etw-event-dumper

etw-event-dumper is designed for bulk collection of ETW event data for research purposes, in particular those scenerios where you know that there must be some relevant data in the ETW traces but you don't know what.

I have hooked into the ETW collection for a previous project where I needed to extract Sysmon data using the ETW provider, but that was only using one provider. Whilst looking for examples of multiple provider support, I found the SilkETW project created by @fuzzysec, which provided an excellent base.

The SilkETW service component allows multiple providers, but has too much overhead in creating the config, as I wanted to hook over 100 providers, and I would rather write my own tool that works exactly for my needs.

I have **borrowed** the event XML parsing code from SilkETW, which flattens the XML data into a Key/Value list, along with the JSON serialisation, so thanks @fuzzysec! 😃

Implementation

The aim of the tool is to allow bulk collection for very large numbers of ETW providers. The code is written using .Net 6, and the new console application format.

ETW

If you are investigating a particular application, then you can use the **logman** tool to identify the provider GUID's implemented for a particular process via it's PID:

```
PS D:\> logman query providers -pid 3812

Provider                                 GUID
-------------------------------------------------------------------------------
.NET Common Language Runtime             {E13C0D23-CCBC-4E12-931B-D9CC2EEE27E4}
FWPUCLNT Trace Provider                  {5A1600D2-68E5-4DE7-BCF4-1C2D215FE0FE}
Microsoft-Antimalware-Scan-Interface     {2A576B87-09A7-520E-C21A-4942F0271D67}
Microsoft-IEFRAME                        {5C8BB950-959E-4309-8908-67961A1205D5}
Microsoft-Windows-Application-Experience {EEF54E71-0661-422D-9A98-82FD4940B820}
Microsoft-Windows-AppModel-Runtime       {F1EF270A-0D32-4352-BA52-DBAB41E1D859}
Microsoft-Windows-AsynchronousCausality  {19A4C69A-28EB-4D4B-8D94-5F19055A1B5C}
Microsoft-Windows-COM-Perf               {B8D6861B-D20F-4EEC-BBAE-87E0DD80602B}
```

It is highly likely that the provider GUID list will have numerous entries that have a GUID as the provider name, I think you will probably want to remove these from the set of the providers you hook, but you can decided that on a case by case basis.

```
{F883EB01-AE8F-486B-82A0-077DF8F6101D}   {F883EB01-AE8F-486B-82A0-077DF8F6101D}
{FB2A6D3F-0896-532E-77FC-2D9191D1C717}   {FB2A6D3F-0896-532E-77FC-2D9191D1C717}
{FF32ADA1-5A4B-583C-889E-A3C027B201F5}   {FF32ADA1-5A4B-583C-889E-A3C027B201F5}
```

Usage

- Provider GUID's must be supplied via text file, with each GUID on a new line, using the **-p** parameter (Required)

```
-p .\providerguids.txt
```

- Output file for event dump using the **-o** parameter
- Filter on PID or Process Name using the **-ft** and **-fv** parameters

```
-ft pid -fv 1234
-ft proc -fv "Outlook"
```

- Filter on Event Names using the **-en** parameter, using a text file containing the individual Event Names, with each Event Name on a new line

```
-en .\eventnames.txt
```

Full command line example:

```
-p .\providers.txt -o output.txt -en .\eventkeywords.txt -ft pid -fv 1234
```