor(string output, string status, ConsoleColor color
        {
            string[] split = output.Substring(1).Split('[');

            foreach (string segment in split)
            {
                string[] split2 = segment.Split(']');

                int i = 0;
                foreach (string segment2 in split2)
                {
                    int j = 0;
                    if (i % 2 == 0)
                    {
                        string[] split3 = segment2.Split('|');
                        Console.Write("[");

                        foreach (string segment3 in split3)
                        {

                            if (j !=0 && j < split3.Length)
                            {
                                Console.Write("|");
                            }

                            Console.ForegroundColor = color;
                            Console.Write(segment3);
                            Console.ResetColor();
                            j++;
                        }

                         Console.Write("]");
                    }
                    else
                    {

                        if (segment2.Contains("\r\n"))
                        {
                            string[] split4 = segment2.Split('\n');

                            if (split4.Length == 2)
                            {
                                Console.Write(split4[0] + "\n");
                                Console.ForegroundColor = color;
                                Console.Write(split4[1]);
                                Console.ResetColor();
                            }
                            else
                            {
                                Console.Write(segment2);
                            }

                        }
                        else
                        {
```

```
                }

                while (Program.ntlmv1FileList.Count > 0)
                {

                    using (StreamWriter outputFileNTLMv1 = new StreamWriter(Path.Combine(Pr
                    {
                        outputFileNTLMv1.WriteLine(Program.ntlmv1FileList[0]);
                        outputFileNTLMv1.Close();

                        lock (Program.ntlmv1FileList)
                        {
                            Program.ntlmv1FileList.RemoveAt(0);
                        }

                    }

                }

                while (Program.ntlmv2FileList.Count > 0)
                {

                    using (StreamWriter outputFileNTLMv2 = new StreamWriter(Path.Combine(Pr
                    {
                        outputFileNTLMv2.WriteLine(Program.ntlmv2FileList[0]);
                        outputFileNTLMv2.Close();

                        lock (Program.ntlmv2FileList)
                        {
                            Program.ntlmv2FileList.RemoveAt(0);
                        }

                    }

                }

                while (Program.ntlmv1UsernameFileList.Count > 0)
                {

                    using (StreamWriter outputUsernameFileNTLMv1 = new StreamWriter(Path.Co
                    {
                        outputUsernameFileNTLMv1.WriteLine(Program.ntlmv1UsernameFileList[0
                        outputUsernameFileNTLMv1.Close();

                        lock (Program.ntlmv1UsernameList)
                        {
                            Program.ntlmv1UsernameFileList.RemoveAt(0);
                        }

                    }

                }
                while (Program.ntlmv2UsernameFileList.Count > 0)
                {

                    using (StreamWriter outputUsernameFileNTLMv2 = new StreamWriter(Path.Co
                    {
                        outputUsernameFileNTLMv2.WriteLine(Program.ntlmv2UsernameFileList[0
                        outputUsernameFileNTLMv2.Close();

                        lock (Program.ntlmv2UsernameFileList)
                        {
                            Program.ntlmv2UsernameFileList.RemoveAt(0);
                        }

                    }

                }

            }

        }
    }
```