0.0.1 Using ETW

As mentioned in ?? (??), the goal of this project is to research the possibilities of using built in telemetry, such as Event Tracing for Windows (ETW), to detect the exploitation of vulnerabilities. Therefore, it is important to discover how ETW can be used to gather telemetry from providers.

One ETW provider that is widely used to detect malicious activity such as exploitation of vulnerabilities is the `Microsoft_windows-Threat-Intelligence`[1] provider. This is widely used by various Antivirus (AV) engines such as Microsoft's own Endpoint Detection and Response (EDR)/AV tool, Microsoft Defender for Endpoint. While this provider gives insight into Windows Application Programming Interface (API) calls often used in an exploitation process, we will not be focusing on this. As mentioned in ?? (??) and discussed in ?? (??), the project will revolve around detection of CVE-2020-24086, which is a vulnerability in the `tcpip.sys` driver of Windows. Due to this, we will in this section explore ETW providers relevant to this specific driver.

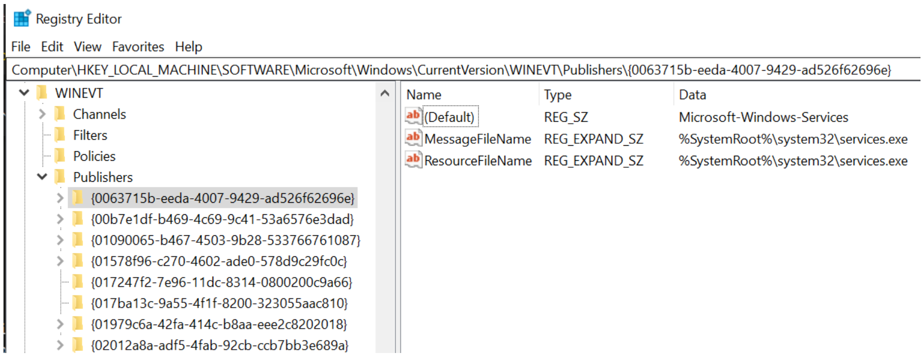
Finding providers

Read this section thoroughly as it has been revised a lot ad-hoc

Getting a list of all available providers in Windows is fairly simply. A few methods exists, such as:

1. Using `logman query providers`
2. Using the PowerShell command `Get-TraceEtwProvider`
3. Enumerate registry keys under `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers`

The output from method *(3)*, the registry, is as mentioned in ?? (??), only for manifest-based providers. Therefore, not all providers will be shown here. Figure 1 (Finding ETW providers using Registry Editor) shows how the information available using Registry Editor. As it can be seen the registry contains information about the binary the provider is implemented in, which in our case is `tcpip.sys`.



Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{0063715b-eeda-4007-9429-ad526f62696e}

- WINEVT
  - Channels
  - Filters
  - Policies
  - Publishers
    - {0063715b-eeda-4007-9429-ad526f62696e}
    - {00b7e1df-b469-4c69-9c41-53a6576e3dad}
    - {01090065-b467-4503-9b28-533766761087}
    - {01578f96-c270-4602-ade0-578d9c29fc0c}
    - {017247f2-7e96-11dc-8314-0800200c9a66}
    - {017ba13c-9a55-4f1f-8200-323055aac810}
    - {01979c6a-42fa-414c-b8aa-eee2c8202018}
    - {02012a8a-adf5-4fab-92cb-ccb7bb3e689a}

Name	Type	Data
(Default)	REG_SZ	Microsoft-Windows-Services
MessageFileName	REG_EXPAND_SZ	%SystemRoot%\system32\services.exe
ResourceFileName	REG_EXPAND_SZ	%SystemRoot%\system32\services.exe

Figure 1: Finding ETW providers using Registry Editor

To find all manifest-based providers we can use the PowerShell script on listing 1, where the output of the command is also shown.

```
Get-ChildItem -Path "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers"
    | Get-ItemProperty
    | Where-Object {$_.ResourceFileName -like '*tcpip.sys*'}
    | Format-List "(default)", ResourceFileName, MessageFilename, PSChildName

(default)        : Microsoft-Windows-TCPIP
ResourceFileName : C:\WINDOWS\system32\drivers\tcpip.sys
MessageFileName  : C:\WINDOWS\system32\drivers\tcpip.sys
PSChildName      : {2f07e2ee-15db-40f1-90ef-9d7ba282188a}
```

Listing 1: `logman query providers` output. See appendix ?? for full output

The same information queried using `logman query providers` can be seen in listing 2. However, the `logman query providers` command does not display the *ResourceFileName* as it can be seen on Figure 1 (Finding ETW providers using Registry Editor).

```
 1  Provider                                 GUID
 2  -------------------------------------------------------------------------------
 3  .NET Common Language Runtime             {E13C0D23-CCBC-4E12-931B-D9CC2EEE27E4}
 4  ACPI Driver Trace Provider               {DAB01D4D-2D48-477D-B1C3-DAAD0CE6F06B}
 5  Active Directory Domain Services: SAM    {8E598056-8993-11D2-819E-0000F875A064}
 6  Active Directory: Kerberos Client        {BBA3ADD2-C229-4CDB-AE2B-57EB6966B0C4}
 7  Active Directory: NetLogon               {F33959B4-DBEC-11D2-895B-00C04F79AB69}
 8  ADODB.1                                  {04C8A86F-3369-12F8-4769-24E484A9E725}
 9  ADOMD.1                                  {7EA56435-3F2F-3F63-A829-F0B35B5CAD41}
10  Application Popup                        {47BFA2B7-BD54-4FAC-B70B-29021084CA8F}
11  Application-Addon-Event-Provider         {A83FA99F-C356-4DED-9FD6-5A5EB8546D68}
12  ...
13  Microsoft-Windows-TCPIP                  {2F07E2EE-15DB-40F1-90EF-9D7BA282188A}
14  ...
15  TCPIP Service Trace                      {EB004A05-9B1A-11D4-9123-0050047759BC}
16  ...
```

Listing 2: `logman query providers` output. See appendix **??** for full output

The number of providers registered according to `logman query providers` is 1162 whereas 972 of these are manifest-based providers according to data from the registry. One example of a non-manifest-based provider that can only be found using logman is the provider *TCPIP Service Trace* as seen on line 15 in listing 2.

Using the second method, `Get-TraceEtwProvider`, simply yields a `Access Denied` error rendering it useless.

To conclude this section, we were able to find two different providers related to the TCP/IP stack on Windows. The first provider, *Microsoft-Windows-TCPIP* is definitely related to `tcpip.sys` according to the *ResourceFileName* property. The second however is a bit more cumbersome as logman does not provide any more information than the name (*TCPIP Service Trace*) and the GUID.

Starting a trace

In the previous sections we have explored the different components of ETW and how they work together. This section will showcase how to easily consume ETW events for specific providers. Once again, we are going to be using the *Microsoft-Windows-TCPIP* provider as an example, as this is the provider we will be exploring later on. To start a trace for *Microsoft-Windows-TCPIP* we need to take the following steps:

1. Start a new trace session that will hold our buffers
2. Add a provider to our trace session
3. Ensure that we can consume events produced in our trace, either through an event log or realtime

```
New-EtwTraceSession -Name TCPIPTrace -LogFileMode 0x8100
↪  -FlushTimer 1
Add-EtwTraceProvider -SessionName TCPIPTrace -Guid
↪  '{2F07E2EE-15DB-40F1-90EF-9D7BA282188A}' -MatchAnyKeyword 0x0
tracefmt -rt TCPIPTrace -displayonly
```

Listing 3: Starting a trace for *Microsoft-Windows-TCPIP - 2F07E2EE-15DB-40F1-90EF-9D7BA282188A*

In these three simple steps, we will be acting as the ETW controller through PowerShell to start a trace of *Microsoft-Windows-TCPIP*. Listing 3 shows how to do this using PowerShell and the `tracefmt` tool from Windows Driver Kit (WDK).

The code is explained here:

Explain bitmask

New-EtwTraceSession Starts a new trace with the name *TCPIPTrace* and sets the `LogFileMode` to `EVENT_TRACE_USE_LOCAL_SEQUENCE` and `EVENT_TRACE_REAL_TIME_MODE`[2]. The `FlushTimer` argument states that the buffer should be flushed every second

Add-EtwTraceProvider Adds the *Microsoft-Windows-TCPIP* provider to the session matching any keywords (ie. we will consume all event keywords)

Maybe explain event keywords, and also how to get them

tracefmt Displays the content of the trace in the current console every time the buffer is flushed

At this point we are able to consume all events produced by `tcpip.sys`, and are therefore at a good position to consider how the data can be used for detection. However, in order to know exactly what to detect, it is important to first analyze CVE-2021-24086. In the next **??** (**??**) we will begin analyzing CVE-2021-24086 in order to understand exactly what we need to detect.

Rewrite this at some point