Capturing Network Traffic Using Built-In Windows Tools

A White Paper From

For more information, see our web site at
http://www.goldstarsoftware.com

Capturing Network Traffic Using Windows Tools

Last Updated: 02/15/2024

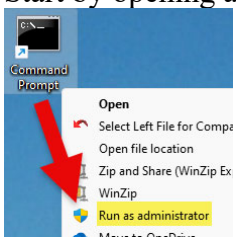
For decades now, we have been recommending the use of the open-source utility known as Wireshark to capture network packets on systems that are experiencing slowness and other types of issues where you want to see what is actually transpiring between two computers. However, some users have balked about needing to install a separate component to capture this data, especially when the server is under strict controls for change management.

Recent improvements to the built-in tools may actually make Wireshark no longer critical for capturing the data on a Windows workstation or server. (It can still be quite helpful when *analyzing* the data, but that is a completely different topic.) To that end, we are now able to provide instructions on how to use the Windows tools to capture your network data.

Capturing an ETL File with NETSH

If you just need to grab a quick network trace while duplicating an error or other specific conditions, then this process may be the quickest option for you. By default, this sequence will give you a trace file with the last 250MB of data. You can add the parameter "maxSize=64" if you want to capture a smaller block of data, or you can even go larger if needed.

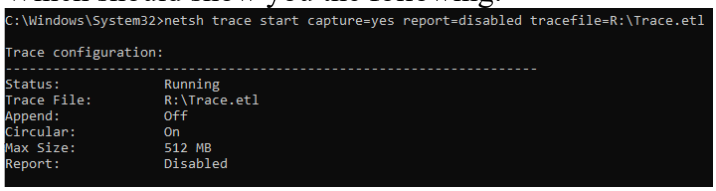
1. Start by opening a Command Prompt window "As Administrator":



2. Start the trace with the following command:

```
netsh trace start capture=yes report=disabled tracefile=R:\Trace.etl
```

Which should show you the following:



3. Duplicate your issue (i.e. run the application as needed).
4. Stop the trace with this command:

```
netsh trace stop
```

Which should show you the following:

```
C:\Windows\System32>netsh trace stop
Merging traces ... done
Tracing session was successfully stopped.
```

Note that this example is writing the data to a file on the root of the R: drive (a RAMdisk on my system). Obviously, you should provide a valid path to a working folder with enough disk space to hold the trace data.

Now, it is important to note that an ETL file is NOT a standard packet trace file that is readable by Wireshark or other such tools. To make this file readable, you must first convert it to a PCAPNG file, as described in the next section.

Converting an ETL File to a PCAPNG File

Once you have the ETL file, you will need to translate that file to make it easier to read. This does require a third-party download, so your first task is to download the tool from this link:

https://github.com/microsoft/etl2pcapng

Follow the link for the "prebuilt binaries" and download the latest release from there, then copy the EXE file into your path (such as in your C:\Windows folder). As of this writing, 1.11.0 was the current release, so we will show screens from that version.

With that downloaded and available in your path, a single command does the job for you:

```
etl2pcapng Trace.etl Trace.pcapng
```

Which should generate a simple screen back:

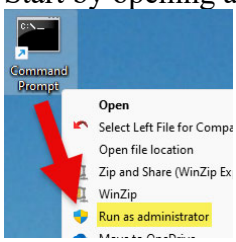
```
R:\>etl2pcapng Trace.etl Trace.pcapng
IF: medium=eth                ID=0    IfIndex=8       VlanID=0
Wrote 200815 frames to Trace.pcapng
```

You can now open this file with Wireshark for further analysis.

Capturing an ETL File with PKTMON

Newer operating systems, such as Windows 10, Server 2019 (and newer), also include a tool called **pktmon** which can handle everything within a single tool (and eliminate the need for the **etl2pcapng** tool). This process is quite similar to the above, but the options are a bit different, and some may find it a bit more complicated.

1. Start by opening a Command Prompt window “As Administrator”:



2. Display a list of the available network adapters in the system with the following command:

```
pktmon list
```

Which should show you a screen like the following:

```
C:\Windows\System32>pktmon list

Network Adapters:
   Id MAC Address       Name
   -- -----------       ----
   11 A4-BB-6D-D3-3A-FD Intel(R) Ethernet Connection (5) I219-LM
   13 00-50-56-C0-00-01 VMware Virtual Ethernet Adapter for VMnet1
   10 98-B7-85-1E-B8-AE Intel(R) Ethernet Controller X550
    9 98-B7-85-1E-B8-AF Intel(R) Ethernet Controller X550 #2
   12 00-50-56-C0-00-08 VMware Virtual Ethernet Adapter for VMnet8
```

3. Review the various network adapters listed and find the ID number of the one you wish to capture from. In the above example, we have a single GbE NIC (11) and a dual-port 10GbE NIC with two ports to choose from, 9 and 10. (Sometimes you have to experiment a bit and try one, and f it doesn’t see the network traffic you want, try another.) In our case, we are going to capture on ID #10.
4. Start the trace with a command like this:

```
pktmon start --capture --comp 10 --pkt-size 0 --file-name R:\trace.etl
```

Which should show you the following:

```
C:\Windows\System32>pktmon start --capture --comp 10 --pkt-size 0 --file-name R:\trace.etl

Logger Parameters:
    Logger name:        PktMon
    Logging mode:       Circular
    Log file:           R:\trace.etl
    Max file size:      512 MB
    Memory used:        640 MB

Collected Data:
    Packet counters, packet capture

Capture Type:
    All packets

Monitored Components:
    Id Driver       Name
    -- ------       ----
    10 ixs68x64.sys Intel(R) Ethernet Controller X550

Packet Filters:
    None
```

5. Duplicate your issue (i.e. run the application as needed).
6. Stop the trace with this command:

```
pktmon stop
```

Which should show you the following:

```
C:\Windows\System32>pktmon stop
Flushing logs...
Merging metadata...
Log file: R:\trace.etl (No events lost)
```

As with most tools, there is a bevy of available options to customize each of these settings, such as the ETL file path (here shown on the R: drive, so make sure you are

using a valid drive letter on your own system), the number of bytes to save from each packet, the maximum ETL file size to collect, and more. This command will show you all of the command line options of the **pktmon** tool:

```
pktmon start help
```

The default trace file is 512MB, which is fairly large, but this is a circular buffer that will containb the LAST 512MB of the trace, which is quite useful. If you wanted to capture a set of much smaller files (say 64MB each), then you could add options like this:

```
--file-size 64 --log-mode multi-file
```

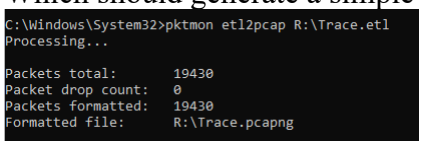
This would tell pktmon to keep tracing and spawning a new file every 64MB. Unlike Wireshark, though, you cannot configure it to automatically overwrite the oldest fiole beyond a certain point, so make sure you have enough disk space!

Converting an ETL File to a PCAPNG File

The **pktmon** tool also has a built-in conversion function to generate the PCAPNG file for you. This command does the job for you:

```
pktmon etl2pcap Trace.etl
```

Which should generate a simple screen back:



Note that this created the Trace.pcapng file for you based on the source filename. If needed, you can use specify the output file with the –out option. If you'd like to see additional options for the data conversion function, use this command:

```
pktmon etl2pcap help
```

What's Next?

Once you have the network capture in a PCAPNG format, you'll be able to open that up with Wireshark or your favorite network analyzer and start to understand what is going on under the covers.

If you intend to use these traces with the **BtrvInterceptor** and **SQLInterceptor** tools from Goldstar Software, note that these tools require the older PCAP format, so you may need to perform one more conversion to that format first.

If you still can't get it to work, contact Goldstar Software and let us work with you to help! Please note that this may be a billable support call if you have already used up your free support time.