

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

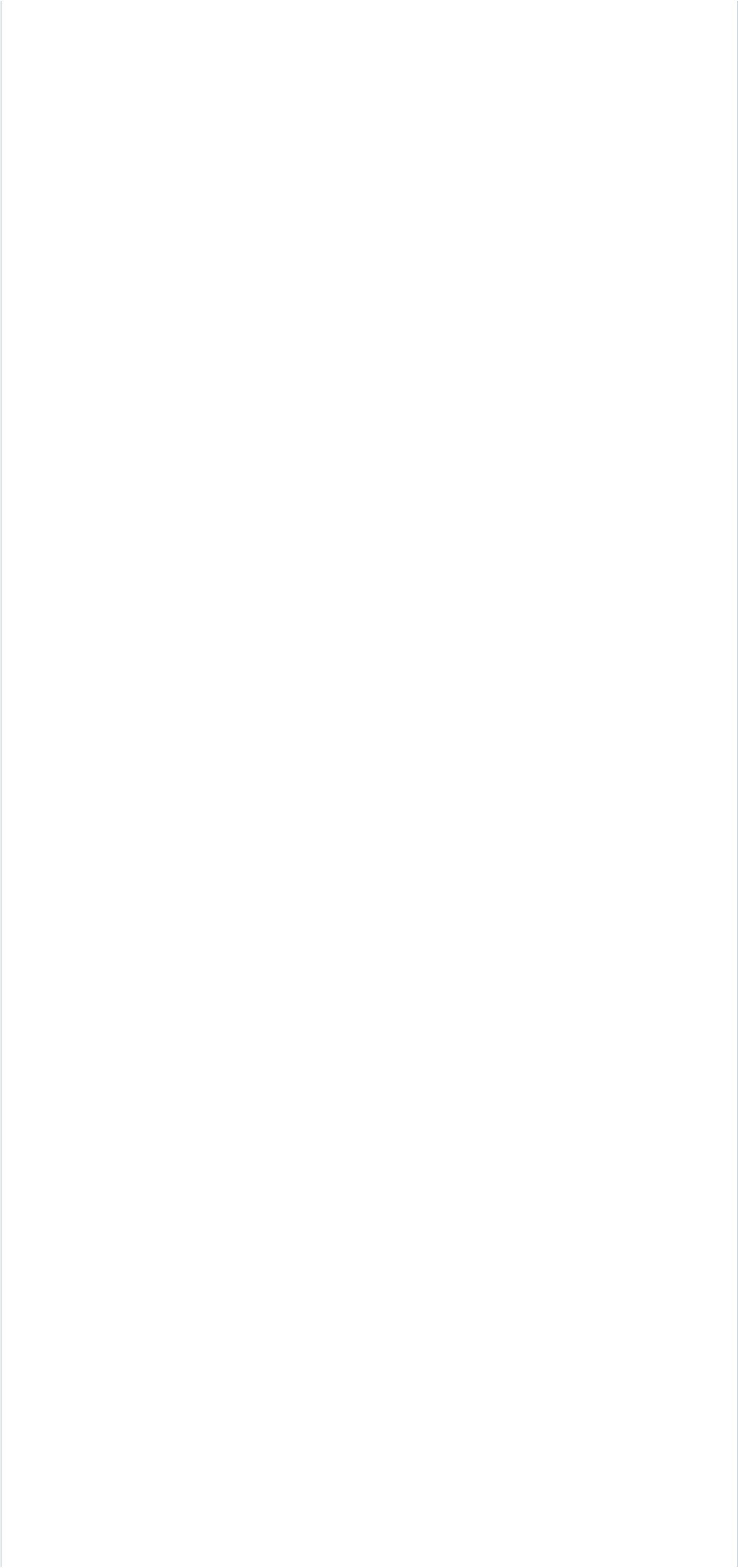
```
1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using System.Threading;
5 using System.Linq;
6 using System.Diagnostics;
7 using Quiddity.Support;
8
9 namespace Inveigh
10 {
11     class Output
12     {
13
14         public static void OutputLoop()
15         {
16             bool keyDetect = true;
17             bool keyPressed = false;
18
19             do
20             {
21
22                 while (Program.enabledConsoleOutput && !keyPressed)
23                 {
24
25                     try
26                     {
27
28                         if (keyDetect && Console.KeyAvailable)
29                         {
30                             keyPressed = true;
31                         }
32                     }
33                     catch { keyDetect = false; }
34
35                     while (Program.consoleList.Count > 0)
36                     {
37                         ConsoleOutputFormat(Program.consoleList[0]);
38                         Program.consoleList.RemoveAt(0);
39                     }
40
41                     if (!Program.isRunning)
42                     {
43                         break;
44                     }
45
46                     Thread.Sleep(5);
47                 }
48             } while (Program.isRunning && Program.enabledConsoleOutput && Console.ReadKey() != null);
49
50         }
51
52         public static void Queue(string Output)
53         {
54
55             lock (Program.outputList)
```

Page 1 of 22

```
57         LOCK (Program.outputList)
58     {
59         Program.outputList.Add(Output);
60     }
61
62 }
63
64 public static void OutputColor(string output, string status, ConsoleColor color)
65 {
66     string[] split = output.Split(' ');
67
68     foreach (string segment in split)
69     {
70         string[] split2 = segment.Split('|');
71
72         int i = 0;
73         foreach (string segment2 in split2)
74         {
75             int j = 0;
76             if (i % 2 == 0)
77             {
78                 string[] split3 = segment2.Split(' ');
79                 Console.Write("[");
80
81                 foreach (string segment3 in split3)
82                 {
83
84                     if (j != 0 && j < split3.Length)
85                     {
86                         Console.Write("|");
87                     }
88
89                     Console.ForegroundColor = color;
90                     Console.Write(segment3);
91                     Console.ResetColor();
92                     j++;
93                 }
94
95                 Console.Write("]");
96             }
97             else
98             {
99
100                 if (segment2.Contains("\r\n"))
101                 {
102                     string[] split4 = segment2.Split('\n');
103
104                     if (split4.Length == 2)
105                     {
106                         Console.Write(split4[0] + "\n");
107                         Console.ForegroundColor = color;
108                         Console.Write(split4[1]);
109                         Console.ResetColor();
110                     }
111                     else
112                     {
113                         Console.Write(segment2);
114                     }
115                 }
116             }
117             else
118             {
```









































```
1497         }
1498
1499         while (Program.ntlmv1FileList.Count > 0)
1500         {
1501
1502             using (StreamWriter outputFileNTLMv1 = new StreamWriter(Path.Combine(Pr
1503             {
1504                 outputFileNTLMv1.WriteLine(Program.ntlmv1FileList[0]);
1505                 outputFileNTLMv1.Close();
1506
1507                 lock (Program.ntlmv1FileList)
1508                 {
1509                     Program.ntlmv1FileList.RemoveAt(0);
1510                 }
1511             }
1512         }
1513
1514     }
1515
1516     while (Program.ntlmv2FileList.Count > 0)
1517     {
1518
1519         using (StreamWriter outputFileNTLMv2 = new StreamWriter(Path.Combine(Pr
1520         {
1521             outputFileNTLMv2.WriteLine(Program.ntlmv2FileList[0]);
1522             outputFileNTLMv2.Close();
1523
1524             lock (Program.ntlmv2FileList)
1525             {
1526                 Program.ntlmv2FileList.RemoveAt(0);
1527             }
1528         }
1529     }
1530
1531 }
1532
1533 while (Program.ntlmv1UsernameFileList.Count > 0)
1534 {
1535
1536     using (StreamWriter outputFileNTLMv1 = new StreamWriter(Path.Co
1537     {
1538         outputFileNTLMv1.WriteLine(Program.ntlmv1UsernameFileList[0
1539         outputFileNTLMv1.Close();
1540
1541         lock (Program.ntlmv1UsernameList)
1542         {
1543             Program.ntlmv1UsernameFileList.RemoveAt(0);
1544         }
1545     }
1546 }
1547
1548 }
```

```
1549
1550         while (Program.ntlmv2UsernameFileList.Count > 0)
1551         {
1552
1553             using (StreamWriter outputUsernameFileNTLMv2 = new StreamWriter(Path.Co
1554             {
1555                 outputUsernameFileNTLMv2.WriteLine(Program.ntlmv2UsernameFileList[0
1556                 outputUsernameFileNTLMv2.Close();
1557
1558                 lock (Program.ntlmv2UsernameFileList)
1559                 {
1560                     Program.ntlmv2UsernameFileList.RemoveAt(0);
1561                 }
1562             }
1563         }
1564     }
1565 }
1566
1567 }
1568
1569 }
1570 }
```