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 / **Control.cs**

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

Code Blame 613 lines (488 loc) · 25.3 KB Raw

```
using System;
using System.IO;
using System.Linq;
using System.Threading;
using System.Diagnostics;
using System.Text.RegularExpressions;
using System.Net;
using System.Collections.Generic;
using System.Text;
using System.Globalization;

namespace Inveigh
{
    class Control
    {

        public static void ControlLoop(string consoleLevel, int consoleQueueLimit, int
        {
            Stopwatch stopwatchConsoleStatus = new Stopwatch();
            stopwatchConsoleStatus.Start();
            Stopwatch stopwatchRunTime = new Stopwatch();
            stopwatchRunTime.Start();
            bool isPromptRefresh = false;

            while (Program.isRunning)
            {

                if (Program.cleartextList.Count > Program.cleartextCount)
                {
                    Program.cleartextCount = Program.cleartextList.Count;
                    Program.isCleartextUpdated = true;
                    isPromptRefresh = true;
                }

                if (Program.ntlmv1List.Count > Program.ntlmv1Count)
                {
                    Program.ntlmv1Count = Program.ntlmv1List.Count;
                    Program.isNTLMv1Updated = true;
                    isPromptRefresh = true;
                }

                if (Program.ntlmv2List.Count > Program.ntlmv2Count)
                {
                    Program.ntlmv2Count = Program.ntlmv2List.Count;
                    Program.isNTLMv2Updated = true;
                    isPromptRefresh = true;
                }

                IList<string> cleartextUnique = Shell.GetUnique(Program.cleartextList,

                if (cleartextUnique.Count > Program.cleartextUniqueCount)
                {
                    Program.cleartextUniqueCount = cleartextUnique.Count;
                    Program.isCleartextUniqueUpdated = true;
                    isPromptRefresh = true;
                }
```

```
                    if (Program.ntlmv1UniqueList.Count > Program.ntlmv1UniqueCount)
                    {
                        Program.ntlmv1UniqueCount = Program.ntlmv1UniqueList.Count;
                        Program.isNTLMv1UniqueUpdated = true;
                        isPromptRefresh = true;
                    }

                    if (Program.ntlmv2UniqueList.Count > Program.ntlmv2UniqueCount)
                    {
                        Program.ntlmv2UniqueCount = Program.ntlmv2UniqueList.Count;
                        Program.isNTLMv2UniqueUpdated = true;
                        isPromptRefresh = true;
                    }

                    if (isPromptRefresh && !Program.enabledConsoleOutput)
                    {
                        Shell.RefreshCurrentLine();
                        isPromptRefresh = false;
                    }
                    else
                    {
                        isPromptRefresh = false;
                    }

                    if (consoleStatus > 0 && Program.enabledConsoleOutput && stopwatchConso
                    {
                        Shell.GetCleartextUnique("");
                        Shell.GetNTLMv1Unique("");
                        Shell.GetNTLMv1Usernames("");
                        Shell.GetNTLMv2Unique("");
                        Shell.GetNTLMv2Usernames("");
                        stopwatchConsoleStatus.Reset();
                        stopwatchConsoleStatus.Start();
                    }

                    if (runTime > 0 && Program.enabledConsoleOutput && stopwatchRunTime.Ela
                    {
                        Output.Queue(String.Format("[*] {0} Inveigh is exiting due to reach
                        StopInveigh();
                    }

                    if (runCount > 0 && Program.enabledConsoleOutput && (Program.ntlmv1List
                    {
                        Output.Queue(String.Format("[*] {0} Inveigh is exiting due to reach
                        StopInveigh();
                    }

                    try
                    {
                        Output.ProcessOutput();
                    }
                    catch (Exception ex)
                    {
                        Output.Queue(String.Format("[-] [{0}] Output error detected - {1}",
                    }

                    Thread.Sleep(5);
                }

            }

```

```
                            EnabledWebDAV = true,
                            IgnoreAgents = Program.argIgnoreAgents,
                            HTTPAuth = Program.argHTTPAuth,
                            WebDAVAuth = Program.argWebDAVAuth,
                            WPADAuth = Program.argWPADAuth,
                            HTTPRealm = Program.argHTTPRealm,
                            HTTPResponse = Program.argHTTPResponse,
                            WPADResponse = Program.argWPADResponse,
                            NetbiosDomain = Program.netbiosDomain,
                            ComputerName = Program.computerName,
                            DNSDomain = Program.dnsDomain
                        };

                        Thread httpv6ListenerThread = new Thread(() => httpv6Listen
                        httpv6ListenerThread.Start();
                    }

                }

                if (Program.enabledHTTPS)
                {

                    foreach (string port in Program.argHTTPPorts)
                    {

                        HTTPListener httpsv6Listener = new HTTPListener
                        {
                            Challenge = Program.argChallenge,
                            Cert = Program.argCert,
                            CertPassword = Program.argCertPassword,
                            EnabledWebDAV = true,
                            IgnoreAgents = Program.argIgnoreAgents,
                            HTTPAuth = Program.argHTTPAuth,
                            WebDAVAuth = Program.argWebDAVAuth,
                            WPADAuth = Program.argWPADAuth,
                            HTTPRealm = Program.argHTTPRealm,
                            HTTPResponse = Program.argHTTPResponse,
                            WPADResponse = Program.argWPADResponse,
                            NetbiosDomain = Program.netbiosDomain,
                            ComputerName = Program.computerName,
                                DNSDomain = Program.dnsDomain
                            };

                            Thread httpsv6ListenerThread = new Thread(() => httpsv6List
                            httpsv6ListenerThread.Start();
                        }

                    }

                    if (Program.enabledICMPv6) // todo check linux
                    {
                        ICMPv6Socket icmpV6Socket = new ICMPv6Socket();
                        Thread icmpv6Thread = new Thread(() => icmpV6Socket.Start());
                        icmpv6Thread.Start();
                    }

                }

            }

            Thread controlThread = new Thread(() => ControlLoop(Program.argConsole, Pro
            controlThread.Start();

            if (Program.enabledFileOutput)
            {
                Thread fileOutputThread = new Thread(() => Output.FileOutput());
                fileOutputThread.Start();
            }

        }

    }

}
```