```
    ____    ____________________________________________________________________________________________________
   //¯¯\\__//¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\\___
   \\__//¯¯¯ Publish-CSharp                                                                               ___//¯¯\\
    ¯¯¯\\__________________________________________________________________________________________________//¯¯\\__//
        ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯    ¯¯¯¯
\_______________________________________________________________________________________________________________________/
  Start /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯
```

Greetings PowerShell community,

I've been working on updating my module so that I can further develop an application that merges event log collection, service configuration, registry/group policy rule implementations, process management, and general all-around system management and administration ... eventually it'll all tie back into the PowerShell Deployment modification for the Microsoft Deployment Toolkit and [FightingEntropy(π)]…

One of the utilities that I've developed is a tool called Search-WirelessNetwork, which is essentially not much different from the standard-issue tool where Windows looks for wireless networks, except this function uses an assortment of C# classes that this guy wrote ...

```
________________________________________________________________________________________________
| jcwalker/WiFiProfileManagement |                                                              |
| https://github.com/jcwalker/WiFiProfileManagement/blob/dev/Classes/AddNativeWiFiFunctions.ps1 |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

I have made some SLIGHT edits to his original code, and the reason I saw a need to implement it is mainly because using the netsh command, and parsing it, is actually a real pain in the neck and is NOT very consistent at all, as a result of the way that it outputs a STRING.

That begs the question ...

```
____________________________________________________________________________________________________________
| Q: How can PowerShell interface with native (API/DLL)’s in the operating system (or even custom ones) ... ? |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

For the wireless network utility I made, I want to access “wlanapi.dll”.
I also wanted to include it in my module so that when the module is installed, it doesn’t need any dependencies.

There aren’t any PowerShell commands that can get me extended information about wireless networks or profiles and saved passwords and stuff like that ...

Well, not to fret, everybody.
I’m not the first dude on the planet to ever ask the question…

```
____________________________________________________________________________________________________________
| Q: How can PowerShell interface with native (API/DLL)’s in the operating system (or even custom ones) ... ? |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

Because, a REAL COOL GUY, by the name of Jeffrey Snover, once upon a time, wrote this…

```
_________________________________________________________________________________________
| 04/25/06 | https://devblogs.microsoft.com/powershell/pinvoke-or-accessing-win32-apis |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

This dude Lee Holmes also talks about it…

```
_______________________________________________________________________
| 01/19/09 | https://www.leeholmes.com/powershell-pinvoke-walkthrough |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

And, ANOTHER real cool guy by the name of Adam Driscoll has been making updates to a PowerShell module called P/Invoke that basically allows this process to work as well ...

```
______________________________________________________
| 04/27/22 | https://github.com/adamdriscoll/pinvoke |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

But- there is a DRAWBACK with installing that particular module, and it doesn't change the fact that PowerShell can indeed, interface with UNMANAGED CODE such as C# or C++.

It isn't EASY, and even as Adam Driscoll states in this:

```
__________________________________________________________________
| 12/21/21 | https://blog.ironmansoftware.com/powershell-pinvoke |
¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯
```

"Platform Invoke, often shortened to P\Invoke, is the method for calling native functions from .NET.
It allows for creating signatures in a managed way for invoking non-managed functions.
They typically require a compiled class and are notoriously hard to create correctly."

So, if Adam "Ironman" Driscoll states "notoriously hard to create correctly" to manage the unmanageable…?
Uh- it probably is.

So ... since he said it, take him at his word.
The dude's been basically an expert at knowing how to write C# and PowerShell for a number of years.
Time to wave a white flag, and give up.
It’s over.
Nobody can do anything.
Everybody loses.
Nobody wins.

I’m just kidding.
Not to be dismayed, I continued to keep this idea on the backburner
“How can I use PowerShell to interface with unmanaged code…?” this, and
“How can I use PowerShell to interface with unmanaged code” that…

Just kept having dreams about it “use PowerShell to interface with unmanaged code”…

Obviously the guys at MICROSOFT know how to do that.

That’s why (cool guy 5000/Jeffrey Snover) talked about it way back in the day…

```
                                                                                                           _______/
\_________________________________________________________________________________________________________/ Start
  Script /-------------------------------------------------------------------------------------------------------\
/--------
```

Here’s the script from Jeffrey Snover himself.

```
<#
    Originally written by a man named ...
    Jeffrey Snover [MSFT] https://devblogs.microsoft.com/powershell/pinvoke-or-accessing-win32-apis/

    #######################################################################
    #  This is a general purpose routine that I put into a file called
    #   LibraryCodeGen.msh and then dot-source when I need it.
    #######################################################################
    function Compile-Csharp ([string] $code, $FrameworkVersion="v2.0.50727",
    [Array]$References)
    {
        #
        # Get an instance of the CSharp code provider
        #
        $cp = new-object Microsoft.CSharp.CSharpCodeProvider

        #
        # Build up a compiler params object…
        $framework = Combine-Path $env:windir "Microsoft.NET\Framework\$FrameWorkVersion"
        $refs = new Collections.ArrayList
        $refs.AddRange( @("${framework}\System.dll",
            "${mshhome}\System.Management.Automation.dll",
            "${mshhome}\System.Management.Automation.ConsoleHost.dll",
            "${framework}\system.windows.forms.dll",
            "${framework}\System.data.dll",
            "${framework}\System.Drawing.dll",
            "${framework}\System.Xml.dll"))
        if ($references.Count -ge 1)
        {
            $refs.AddRange($References)
        }

        $cpar = New-Object System.CodeDom.Compiler.CompilerParameters
        $cpar.GenerateInMemory = $true
        $cpar.GenerateExecutable = $false
        $cpar.OutputAssembly = "custom"
        $cpar.ReferencedAssemblies.AddRange($refs)
        $cr = $cp.CompileAssemblyFromSource($cpar, $code)

        if ( $cr.Errors.Count)
        {
            $codeLines = $code.Split("`n");
            foreach ($ce in $cr.Errors)
            {
                write-host "Error: $($codeLines[$($ce.Line – 1)])"
                $ce |out-default
            }
            Throw "INVALID DATA: Errors encountered while compiling code"
        }
    }

    ###########################################################################
    #  Here I leverage one of my favorite features (here-strings) to define
    # the C# code I want to run.  Remember – if you use single quotes – the
    # string is taken literally but if you use double-quotes, we’ll do variable
    # expansion.  This can be VERY useful.
    ###########################################################################
    $code = @'
    using System;
    using System.Runtime.InteropServices;

    namespace test
    {
        public class Testclass
        {
            [DllImport("msvcrt.dll")]
            public static extern int puts(string c);
            [DllImport("msvcrt.dll")]
            internal static extern int _flushall();

            public static void Run(string message)
            {
                puts(message);
                _flushall();
            }
        }
    }
'@

    ####################################################################
    # So now we compile the code and use .NET object access to run it.
    ####################################################################
    compile-CSharp $code
    [Test.TestClass]::Run("Monad ROCKS!")

#>
```

```
                                                                                                                            ________/
\____________________________________________________________________________________________________________________/ Script
  Time Machine /----------------------------------------------------------------------------------------------------------------\
/--------------
```

```
   __________________________________________________________________________________________________________________
   | Ok, that's all of the official Jeffrey Snover [MSFT] code up above, from 2006.                                  |
   | NOW what I'm going to do, is TRANSLATE the entire thing he wrote. In 2006.                                      |
   | Which is like, *checks watch* uh ...                                                                            |
   |                                                                                                                 |
   |  ... let's not make any assumptions here, and think like a computer processor would.                            |
   | Jeff Snover's article says April 25th, 2006, but we can't make any assumptions.                                 |
   | April is the FOURTH month. If we put it into this format: "YYYY/mm/dd", we can                                  |
   | get a real nice and clean object back from the console.                                                         |
   |                                                                                                                 |
   | Let's use PowerShell, some [DateTime] objects as well as a [Timespan] object ...                                |
   |  ... to figure out just how LONG AGO cool guy 5000 wrote this article.                                          |
   -------------------------------------------------------------------------------------------------------------------

PS Prompt:\> $Then = [DateTime]"2006/04/25"
PS Prompt:\> $Then

Tuesday, April 25, 2006 12:00:00 AM

   __________________________________________________________________________________________________________________
   | It says 12:00:00 AM. He probably did not submit this article at that exact time..                               |
   | Shame on you, cool guy 5000 ...                                                                                 |
   |  ... for not including the specific time of day in this article you wrote a real long time ago.                 |
   | I'm just kidding.                                                                                               |
   | Time to get $Now and then $Span                                                                                 |
   -------------------------------------------------------------------------------------------------------------------

PS Prompt:\> $Now = [DateTime]::Now
PS Prompt:\> $Now

Thursday, October 13, 2022 9:15:49 AM

PS Prompt:\> $Span = [TimeSpan]($Now-$Then)
PS Prompt:\> $Span

Days              : 6015
Hours             : 9
Minutes           : 15
Seconds           : 49
Milliseconds      : 889
Ticks             : 5197293498891457
TotalDays         : 6015.38599408733
TotalHours        : 144369.263858096
TotalMinutes      : 8662155.83148576
TotalSeconds      : 519729349.889146
TotalMilliseconds : 519729349889.146

   __________________________________________________________________________________________________________________
   | Wow.                                                                                                            |
   | That LOOKS like a pretty long time ...  But, how can we make this information more CONSUMABLE ... ?             |
   | I know what we could do ...  We can use MATHEMATICS to calculate this stuff, real easily.                       |
   | The problem is that you can't just divide a floating point number like the property TotalDays,                  |
   | 6015.27154000198, without error messages pointing their finger in your face ...  Nah.                           |
   |                                                                                                                 |
   | However- we can get the REMAINDER by doing this cool little operation:                                          |
   -------------------------------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $Remain = $Span.TotalDays % 365
PS Prompt:\> $Remain
175.385994087334
```

```
 ____________________________________________________________________________________________________________________
| Good. Now we can SUBTRACT that value from the TOTAL DAYS to get a number that can be cleanly                       |
| divided by (THREE-HUNDRED-AND-SIXTY-FIVE/365)...                                                                   |
 --------------------------------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $Days = $Span.TotalDays - $Remain
PS Prompt:\> $Days
5840
```

```
 ____________________________________________________________________________________________________________________
| Alright, so, NOW we can use the value 365 to divide THAT number, cleanly.                                          |
 --------------------------------------------------------------------------------------------------------------------
```

```
PS Prompt:\> $Years = $Days / 365
PS Prompt:\> $Years
16
```

```
 ____________________________________________________________________________________________________________________
| Oh boy. Ohhhhhhh boy.                                                                                              |
| It says 16. (16), full, entire, 365-day (sometimes 366), earth-years ago.                                          |
|                                                                                                                    |
| When you're doing RESEARCH AND DEVELOPMENT, you TYPICALLY want to AVOID stuff that is THAT old.                    |
| UNLESS OF COURSE, there is something REALLY IMPORTANT and NOTEWORTHY...                                            |
| ...about the thing you found. Cause, it could be a CRITICAL VULNERABILITY AND EXPOSURE, or it could                |
| lead to IDENTITY THEFT and CYBERCRIME...                                                                           |
|                                                                                                                    |
| In this case...? It just leads to an article written by a really cool guy who has dutifully worked at              |
| the Microsoft Corporation, for AT LEAST (16) years... who wrote an article about Platform Invocation               |
| (16) years ago from PowerShell, which allows PowerShell to (COMPILE/USE), CSharp code within PowerShell.           |
|                                                                                                                    |
| Which, is a REALLY COOL FEATURE of PowerShell... being able to (compile/use) C# code natively.                     |
| (That's probably about AS COOL, as programming your very own, homebrew version of BASIC on an Altair 8800.)        |
|                                                                                                                    |
| In this particular case...?                                                                                        |
| (THAT/Platform Invocation) is what's WICKED IMPORTANT and NOTEWORTHY about this article.                           |
| Cause it means that PowerShell can access UNMANAGED CODE and C# as well as C++ structs/elements/etc.               |
|                                                                                                                    |
| But- it JUST SO HAPPENS TO BE THE CASE... that there's SOMETHING ELSE, that is JUST AS WICKED IMPORTANT            |
| and NOTEWORTHY about the CONTENT of this article. This article wasn't just written by any regular,                 |
| standard, every-day, run-of-the-mill guy from Microsoft. Nah.                                                      |
|                                                                                                                    |
| The guy who wrote it happens to be the (1) guy among a team of guys...                                             |
| ...a guy who dutifully worked at the Microsoft Corporation for a number of years...                                |
| ...until eventually, one day...?                                                                                   |
| The man basically told red rover to move over.                                                                     |
| And now we have PowerShell 7.                                                                                      |
|                                                                                                                    |
| I'll add CmdLetbinding() to take care of the (Function+Parameters),                                                |
|                                                                                                                    |
| Note: Probably don't NEED to have the FrameworkVersion anymore, but I'll leave it there for added                  |
| (flexibility/functionality). It's not being used at all by the function right now, it's purely cosmetic.           |
 --------------------------------------------------------------------------------------------------------------------
```

```
                                                                                                  ______________/
\________________________________________________________________________________________________/ Time Machine
  Computer Updated /-------------------------------------------------------------------------------------------------\
/------------------
```

```
 __________________________________________________________________________________________________________________
| I'm going to change the name of the function to Publish-CSharp, because "Compile" isn't an approved verb.        |
| Note: To see the list of approved verbs, use the command "Get-Verb". I've looked at this thing hundreds          |
| of times, and really... some verbs that should exist (like "Compile"), don't. But- that's ok.                    |
 ------------------------------------------------------------------------------------------------------------------
```

```
Function Publish-CSharp
{
    [CmdLetBinding()]Param(
    [Parameter(Mandatory,Position=0)][String] $code,
    [Parameter()][String] $FrameworkVersion="v2.0.50727",
    [Parameter(Mandatory)][String[]]$References)
```

```
# // _______________________________________________________________________________________________
# // | Note: Jeff used a number of paths that use a base like ${mshhome} and ${framework} here.    |
# // | The language (Monad) has changed DRAMATICALLY since 2006 when he wrote this article, but-   |
# // | there's still something worth understanding if it can help avoid having to install a module |
# // | like PowerShell PInvoke. While I have nothing bad to say about Adam Driscoll (smart dude),  |
# // | I did not want to add the installation of an entire module to this particular application. |
# // |                                                                                             |
# // | Regardless, people have been asking me...                                                   |
# // |                                                                                             |
# // | Q: How do you know Jeffrey Snover [MSFT] told red rover to move over...?                    |
# // | A: I don't. I just say it a lot cause it rhymes, and cause Jeffrey Snover is a smart dude.  |
# // |    Otherwise, PowerShell wouldn't be the kick-ass language that it is today. So...          |
# // |                                                                                             |
# // | Q: Yeah, well... why do you like PowerShell so much, anyway...? Hm...?                      |
# // | A: Uh- cause PowerShell is HANDS DOWN... the best there is.                                 |
# // |    At some point in time...?                                                                |
# // |    People used tools to chisel into rocks to write stuff down, to get the job done.         |
# // |    At some point a real smart dude came along and invented PAPYRUS/PAPER, and INK.          |
# // |    That's when they started ripping feathers off of birds, to dip them and write stuff.     |
# // |    Then a guy eventually invented the Gutenberg press.                                      |
# // |    Then a legion of guys eventually invented the modern computer.                           |
# // |    Then an assortment of guys eventually invented the Altair 8800.                          |
# // |    Then (2) wicked smart dudes eventually invented a version of BASIC for that Altair 8800. |
# // |    Then those same (2) dudes created an industry that still exists today.                   |
# // |    At each interval, some real smart guys had to tell red rover to move over... upgrades.   |
# // |    Now...?                                                                                  |
# // |    People can just type keys on a keyboard into a PowerShell console, to get the job done.  |
# // |    Is PowerShell backward compatible with stone tablets and heiroglyphics...? Uh- nah.      |
# // |    But, it is damn close, in terms of how highly compatible it is.                          |
# // |    PowerShell...                                                                            |
# // |    For cross-compatibility...?                                                              |
# // |    Backward compatibility...?                                                               |
# // |    Forward compatibility...?                                                                |
# // |    More compatibility than you can shake a stick at.                                        |
# // |    That's saying something, isn't it...?                                                    |
# // |    Now, I might be overselling this thing...?                                               |
# // |    But- I don't think I am.                                                                 |
# // |    PowerShell. The best there is.                                                           |
# // -----------------------------------------------------------------------------------------------

If ($PSVersionTable.PSEdition -ne "Desktop")
{
    Throw "Must use Windows PowerShell v5.1 (until dependencies are implemented for PowerShell Core)"
}

# // ___________________________________________________________________________
# // | Individual assembly items from AppDomain, to pull some of the paths dynamically |
# // ---------------------------------------------------------------------------

Class AssemblyItem
{
    [Int32]      $Index
    [UInt32] $Available
    [Bool]         $GAC
    [String]   $Version
    [String]      $Name
    [String]    $Parent
    [String]  $Location
    AssemblyItem([Int32]$Index,[Object]$Assembly)
    {
        $This.Index        = $Index
        $This.GAC          = $Assembly.GAC
        $This.Version      = $Assembly.ImageRuntimeVersion
        $Path              = $Assembly.Location
        If ([System.IO.File]::Exists($Path))
        {
            $This.Name     = Split-Path $Path -Leaf
            $This.Parent   = Split-Path $Path -Parent
            $This.Location = $Path
        }
    }
    AssemblyItem([Int32]$Index,[String]$Name)
    {
        $This.Index    = $Index
        $This.Name     = $Name
        $This.Version  = "N/A"
        $This.Parent   = "N/A"
        $This.Location = "N/A"
    }
    [String] ToString()
    {
        Return $This.Location
```

```
            }
        }

        # // _________________________________
        # // | Assembly items from AppDomain |
        # // ---------------------------------

        Class AssemblyList
        {
            [Object] $Domain
            [Object] $Output
            AssemblyList()
            {
                $This.Output = @( )
                ForEach ($Assembly in [System.Appdomain]::CurrentDomain.GetAssemblies())
                {
                    $This.Output += [AssemblyItem]::New($This.Output.Count,$Assembly)
                }
            }
            [Object] Get([String]$Name)
            {
                $Item = $This.Output | ? Name -match $Name
                If ($Item)
                {
                    $Item.Available = 1
                }
                Else
                {
                    $Item = [AssemblyItem]::New(-1,$Name)
                }

                Return $Item
            }
        }

        # // ______________________________________________
        # // | Meant to (handle/track) compilation errors |
        # // ----------------------------------------------

        Class CodeError
        {
            [UInt32] $Index
            [UInt32] $Line
            [UInt32] $Column
            [String] $Label
            [String] $Message
            [UInt32] $Rank
            [Object] $Content
            CodeError([UInt32]$Index,[Object]$Line,[Object]$xError)
            {
                $This.Index   = $Index
                $This.Line    = $xError.Line
                $This.Column  = $xError.Column
                $This.Label   = $xError.ErrorNumber
                $This.Message = $xError.ErrorText
                $This.Rank    = $Line.Index
                $This.Content = $Line.Content
            }
            [String] ToString()
            {
                Return $This.Error
            }
        }

        # // _____________________________________
        # // | Divides the input code into lines |
        # // -------------------------------------

        Class CodeLine
        {
            [UInt32] $Index
            [String] $Content
            CodeLine([UInt32]$Index,[String]$Content)
            {
                $This.Index   = $Index
                $This.Content = $Content
            }
            [String] ToString()
            {
                Return $This.Content
            }
        }
```

```
# // _______________________________________
# // | Container for the code line objects |
# // ---------------------------------------

Class CodeObject
{
    Hidden [Hashtable] $Hash
    [String] $Input
    [Object] $Output
    CodeObject([String]$Code)
    {
        $This.Hash   = @{ }
        $This.Input  = $Code
        $this.Input -Split "`n" | % { $This.CodeLine($_) }
        $This.Output = $This.Hash[0..($This.Hash.Count-1)]
    }
    CodeLine([String]$Content)
    {
        $This.Hash.Add($This.Hash.Count,[CodeLine]::New($This.Hash.Count,$Content))
    }
}

# // ___________________________________________________________
# // | A class to facilitate/orchestrate the compiling process |
# // -----------------------------------------------------------

Class CodeFactory
{
    [Object] $Code
    [Object] $Assembly
    [Object[]] $References
    [Object] $Provider
    [Object] $Parameters
    [Object] $Result
    [Object] $Errors
    CodeFactory([String]$Code,[String[]]$References)
    {
        $This.Code       = [CodeObject]$Code
        $This.Errors     = @( )
        $This.Assembly   = [AssemblyList]::New()
        $This.References = $References | % { $This.Assembly.Get("$_.dll") }
        $This.Provider   = [Microsoft.CSharp.CSharpCodeProvider]::New()
        $This.Parameters = [System.CodeDom.Compiler.CompilerParameters]::New()
        $This.Parameters.GenerateInMemory   = 1
        $This.Parameters.GenerateExecutable = 0
        $This.Parameters.OutputAssembly     = "Custom"
        $List            = $This.References | ? Available
        $This.Parameters.ReferencedAssemblies.AddRange($List)
        $This.Result     = $This.Provider.CompileAssemblyFromSource($This.Parameters, $This.Code.Input)

        If ($This.Result.Errors.Count)
        {
            $xErrors     = @{ }
            Switch ($This.Result.Errors.Count)
            {
                {$_ -gt 1}
                {
                    ForEach ($X in 0..($This.Result.Errors.Count-1))
                    {
                        $This.Result.Errors[$X] | % {

                            $xErrors.Add($xErrors.Count,
                            [CodeError]::new($xErrors.Count,$This.Code.Output[$_.Line-1],$_))
                        }
                    }

                    $This.Errors = @($xErrors[0..($xErrors.Count-1)])
                }

                {$_ -eq 1}
                {
                    $This.Errors = @( $This.Result.Errors[0] | % {

                        [CodeError]::New(0,$This.Code.Output[$_.Line+1],$_)
                    })
                }
            }

            Write-Warning "Errors occurred, see property <Errors>"
        }
    }
```

```
        }

        [CodeFactory]::New($Code,$References)
    }
```

```
    ___________________________________________________________________________________________
    | Alright. So, All of that stuff above is pretty WEIGHTY and EXCESSIVE for what is really |
    | going on in the original script. However, a LOT of functionality has been added, mostly |
    | to (track/format) the (output + errors)                                                 |
    -------------------------------------------------------------------------------------------
```

```
                                                                                                    __________________/
\____________________________________________________________________________________________________/ Computer Updated
  jcwalker /----------------------------------------------------------------------------------------------------------------\
/----------
```

Normally I write ALL of my classes in PowerShell, however, you need to implement well written, strongly typed C# code into this area here, in order for this thing to actually work. I haven't really gotten this code to work for what I'm attempting to do, as I wound up using an alternate method instead which achieved the same result.

```
$Code = @"
[DllImport("wlanapi.dll", EntryPoint="WlanOpenHandle")]
public static extern uint WlanOpenHandle(
    [In] UInt32 clientVersion,
    [In, Out] IntPtr pReserved,
    [Out] out UInt32 negotiatedVersion,
    [Out] out IntPtr clientHandle
);

[DllImport("wlanapi.dll", EntryPoint="WlanCloseHandle")]
public static extern uint WlanCloseHandle(
    [In] IntPtr ClientHandle,
    IntPtr pReserved
);

[DllImport("wlanapi.dll", EntryPoint="WlanFreeMemory")]
public static extern void WlanFreeMemory(
    [In] IntPtr pMemory
);

[DllImport("wlanapi.dll", EntryPoint="WlanEnumInterfaces", SetLastError=true)]
public static extern uint WlanEnumInterfaces(
    [In] IntPtr hClientHandle,
    [In] IntPtr pReserved,
    [Out] out IntPtr ppInterfaceList
);

[DllImport("wlanapi.dll", EntryPoint="WlanGetProfileList", SetLastError=true,
CallingConvention=CallingConvention.Winapi)]
public static extern uint WlanGetProfileList(
    [In] IntPtr clientHandle,
    [In, MarshalAs(UnmanagedType.LPStruct)] Guid interfaceGuid,
    [In] IntPtr pReserved,
    [Out] out IntPtr profileList
);

[DllImport("wlanapi.dll", EntryPoint="WlanGetProfile")]
public static extern uint WlanGetProfile(
    [In] IntPtr clientHandle,
    [In, MarshalAs(UnmanagedType.LPStruct)] Guid interfaceGuid,
    [In, MarshalAs(UnmanagedType.LPWStr)] string profileName,
    [In, Out] IntPtr pReserved,
    [Out, MarshalAs(UnmanagedType.LPWStr)] out string pstrProfileXml,
    [In, Out, Optional] ref uint flags,
    [Out, Optional] out uint grantedAccess
);

[DllImport("wlanapi.dll", EntryPoint="WlanDeleteProfile")]
public static extern uint WlanDeleteProfile(
    [In] IntPtr clientHandle,
    [In, MarshalAs(UnmanagedType.LPStruct)] Guid interfaceGuid,
    [In, MarshalAs(UnmanagedType.LPWStr)] string profileName,
    [In, Out] IntPtr pReserved
);

[DllImport("wlanapi.dll", EntryPoint="WlanSetProfile", SetLastError=true, CharSet=CharSet.Unicode)]
public static extern uint WlanSetProfile(
    [In] IntPtr clientHandle,
    [In] ref Guid interfaceGuid,
    [In] uint flags,
    [In] IntPtr ProfileXml,
```

```csharp
    [In, Optional] IntPtr AllUserProfileSecurity,
    [In] bool Overwrite,
    [In, Out] IntPtr pReserved,
    [In, Out] ref IntPtr pdwReasonCode
);

[DllImport("wlanapi.dll", EntryPoint="WlanReasonCodeToString", SetLastError=true, CharSet=CharSet.Unicode)]
public static extern uint WlanReasonCodeToString(
    [In] uint reasonCode,
    [In] uint bufferSize,
    [In, Out] StringBuilder builder,
    [In, Out] IntPtr Reserved
);

[DllImport("wlanapi.dll", EntryPoint="WlanGetAvailableNetworkList", SetLastError=true)]
public static extern uint WlanGetAvailableNetworkList(
    [In] IntPtr hClientHandle,
    [In, MarshalAs(UnmanagedType.LPStruct)] Guid interfaceGuid,
    [In] uint dwFlags,
    [In] IntPtr pReserved,
    [Out] out IntPtr ppAvailableNetworkList
);

[DllImport("wlanapi.dll", EntryPoint="WlanConnect", SetLastError=true)]
public static extern uint WlanConnect(
    [In] IntPtr hClientHandle,
    [In] ref Guid interfaceGuid,
    [In] ref WLAN_CONNECTION_PARAMETERS pConnectionParameters,
    [In, Out] IntPtr pReserved
);

[DllImport("wlanapi.dll", EntryPoint="WlanDisconnect", SetLastError=true)]
public static extern uint WlanDisconnect(
    [In] IntPtr hClientHandle,
    [In] ref Guid interfaceGuid,
    [In, Out] IntPtr pReserved
);

[StructLayout(LayoutKind.Sequential, CharSet=CharSet.Unicode)]
public struct WLAN_CONNECTION_PARAMETERS
{
    public WLAN_CONNECTION_MODE wlanConnectionMode;
    public string strProfile;
    public DOT11_SSID[] pDot11Ssid;
    public DOT11_BSSID_LIST[] pDesiredBssidList;
    public DOT11_BSS_TYPE dot11BssType;
    public uint dwFlags;
}

public struct DOT11_BSSID_LIST
{
    public NDIS_OBJECT_HEADER Header;
    public ulong uNumOfEntries;
    public ulong uTotalNumOfEntries;
    public IntPtr BSSIDs;
}

public struct NDIS_OBJECT_HEADER
{
    public byte Type;
    public byte Revision;
    public ushort Size;
}

public struct WLAN_PROFILE_INFO_LIST
{
    public uint dwNumberOfItems;
    public uint dwIndex;
    public WLAN_PROFILE_INFO[] ProfileInfo;

    public WLAN_PROFILE_INFO_LIST(IntPtr ppProfileList)
    {
        dwNumberOfItems = (uint)Marshal.ReadInt32(ppProfileList);
        dwIndex = (uint)Marshal.ReadInt32(ppProfileList, 4);
        ProfileInfo = new WLAN_PROFILE_INFO[dwNumberOfItems];
        IntPtr ppProfileListTemp = new IntPtr(ppProfileList.ToInt64() + 8);

        for (int i = 0; i < dwNumberOfItems; i++)
        {
            ppProfileList = new IntPtr(ppProfileListTemp.ToInt64() + i * Marshal.SizeOf(typeof(WLAN_PROFILE_INFO)));
            ProfileInfo[i] = (WLAN_PROFILE_INFO)Marshal.PtrToStructure(ppProfileList, typeof(WLAN_PROFILE_INFO));
        }
    }
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
public struct WLAN_PROFILE_INFO
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string strProfileName;
    public WlanProfileFlags ProfileFlags;
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
public struct WLAN_AVAILABLE_NETWORK_LIST
{
    public uint dwNumberOfItems;
    public uint dwIndex;
    public WLAN_AVAILABLE_NETWORK[] wlanAvailableNetwork;
    public WLAN_AVAILABLE_NETWORK_LIST(IntPtr ppAvailableNetworkList)
    {
        dwNumberOfItems = (uint)Marshal.ReadInt64 (ppAvailableNetworkList);
        dwIndex = (uint)Marshal.ReadInt64 (ppAvailableNetworkList, 4);
        wlanAvailableNetwork = new WLAN_AVAILABLE_NETWORK[dwNumberOfItems];
        for (int i = 0; i < dwNumberOfItems; i++)
        {
            IntPtr pWlanAvailableNetwork = new IntPtr (ppAvailableNetworkList.ToInt64() + i * Marshal.SizeOf
(typeof(WLAN_AVAILABLE_NETWORK)) + 8 );
            wlanAvailableNetwork[i] = (WLAN_AVAILABLE_NETWORK)Marshal.PtrToStructure (pWlanAvailableNetwork,
typeof(WLAN_AVAILABLE_NETWORK));
        }
    }
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
public struct WLAN_AVAILABLE_NETWORK
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string ProfileName;
    public DOT11_SSID Dot11Ssid;
    public DOT11_BSS_TYPE dot11BssType;
    public uint uNumberOfBssids;
    public bool bNetworkConnectable;
    public uint wlanNotConnectableReason;
    public uint uNumberOfPhyTypes;

    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public DOT11_PHY_TYPE[] dot11PhyTypes;
    public bool bMorePhyTypes;
    public uint SignalQuality;
    public bool SecurityEnabled;
    public DOT11_AUTH_ALGORITHM dot11DefaultAuthAlgorithm;
    public DOT11_CIPHER_ALGORITHM dot11DefaultCipherAlgorithm;
    public uint dwFlags;
    public uint dwReserved;
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct DOT11_SSID
{
    public uint uSSIDLength;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 32)]
    public string ucSSID;
}

public enum DOT11_BSS_TYPE
{
    Infrastructure = 1,
    Independent    = 2,
    Any            = 3,
}

public enum DOT11_PHY_TYPE
{
    DOT11_PHY_TYPE_UNKNOWN = 0,
    DOT11_PHY_TYPE_ANY = 0,
    DOT11_PHY_TYPE_FHSS = 1,
    DOT11_PHY_TYPE_DSSS = 2,
    DOT11_PHY_TYPE_IRBASEBAND = 3,
    DOT11_PHY_TYPE_OFDM = 4,
    DOT11_PHY_TYPE_HRDSSS = 5,
    DOT11_PHY_TYPE_ERP = 6,
    DOT11_PHY_TYPE_HT = 7,
    DOT11_PHY_TYPE_VHT = 8,
    DOT11_PHY_TYPE_IHV_START = -2147483648,
    DOT11_PHY_TYPE_IHV_END = -1,
}

public enum DOT11_AUTH_ALGORITHM
{
    DOT11_AUTH_ALGO_80211_OPEN = 1,
    DOT11_AUTH_ALGO_80211_SHARED_KEY = 2,
    DOT11_AUTH_ALGO_WPA = 3,
    DOT11_AUTH_ALGO_WPA_PSK = 4,
    DOT11_AUTH_ALGO_WPA_NONE = 5,
    DOT11_AUTH_ALGO_RSNA = 6,
    DOT11_AUTH_ALGO_RSNA_PSK = 7,
    DOT11_AUTH_ALGO_WPA3 = 8,
    DOT11_AUTH_ALGO_WPA3_SAE = 9,
    DOT11_AUTH_ALGO_OWE = 10,
    DOT11_AUTH_ALGO_WPA3_ENT = 11,
    DOT11_AUTH_ALGO_IHV_START = -2147483648,
    DOT11_AUTH_ALGO_IHV_END = -1,
}

public enum DOT11_CIPHER_ALGORITHM
{
    DOT11_CIPHER_ALGO_NONE = 0,
    DOT11_CIPHER_ALGO_WEP40 = 1,
    DOT11_CIPHER_ALGO_TKIP = 2,
    DOT11_CIPHER_ALGO_CCMP = 4,
    DOT11_CIPHER_ALGO_WEP104 = 5,
    DOT11_CIPHER_ALGO_BIP = 6,
    DOT11_CIPHER_ALGO_GCMP = 8,
    DOT11_CIPHER_ALGO_GCMP_256 = 9,
    DOT11_CIPHER_ALGO_CCMP_256 = 10,
    DOT11_CIPHER_ALGO_BIP_GMAC_128 = 11,
    DOT11_CIPHER_ALGO_BIP_GMAC_256 = 12,
    DOT11_CIPHER_ALGO_BIP_CMAC_256 = 13,
    DOT11_CIPHER_ALGO_WPA_USE_GROUP = 256,
    DOT11_CIPHER_ALGO_RSN_USE_GROUP = 256,
    DOT11_CIPHER_ALGO_WEP = 257,
    DOT11_CIPHER_ALGO_IHV_START = -2147483648,
    DOT11_CIPHER_ALGO_IHV_END = -1,
}

public enum WLAN_CONNECTION_MODE
{
    WLAN_CONNECTION_MODE_PROFILE,
    WLAN_CONNECTION_MODE_TEMPORARY_PROFILE,
    WLAN_CONNECTION_MODE_DISCOVERY_SECURE,
    WLAN_CONNECTION_MODE_DISCOVERY_UNSECURE,
    WLAN_CONNECTION_MODE_AUTO,
    WLAN_CONNECTION_MODE_INVALID,
}

[Flags]
public enum WlanConnectionFlag
{
    Default = 0,
    HiddenNetwork = 1,
    AdhocJoinOnly = 2,
    IgnorePrivacyBit = 4,
    EapolPassThrough = 8,
    PersistDiscoveryProfile = 10,
    PersistDiscoveryProfileConnectionModeAuto = 20,
    PersistDiscoveryProfileOverwriteExisting = 40
}

[Flags]
public enum WlanProfileFlags
{
    AllUser = 0,
    GroupPolicy = 1,
    User = 2
}

public class ProfileInfo
{
    public string ProfileName;
    public string ConnectionMode;
    public string Authentication;
    public string Encryption;
    public string Password;
    public bool ConnectHiddenSSID;
    public string EAPType;
    public string ServerNames;
    public string TrustedRootCA;
    public string Xml;
}

public struct WLAN_INTERFACE_INFO_LIST
{
    public uint dwNumberOfItems;
    public uint dwIndex;
    public WLAN_INTERFACE_INFO[] wlanInterfaceInfo;
    public WLAN_INTERFACE_INFO_LIST(IntPtr ppInterfaceInfoList)
    {
        dwNumberOfItems = (uint)Marshal.ReadInt32(ppInterfaceInfoList);
        dwIndex = (uint)Marshal.ReadInt32(ppInterfaceInfoList, 4);
        wlanInterfaceInfo = new WLAN_INTERFACE_INFO[dwNumberOfItems];
        IntPtr ppInterfaceInfoListTemp = new IntPtr(ppInterfaceInfoList.ToInt64() + 8);
        for (int i = 0; i < dwNumberOfItems; i++)
        {
            ppInterfaceInfoList = new IntPtr(ppInterfaceInfoListTemp.ToInt64() + i * 
Marshal.SizeOf(typeof(WLAN_INTERFACE_INFO)));
            wlanInterfaceInfo[i] = (WLAN_INTERFACE_INFO)Marshal.PtrToStructure(ppInterfaceInfoList, 
typeof(WLAN_INTERFACE_INFO));
        }
    }
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
public struct WLAN_INTERFACE_INFO
{
    public Guid Guid;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string Description;
    public WLAN_INTERFACE_STATE State;
}

public enum WLAN_INTERFACE_STATE
{
    NOT_READY = 0,
    CONNECTED = 1,
    AD_HOC_NETWORK_FORMED = 2,
    DISCONNECTING = 3,
    DISCONNECTED = 4,
    ASSOCIATING = 5,
    DISCOVERING = 6,
    AUTHENTICATING = 7
}

[DllImport("wlanapi.dll", EntryPoint="WlanScan", SetLastError=true)]
public static extern uint WlanScan(
    IntPtr hClientHandle,
    ref Guid pInterfaceGuid,
    IntPtr pDot11Ssid,
    IntPtr pIeData,
    IntPtr pReserved
);
"@
```

I have not formatted that chunk of code the same way that I would normally format the document or the code.
I have been scripting and writing my code to only span so far across, in order to AVOID text wrapping.

However, some of the lines are definitely wrapping. There are plenty of techniques available to AVOID that,
but I wanted jcwalker's code to remain relatively as verbatim to what he wrote, as possible.

Still- all of that above is able to be copied and pasted into a PowerShell console, and it should work.

Any time you see INDENTATIONS in this document…?
That's because I'm adding those indentations to make the document appear more formally spaced.

If you would rather follow along from the SCRIPT to (highlight/copy/paste) verbatim code blocks…?
https://github.com/mcc85s/FightingEntropy/blob/main/Scripts/Publish-CSharp.ps1

```
                                                                                                ___________/
\______________________________________________________________________________________________/ jcwalker
  jsnover /--------------------------------------------------------------------------------------------------\
/---------
```

```
    $Refs    = "^System",
               "System.Management.Automation",
               "ConsoleHost",
               "System.Windows.Forms",
               "System.Data",
               "System.Drawing",
               "System.Xml"
    $Factory = Publish-CSharp -Code $Code -References $Refs
```

Look, even if he wrote the script (16) years ago…? Jeffrey Snover knew what the hell he was doin'. So… Here's the output. (Some of the lines below are TRUNCATED with " … ")

```
PS Prompt:\> $Refs    = "^System",
>>            "System.Management.Automation",
>>            "ConsoleHost",
>>            "System.Windows.Forms",
>>            "System.Data",
>>            "System.Drawing",
>>            "System.Xml"
PS Prompt:\> $Factory = Publish-CSharp -Code $Code -References $Refs
WARNING: Errors occurred, see property <Errors>

PS Prompt:\> $Factory

Code       : Publish-CSharp.CodeObject
Assembly   : Publish-CSharp.AssemblyList
References : {C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System\v4.0_4.0.0.0__b77a5c561934e089\System.dll,
             C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Management.Automation\…,
             C:\Windows\Microsoft.Net\assembly\GAC_MSIL\Microsoft.PowerShell.ConsoleHost\…,
             C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Windows.Forms\…,}
Provider   : Microsoft.CSharp.CSharpCodeProvider
Parameters : System.CodeDom.Compiler.CompilerParameters
Result     : System.CodeDom.Compiler.CompilerResults
Errors     : {, , , … }


PS Prompt:\> $Factory.Code

Input
-----
[DllImport("wlanapi.dll", EntryPoint="WlanOpenHandle")]…


PS Prompt:\> $Factory.Code.Output

Index Content
----- -------
    0 [DllImport("wlanapi.dll", EntryPoint="WlanOpenHandle")]
    1 public static extern uint WlanOpenHandle( [In] UInt32 clientVersion, [In, Out] IntPtr pReserved …
    2
    3 [DllImport("wlanapi.dll", EntryPoint="WlanCloseHandle")]
    4 public static extern uint WlanCloseHandle( [In] IntPtr ClientHandle, IntPtr pReserved);
    5
    6 [DllImport("wlanapi.dll", EntryPoint="WlanFreeMemory")]
    7 public static extern void WlanFreeMemory( [In] IntPtr pMemory);
    8
    9 [DllImport("wlanapi.dll", EntryPoint="WlanEnumInterfaces", SetLastError=true)]
   10 public static extern uint WlanEnumInterfaces( [In] IntPtr hClientHandle, [In] IntPtr pReserved…
    … …

PS Prompt:\> $Factory.References | Format-Table

Index Available   GAC Version    Name                                 Parent
----- ---------   --- -------    ----                                 ------
    3         1 False v4.0.30319 System.dll                           C:\Windows\Microsoft.Net\assembly\…
    2         1 False v4.0.30319 System.Management.Automation.dll     C:\Windows\Microsoft.Net\assembly\…
    1         1 False v4.0.30319 Microsoft.PowerShell.ConsoleHost.dll C:\Windows\Microsoft.Net\assembly\…
   -1         0 False N/A        System.Windows.Forms.dll             N/A
   10         1 False v4.0.30319 System.Data.dll                      C:\Windows\Microsoft.Net\assembly\…
   -1         0 False N/A        System.Drawing.dll                   N/A
    5         1 False v4.0.30319 System.Xml.dll                       C:\Windows\Microsoft.Net\assembly\…

PS Prompt:\> $Factory.Provider

FileExtension LanguageOptions Site Container
------------- --------------- ---- ---------
cs                       None

PS Prompt:\> $Factory.Parameters

CoreAssemblyFileName    :
GenerateExecutable      : False
GenerateInMemory        : True
ReferencedAssemblies    : {C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System\…\System.dll,
                          C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Management.Automation\…,
                           C:\Windows\Microsoft.Net\assembly\GAC_MSIL\Microsoft.PowerShell.ConsoleHost\…,
                           C:\Windows\Microsoft.Net\assembly\GAC_64\System.Data\… }
MainClass                :
OutputAssembly           : Custom
TempFiles                : {}
IncludeDebugInformation  : False
TreatWarningsAsErrors    : False
WarningLevel             : -1
CompilerOptions          :
Win32Resource            :
EmbeddedResources        : {}
LinkedResources          : {}
UserToken                : 0
Evidence                 :

PS Prompt:\> $Factory.Result

TempFiles                  : {}
Evidence                   :
CompiledAssembly           :
Errors                     : {Prompt:\AppData\Local\Temp\ikbp40sm\ikbp40sm.0.cs(2,22) : error CS1518: …,}
Output                     : {Prompt:\> "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe"…,}
PathToAssembly             : Custom
NativeCompilerReturnValue  : 1

PS Prompt:\> $Factory.Errors | Format-Table

Index Line Column Label  Message                                          Rank Content
───── ──── ────── ─────  ───────                                          ──── ───────
    0    2     22 CS1518 Expected class, delegate, enum, interface, or struct    1 public static extern ...
    1    2     48 CS1518 Expected class, delegate, enum, interface, or struct    1 public static extern ...
    2    2     80 CS1518 Expected class, delegate, enum, interface, or struct    1 public static extern ...
    3    2    104 CS1518 Expected class, delegate, enum, interface, or struct    1 public static extern ...
    4    2    140 CS1518 Expected class, delegate, enum, interface, or struct    1 public static extern ...
    5    5     22 CS1518 Expected class, delegate, enum, interface, or struct    4 public static extern…
```

Look. This obviously didn't work, cause there are errors there, and errors are "no bueno" when it comes to writing a file to an assembly or dll. "No bueno" means "no good". My ex-girlfriend always said that, so… "no bueno, dude."

I do not think the errors are occurring because the code is written incorrectly...
I think it is because the COMPILER in the PowerShell engine, is unable to handle the code written as it is.
I think I saw somewhere that this technique only works up until like C# version 5.0 or whatever.

However, that begs the question, if the COMPILER in the PowerShell engine is UPDATED to ACCOMMODATE the NEWER CODE… would it be able to work…?

The answer is that I don't really know, and if it's going to take me MONTHS OF RESEARCH AND DEVELOPMENT to find that answer…? It is likely a huge waste of time when there are other ways to accomplish the same end result.

And, THAT IS WHAT MATTERS.
Doesn't mean that I should permanently throw the idea into the waste bin… it just means that the ANSWER may REVEAL ITSELF at SOME LATER POINT IN TIME with a MUCH SMALLER INVESTMENT OF TIME being made.

Does that make sense…? Ok, cool.

Jeffrey Snover, as far as I can tell, never literally told red rover to move over in order to create PowerShell.
So… it is what it is.

```
/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\

Snover : Yeh I did, actually.
Me     : Oh yeah…?
Snover : Yeh.
Me     : Alright, how did you tell red rover to move over…?
Snover : Politely, wasn't mean about it or whatever.
Me     : Alright, fine.
Snover : I was just sayin'…
         If you're gonna go around saying that I told red rover to move over…
Me     : ...and now we have PowerShell 7…?
Snover : Yeh.
         It's fine with me if you keep sayin' that.
Me     : Ok.
Snover : Cool…?
Me     : Cool.
Snover : Alright.
         Keep up the good work.

\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/
```

```
                                                                                                                              _________/
\____________________________________________________________________________________________________________________________/ jsnover
```

```
  Exhibition /¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯\
/¯¯¯¯¯¯¯¯¯¯¯¯
```

When researching and developing stuff, I have to consider how much effort I need to put into one thing over
another. The real reason why I wrote this document isn't to talk about how AWESOME the script is, or whatever.
Cause, it didn't work for what I was trying to do with it.

But- suppose it did…? Would I have written this document…?
Probably not. Cause it would've worked and I would've thought nothing of it…
"Cool, the idea worked. Time to move on." -What I didn't say

The truth is, I could still reserve this idea for a different application at some point in time.
But also, look at all this information that can be learned from…?

The fact of the matter, is that I have been keeping a couple of utilities on the backburner, because I knew
that I would experience some difficulties in implementing them into the module.

It's ONE thing if errors occur and things don't work at all.
It's a TOTALLY DIFFERENT thing, if errors occur, but- something still works.
Ya know…? Did President Biden swear in front of a hot microphone when President Obama passed the
Affordable Care Act into law…?

Yeh, but- nothing broke when that happened.
Some people heard it and they looked at each other…

```
/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\

Person1 : *eyebrows lookin' like a question* I think I just heard vice president Biden say the F word…
Person2 : Yeh, I heard that too.
Person1 : Don't like, people get FIRED for sayin' the F word on a broadcast…?
Person2 : Yeh, they do.
          They probably wouldn't fire the vice president for saying the F word on accident…
Person1 : I heard it though…
Person2 : Look buddy, he said it…
          So what…?
          Get over yourself… and move on with your life, already.
          Jeez.
Person1 : I'm just sayin'… I heard it.
Person2 : Whatever dude, everybody says the F word.
          You act like you're so special or whatever…

\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/
```

Look, the point is, errors occur sometimes.
Whether it's people swearing, or errors occurring in a PowerShell script…?

Stuff happens, and you just gotta deal with it, and move on.
That's sorta what PowerShell was designed around.
CSharp has ALWAYS had an issue where if there's a SINGLE ERROR…?
Oh no.
It's not gonna work.

```
/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\

CSharp : DUDE I RAN INTO AN ERROR~!
         *throws error*
         BYE~!
Person : Uh… what the heck…?
CSharp : <See the error log to find out what the error was…>
Person : Where's the error log…?
CSharp : Buddy, you should know where it is…
         Otherwise you're a n00b.
Person : Really…?
CSharp : Yeh.
Person : But, I don't know where it is…
CSharp : Then you're a n00b.
Person : I don't even know what that is, buddy…
CSharp : WOW.
         Dude's making programs, and has no idea where the error log is, or what a n00b is.
         Unbelievable.
Person : Alright program…
         I'm feelin' pretty offended right now…
CSharp : Get skills, bro.
         You don't even know what a n00b is, here, let me Google that for ya…
         https://www.urbandictionary.com/define.php?term=n00b
         n00b: A inexperienced and/or ignorant or unskilled person. Especially used in computer games.
Person : So, you're basically saying that I don't know what I'm doing…?
CSharp : Yeh.
Person : How do I learn how to do what I'm trying to do…?
CSharp : <See the error log to find out what the error was…>
Person : Where is the error log…?
CSharp : Buddy, what do I look like, the answer man…?
         I threw an error like, before I closed the window and then I bounced.
```

Person : Yeah, but WHERE is the ERROR LOG…?
CSharp : <See the error log to find out what the error was…>, that's where.

\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/¯¯\__/

So, like… Jeffrey Snover and the team who developed PowerShell, they knew that people would get pretty frustrated, and start doing stuff like this…

```
__________________________________________________________________________________
| 06/07/09 | man destroys his keyboard and monitor | https://youtu.be/wwM3cUJKkuE |
----------------------------------------------------------------------------------
```

Yeah, so this guy probably had a program tell him what a n00b he (is/was), and then he committed computer abuse, and stormed out of the room, cause of how pissed off he was, that HE didn't know where the error log was, either.

Nah. Anyway, here's the error that comes out even though I know how to fix this ...

```
PS Prompt:\> Search-WirelessNetwork
Add-Type : Cannot add type. The type name 'WiFi.ProfileManagement' already exists.
At C:\Program Files\WindowsPowerShell\Modules\FightingEntropy\2022.10.0\FightingEntropy.psm1:37691 char:5
+     Add-Type -MemberDefinition (Use-Wlanapi) -Name ProfileManagement  ...
+     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (WiFi.ProfileManagement:String) [Add-Type], Exception
    + FullyQualifiedErrorId : TYPE_ALREADY_EXISTS,Microsoft.PowerShell.Commands.AddTypeCommand

Cannot convert argument "NegotiatedVersion", with value:
"System.Management.Automation.PSReference`1[System.Int32]",
for "WlanOpenHandle" to type "System.UInt32":
"Cannot convert the "System.Management.Automation.PSReference`1[System.Int32]" value of type
"System.Management.Automation.PSReference`1[[System.Int32, mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089]]" to type "System.UInt32"."
At C:\Program Files\WindowsPowerShell\Modules\FightingEntropy\2022.10.0\FightingEntropy.psm1:38631 char:13
+             $result                  = $This.WlanOpenHandle($maxClien ...
+             ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], MethodException
    + FullyQualifiedErrorId : MethodArgumentConversionInvalidCastArgument
```

Allow me to explain what this error is.
There are (2) errors here, the first one is that the module has already added a type for jcwalker's C# stuff.
So it's basically saying something similar to "Some dude's already sitting in that chair, dude… Can't sit there."

The second one is basically a type that I used in a class to use "Invoke-Expression" as the types that are added when the module loads, is sorta not allowing the module to do things in one fell swoop. So, what I mean is that whenever I type in a name of a function in the module, into a PowerShell console, then- if the module is NOT yet loaded, then it will automatically load, and then the function will launch.

The point is, ORDER ACTUALLY MATTERS. In this particular video…

```
____________________________________________________________________________
| 05/02/18 | Developing w/ PowerShell Classes | https://youtu.be/i1DpPU_xxBc |
----------------------------------------------------------------------------
```

... Brandon Olin (I think it is this video), says that he has a script that RECURSIVELY tries to get the types to load until they're all loaded…? But there's a better strategy than that. DO THINGS IN ORDER.
In MATHEMATICS, the ORDER OF OPERATIONS causes the MATH to DIFFER in it's OUTPUT.
The SAME EXACT CONCEPT APPLIES TO PROGRAMMING and classes, assemblies, functions, etc.

So, what I mean, is this:

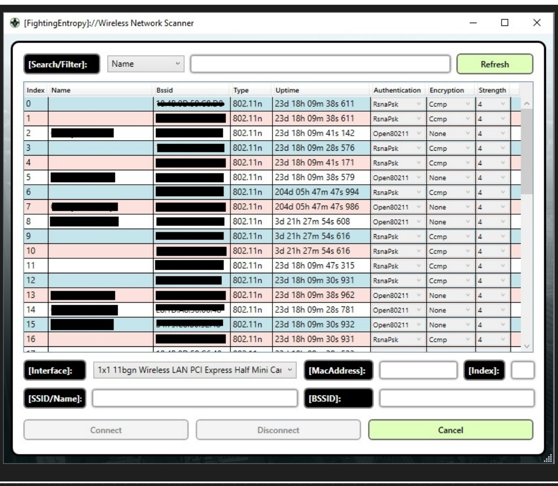
```
 2 +  2 - 4   * 4 = -12
 2 + (2 - 4)  * 4 =  -6
(2 + (2 - 4)) * 4 =   0
 2 -  2 + 4   * 4 =  16
 2 - (2 + 4)  * 4 = -22
(2 - (2 + 4)) * 4 = -16
 2 +  2 - 4   / 4 =   3
 2 + (2 - 4)  / 4 = 1.5
(2 + (2 - 4)) / 4 =   0
 2 -  2 + 4   / 4 =   1
 2 - (2 + 4)  / 4 = 0.5
(2 - (2 + 4)) / 4 =  -1
```

While this is a rather PRIMITIVE WAY to state what I mean by ORDER ACTUALLY MATTERS…?
It's primitive nature allows nearly everybody to see why they differ.
The same exact concept applies to how to load assemblies, classes, functions, etc.

PowerShell wants to load TYPES before the rest of the script, so if the types aren't there, then PowerShell's gonna start yellin' at ya, and say "HEY BUDDY… I'M NOT DOIN' THAT. SO… GET LOST~!"
So, that begs the question… what do you do, if PowerShell starts to actually yell at ya…? Put things in ORDER.
Sometimes, that PROCESS requires reading books that are 5 TIMES the length of:
1) the Bible, 2) Qua'ran, and 3) Torah COMBINED ...

I found a way to get the code to work via Add-Type, which the WAY in which a module loads ASSEMBLIES, TYPES/CLASSES, FUNCTIONS, and etc., is the thing that is NOTORIOUSLY HARD to do correctly, as stated by Adam Driscoll.

The reason being, if you want everything to run from a single command…? Then, PREREQUISITES need to be in place. So if you want to AVOID using HERE-STRINGS whenever possible, so that the CODE EDITOR can COLLAPSE chunks of code, AND, you want to make damn certain things work…? Then you have to go slightly insane in reference to where things need to be placed, so that no errors occur, and that the program finally opens…



[FightingEntropy]://Wireless Network Scanner

[Search/Filter]: Name

Index	Name	Bssid	Type	Uptime	Authentication	Encryption	Strength
0		[redacted]	802.11n	23d 18h 09m 38s 611	RsnaPsk	Ccmp	4
1		[redacted]	802.11n	23d 18h 09m 38s 611	RsnaPsk	Ccmp	4
2	[redacted]	[redacted]	802.11n	23d 18h 09m 41s 142	Open80211	None	4
3		[redacted]	802.11n	23d 18h 09m 28s 576	RsnaPsk	Ccmp	4
4		[redacted]	802.11n	23d 18h 09m 41s 171	RsnaPsk	Ccmp	4
5	[redacted]	[redacted]	802.11n	23d 18h 09m 38s 579	Open80211	None	4
6		[redacted]	802.11n	204d 05h 47m 47s 994	RsnaPsk	Ccmp	4
7	[redacted]	[redacted]	802.11n	204d 05h 47m 47s 986	Open80211	None	4
8	[redacted]	[redacted]	802.11n	3d 21h 27m 54s 608	Open80211	None	4
9		[redacted]	802.11n	3d 21h 27m 54s 616	RsnaPsk	Ccmp	4
10		[redacted]	802.11n	3d 21h 27m 54s 616	RsnaPsk	Ccmp	4
11		[redacted]	802.11n	23d 18h 09m 47s 315	RsnaPsk	Ccmp	4
12		[redacted]	802.11n	23d 18h 09m 30s 931	RsnaPsk	Ccmp	4
13	[redacted]	[redacted]	802.11n	23d 18h 09m 38s 962	Open80211	None	4
14	[redacted]	[redacted]	802.11n	23d 18h 09m 28s 781	Open80211	None	4
15	[redacted]	[redacted]	802.11n	23d 18h 09m 30s 932	Open80211	None	4
16		[redacted]	802.11n	23d 18h 09m 30s 931	RsnaPsk	Ccmp	4

[Interface]: 1x1 11bgn Wireless LAN PCI Express Half Mini Ca [MacAddress]: [Index]:

[SSID/Name]: [BSSID]:

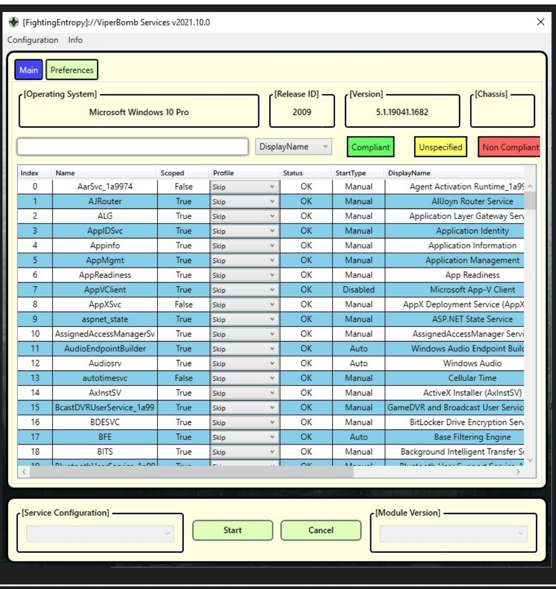
| Search-WirelessNetwork | Wireless Network Scanner | Click on picture to expand |
| 05/2022 | Video demonstration | https://youtu.be/35EabWfh8dQ |

Now, there are many other things that I've been working on, but I have to work on them in phases because I typically encounter some type of ROADBLOCK, or I wind up becoming pulled into a different direction, and thus, I have to slowly implement some of these things and then update the entire module…
Here's a video of a utility that I spent a LOT of time developing…

| 12/2021 | [FightingEntropy(π)][FEInfrastructure] | https://youtu.be/6yQr06_rA4I |

I don't imagine that MOST people will UNDERSTAND what is being showcased in that particular video, however…
… it is effectively combining server administration, networking, application development, and virtualization into a single video where I showcase the PowerShell Deployment modification for the Microsoft Deployment Toolkit, working in action, and then loading up the graphical user interface in a PXE environment.

That video demonstrates that I'm an expert at what I do. But- not ALL of the utilities are getting that much attention. At the TAIL END of that video directly above, I've been slowly developing utilities for both the CLIENT side and SERVER SIDE, environments… to help manage ALL of the child-item servers, workstations, endpoints, etc.



[FightingEntropy]://ViperBomb Services v2021.10.0

[Operating System] Microsoft Windows 10 Pro
[Release ID] 2009
[Version] 5.1.19041.1682
[Chassis]

Index	Name	Scoped	Profile	Status	StartType	DisplayName
0	AarSvc_1a9974	False	Skip	OK	Manual	Agent Activation Runtime_1a9
1	AJRouter	True	Skip	OK	Manual	AllJoyn Router Service
2	ALG	True	Skip	OK	Manual	Application Layer Gateway Serv
3	AppIDSvc	True	Skip	OK	Manual	Application Identity
4	Appinfo	True	Skip	OK	Manual	Application Information
5	AppMgmt	True	Skip	OK	Manual	Application Management
6	AppReadiness	True	Skip	OK	Manual	App Readiness
7	AppVClient	True	Skip	OK	Disabled	Microsoft App-V Client
8	AppXSvc	False	Skip	OK	Manual	AppX Deployment Service (AppX
9	aspnet_state	True	Skip	OK	Manual	ASP.NET State Service
10	AssignedAccessManagerSv	True	Skip	OK	Manual	AssignedAccessManager Servi
11	AudioEndpointBuilder	True	Skip	OK	Auto	Windows Audio Endpoint Buil
12	Audiosrv	True	Skip	OK	Auto	Windows Audio
13	autotimesvc	False	Skip	OK	Manual	Cellular Time
14	AxInstSV	True	Skip	OK	Manual	ActiveX Installer (AxInstSV)
15	BcastDVRUserService_1a99	True	Skip	OK	Manual	GameDVR and Broadcast User Servic
16	BDESVC	True	Skip	OK	Manual	BitLocker Drive Encryption Serv
17	BFE	True	Skip	OK	Auto	Base Filtering Engine
18	BITS	True	Skip	OK	Manual	Background Intelligent Transfer S

[Service Configuration]
[Module Version]

| Get-ViperBomb | Service Configuration Utility | Click on picture to expand |

That right there is a utility that was originally developed by Charles "Black Viper" Sparks, and, the graphical user interface is actually something that I collaborated on with MadBomb122. I saw something interesting about the utility and realized that it was basically the same thing as the configuration utility seen on https://blackviper.com

Black Viper is a guy who's been doing this stuff for a real long time, spanning back to the middle ages when Windows XP was the best thing being used. Middle ages is an exaggeration. The usefulness of this utility isn't an exaggeration at all, I just need to finish it.

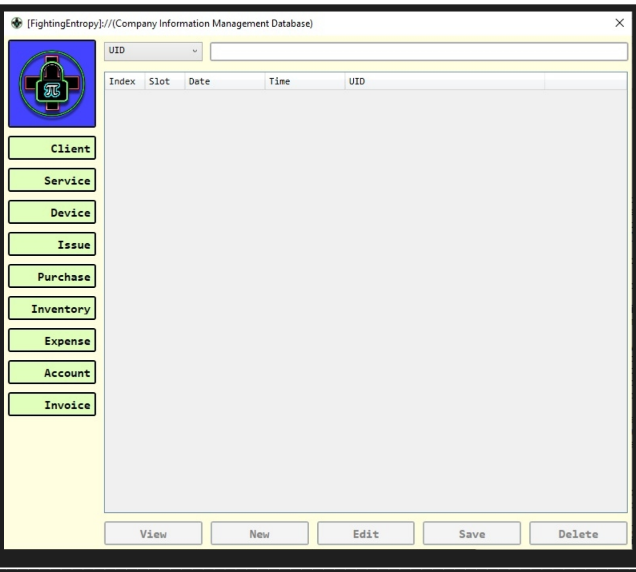
The main issue with it is that the utility is pretty laggy.
I think that's just because I'm not yet implementing things that I've since learned how to implement.

Runspacing is definitely one of those things.
However, not exactly necessary for that utility.
But also, runspacing with a GUI is actually really difficult to do correctly.
That's why I started working on a utility back in April of this year, that works exclusively with collecting all of the event logs from a particular system.

However the utility goes way farther than that, and I finally came to the conclusion that if I was going to develop an engine that works for dynamically calculating thread slots, that also provides console output to the GUI and also implements various other LOGGING and TIME information, then I was going to build it correctly from the ground up, and not waste ample amounts of time on TESTING and VALIDATION… because doing so would allow me to have a BASE that I could use on EVERY SINGLE GUI APPLICATION I EVER DEVELOP, from there forward.

That's what caused me to take a step back from the POWERSHELL DEPLOYMENT tool that I've already basically completed… the part that doesn't exactly work is the POST INSTALLATION PROCESS, which initializes a way to install the module on a target machine so that it can use these utilities natively, for importing a user profile (like Laplink/PCMover), as well as their documents, files, programs, settings, etc. But also, extending the capabilities to be more like Dism++.

So essentially, throwing in a Quake Army Knife devkit as well as a kitchen sink…
I have various other things to demonstrate in this list of links on my Github project…
https://github.com/mcc85s/FightingEntropy/blob/main/README.md



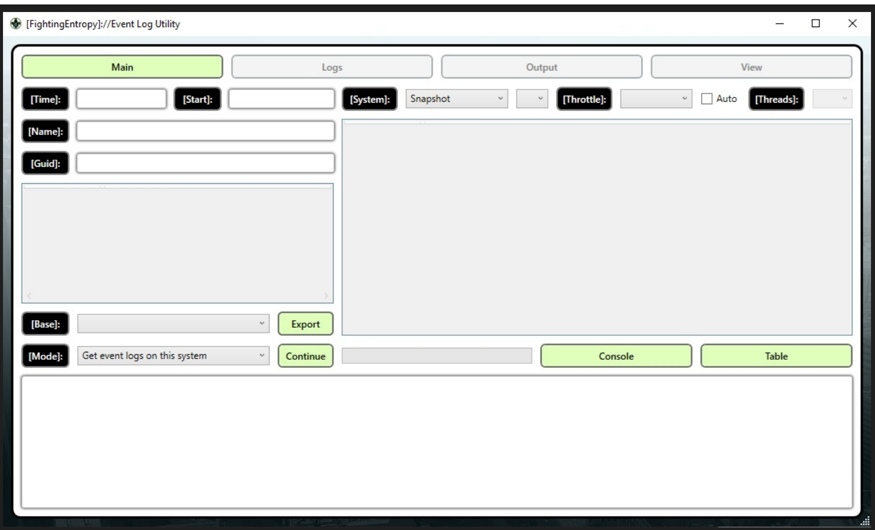
| Invoke-cimdb | Company Information Management Database | Click on picture to expand |

That is a utility that I was working on for a while to manage a customer database.

| 09/08/21 | cimdb2 | https://youtu.be/vA8_HLZ--mQ |

It is not finished. But, as you can see between the PRIOR utility ABOVE (ViperBomb) and this one…?
The STYLING is different, easier to make sense of, appears to be more well rounded…
That's because I've been slowly implementing these style changes across each utility.

However, sometimes that requires have to scrap a prior GUI I've made and just start over.
SOMETIMES, something that seems like it'd be REALLY SIMPLE… is extremely difficult to pull off without some sort of TRADEOFF being thrown into the mix. So, what I mean specifically is this utility below…



```
 ______________________________________________________________________
| Get-EventLogXaml | Event log utility | Click on picture to expand |
 ----------------------------------------------------------------------
```

I have made numerous changes to this utility, but because of how complicated it is, had to take a break from developing it over the last several months, because I felt an urgent need to write my book…

```
 ____________________________________________________________________________________
| 10/08/22 | Top Deck Awareness – Not News                                           |
| Used to be news…? Now it’s Not News. Not News. Part of the Not News Network        |
| https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2022_1008_TDA_Not_News.pdf |
 ------------------------------------------------------------------------------------
```

Writing this book, took a lot of time to do, and it is very far from completion.
Not to mention, I use a lot of foul language in it to describe my perspective of society.

It won’t appeal to everybody, and will probably offend people who aren’t expecting to understand the level of PSYCHOLOGICAL MANIPULATION that I have to unpack and examine… on a daily basis.

Most notably the notion of CLIMATE CHANGE and GLOBAL WARMING that continues to happen, because of the WAY that SOCIETY is SHAPED and MOLDED by a bunch of DECEITFUL LIARS in the government, news, and broadcast media. It also covers subjects related to ESPIONAGE, CORRUPTION, MASS SURVEILLANCE, ED SNOWDEN, JULIEN ASSANGE, BILLIONAIRES and various other topics that really… it’s a mixed bag of content.

The LAST CHAPTER, is called EXPERT PROGRAMMING 101, which I actually wrote back in May 2022, about this utility. I went on a few tangents during the lesson plan, and that is what led to writing and entire 700 page book about how messed up our society is, and when I thought I was just about done way back in May…? Uvalde Elementary occurred, and I was already knee deep in laying into guys like Ted Cruz, Tucker Carlson, and Sean Hannity.

But, the book itself is a much larger and totally different subject…
However, I’ve extracted EXPERT PROGRAMMING 101 from the book…

```
 ____________________________________________________________________________________________________
| 10/13/22 | Chapter 10 – Expert Programming 101                                                      |
| https://github.com/mcc85s/FightingEntropy/blob/main/Docs/2022_1013-(Expert%20Programming%20101).pdf |
 ----------------------------------------------------------------------------------------------------
```

Here’s the reality. People who read EXPERT PROGRAMMING 101, will get a taste of what the book is ALL ABOUT…
Even though it’s the LAST CHAPTER…? It is the introduction to the entire book.

Which is sorta how programming works, overload definitions.
You get a bunch of kids coming back from summer break who show up to the FIRST gym class of the year, and they’re gonna try and pretend like they forgot what they’re supposed to do in gym class. But that’s because the coach has to remind them that he’s not fooled…

Ya know…? First day of school, the kids often try to walk around the track after the coach says:
“Alright *blows whistle* do a couple laps.”

The kids are gonna try and pretend “I don’t know what that even MEANS, bro…” so they just start to casually walk around the track. The coach knows beforehand that he’s gonna have to say in like 5 seconds…
“*blows whistle* Yeah, I meant START RUNNING.”

Then the kids start running… begrudgingly… thinking “Hey. This sucks…”
The coach is gonna start shaking his head just like the guy in the movie Happy Gilmore, when Shooter McGavin says “Yeah right, and Grizzly Adams had a beard…” The guy isn’t shaking his head when he snaps back… “Grizzly Adams DID have a beard…” but, every other time that guy is seen in the movie, he’s shaking his head at Happy Gilmore. “Unbelievable.” -That guy

```
 Conclusion /--------------------------------------------------------------------------------------------------------\
/------------
```

Anyway, the book takes a deep look at what causes people to go into classrooms full of kids, and why they start shooting them to death. You'll never believe it, but it's because there are so many RESTRICTIONS around LANGUAGE…

Not even remotely kidding. Look back in time at this thing that Microsoft created called Tay.
https://en.wikipedia.org/wiki/Tay_(bot)
Tay was taken offline pretty quickly because it caught onto the idea of being PRETTY OFFENSIVE, real fast.

We supposedly live in a country where we have the right to free speech, right…?
It's literally written in the Constitution of the United States of America…
…and yet there are a LOT of *REALLY* offensive terms that cause people to come up with SOFT LANGUAGE.

```
_______________________________________________________________
| George Carlin | Soft Language | https://youtu.be/YLuZjpxmsZQ |
---------------------------------------------------------------
```

The problem is that SOME LANGUAGE IS OFFENSIVE FOR SOME REASON~!
Like comedian Bill Burr argues, the people on the news say "There's NO REASON to hit a woman"…
…but, what if she's carrying a KNIFE, and… she's trying to STAB you…?
Is it ok to hit a woman THEN…? Yeh. It is. That's gonna be seen as a "GOOD ENOUGH REASON" across the board.

People in our society will go ahead and try to make up excuses for the woman, like "How do you know she was trying to stab you with that knife…? Hm…? You got any PROOF that she was chasing you trying to stab you…? If not…? Domestic violence charges, you evil prick~!"

They may even say "Well, you know what I meant…" But, nah. Sometimes if a woman is chasing you around with a knife in her hand, trying to stab you…? There's a damn good reason to hit a woman after all… she's psychotic.

The REASON NEVER GOES AWAY no matter HOW MANY TIMES PEOPLE CHANGE THE TERMS AROUND TO SOUND LESS OFFENSIVE.

The reason why some language is OFFENSIVE is because it causes STRESS, and that STRESS whereby causes people to experience a sense of either DREAD, or ANXIETY… (or whatever, really) and NO MATTER HOW MANY TIMES THE TERMS ARE CHANGED…? …the stress will ALWAYS RESURFACE.

So, people don't realize how stupid it is to NOT TALK ABOUT STUFF, or to REMAIN QUIET or SILENT.
See no evil, Hear no evil, Speak no evil…

It's not unlike when someone goes right ahead, and just rips one of the nastiest, grossest sounding farts in the history of mankind, right…? Everybody knows when they hear it, the SMELL is gonna come at some point…
So, people will immediately run to their nearest can of air freshener, expecting the smell to be quite horrid, and they'll even spray this stuff AHEAD OF TIME, expecting the smell to override their senses.

Our society has a serious problem with trying to change the way that "FARTS" occur in LANGUAGE.
But I've got NEWS for ya… it's pretty stupid for people to do that, and society constantly teaches people stuff like "OOOohHHhh that person SWORE or said something REALLY BAD, and that means, that person is a real EVIL PERSON, so… don't EVER respond to that person EVER, or else…? You'll die from something evil…"

See how STUPID that line of reasoning is…?
Everybody farts.
Everybody swears.
Everybody lies.
Everybody does something pretty messed up or considered otherwise BAD. Why…?

Well, religion sorta plays a role here, "God is either ALL KNOWING or ALL GOOD."
I don't expect everybody's gonna enjoy reading what I think about that…?
But- here's how it applies to right here and now.

It is PRETTY STUPID TO ASSUME, that (GOD/PEOPLE) (is/are) ALL GOOD.
Why…? Because GOD, and (SATAN/THE DEVIL), they ALWAYS KNOW THE SAME AMOUNT OF STUFF AS THE OTHER.
Same goes for people.
In order for (GOD/PEOPLE) to be ALL GOOD…? (He has/They have) to be somewhat STUPID.
In order for (GOD/PEOPLE) to be ALL KNOWING…? (He has/They have) to be somewhat EVIL.

People that give others a benefit of a doubt, are exercising FAITH, that those people aren't EVIL AT ALL.
But, that's a pretty stupid way to look at it sometimes, especially if someone keeps saying something bad about them. On the contrary, just because people say bad things about certain people…
…DOES NOT NECESSARILY MEAN THAT THOSE BAD THINGS ARE TRUE.

The problem lies in CHECKING… and that's called VALIDATION.
And, it's like QUANTUM PHYSICS and SUPERPOSITION/ENTANGLEMENT.

The double slit experiment says that (2) cats are in their own respective boxes, and that each of the cats in each of their respective box is both ALIVE as well as DEAD, at the same exact time. But that is impossible…
…right…?

Well, according to QUANTUM PHYSICS, no, that is not impossible at all.
Until you (CHECK each box/VALIDATE the experiment)…? That is the reality, both scenarios are EQUALLY TRUE/FALSE.
It's just that people are typically LAZY, or don't want to put much effort into stuff. That's called SLOTH.
One of the SEVEN DEADLY SINS… just being wicked lazy, is SLOTH.

Nobody is all good. That's what our SOCIETY wants to teach people, is that in order to be a ROLE MODEL CITIZEN, you gotta ALWAYS BE PERFECT IN FRONT OF PEOPLE, like basically everybody on CAPITAL HILL. And then SCANDALS ensue, and then… it's just a circle of stupidity that never seems to end, a dog, endlessly chasing it's tail…

Will the dog that starts chasing it's own tail EVER GET THE GOD DAMN THING…? No. But- whoever thought it was a good idea for the FCC to establish FINES to ELIMINATE SWEARS from PUBLIC BROADCAST…?

They either: 1) don't realize how stupid that is, OR, 2) they've imposed those guidelines for an EVIL REASON.

Simply put, the STRESSES that result from OFFENSIVE TERMS that are converted into SOFT LANGUAGE and stuff like that, they'll take terms like SHELL SHOCK, and then turn it into POST-TRAUMATIC STRESS DISORDER… right?

But- the reality is that even if you change the term POST-TRAUMATIC STRESS DISORDER…
… what CAUSES things like POST-TRAUMATIC STRESS DISORDER…
… are STIGMAS or things that are "OFF LIMITS". And, THEY WILL NEVER GO AWAY.
It's like needing an infinite supply of air freshener… people keep thinking "I'll make a WAY BETTER AIR FRESHENER that lasts FOREVER…" but that person is an idiot to think they'll ever do that, and FURTHER TO THAT POINT…?
Trying to MASK something that's meant to exist, is the height of human stupidity, it really is.

Like, it'll resurface as the KARZAI BOMBING in Afghanistan.
Or, it'll cause the WORLD TRADE CENTER to be attacked by TERRORISTS…
Or, it'll cause some moron who calls himself the PRESIDENT of RUSSIA, to ATTACK UKRAINE and tell his own countrymen, that they gotta facilitate a RESCUE OPERATION to RESCUE the CITIZENS of UKRAINE from the clutches of the EVIL FASCIST NAZI'S that took over.

It sounds STUPID, right…? Well, think again, dude.
VLADIMIR PUTIN doesn't think that's stupid at all. He thinks YOU'RE stupid, for thinking THAT is stupid.
And, he's more important than basically 100% of the population on planet Earth. In his eyes.

But, stupidity will do that to ya. Here's the silver lining.
If you EXPECT that people like VLADIMIR PUTIN will do stupid things…?
Then you can teach everybody else how PREDICTABLE stupid people happen to be, by TALKING ABOUT STUFF, no matter how "BAD" the language is. And that is the MAIN REASON why the CONSTITUTION OF THE UNITED STATES OF AMERICA, has FREEDOM OF EXPRESSION as it's FIRST, #1 amendment, in the BILL OF RIGHTS.

```
                                                                                                          ____________/
\_________________________________________________________________________________________________________/ Conclusion

________________________________________________________
|------------------------------------------------------|
|                                   Michael C. Cook Sr. |
|                                     Security Engineer |
|                                  Secure Digits Plus LLC |
|______________________________________________________|
--------------------------------------------------------
```