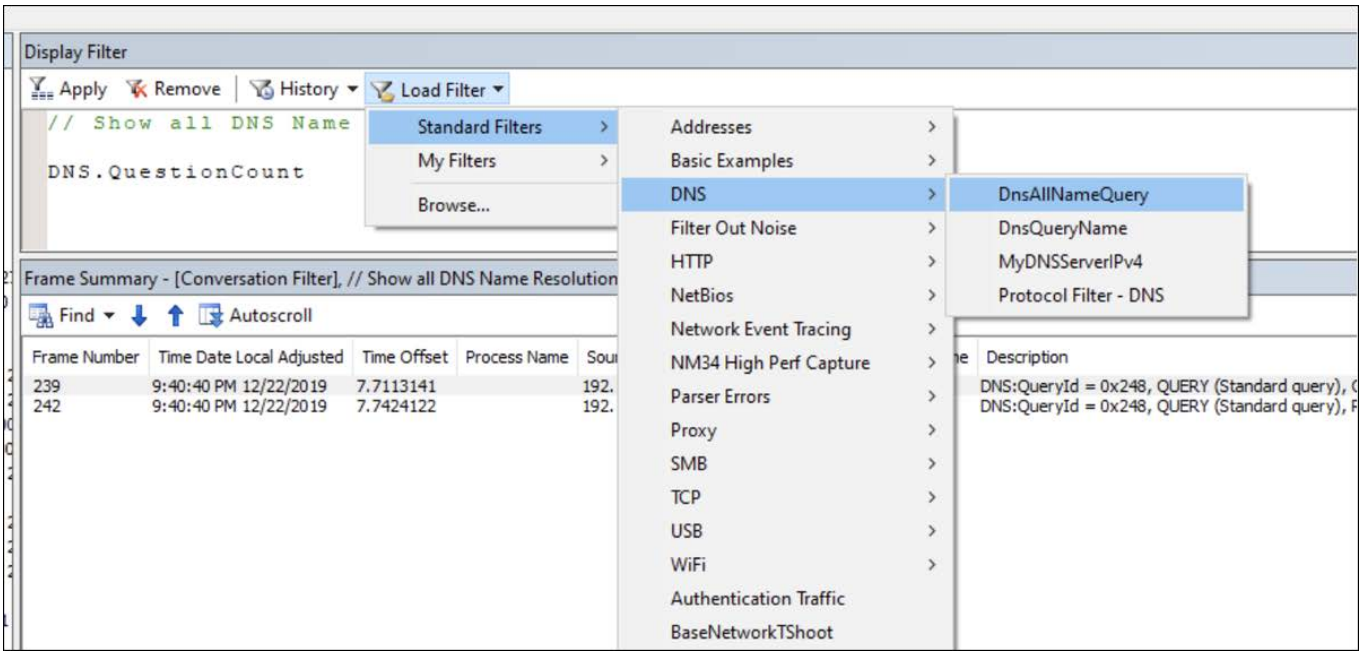
Microsoft

How to Capture and Inspect Network Packets in Windows Server

ADAM BERTRAM @adbertram

UPDATED AUG 19, 2020, 1:47 AM EDT | 4 min read



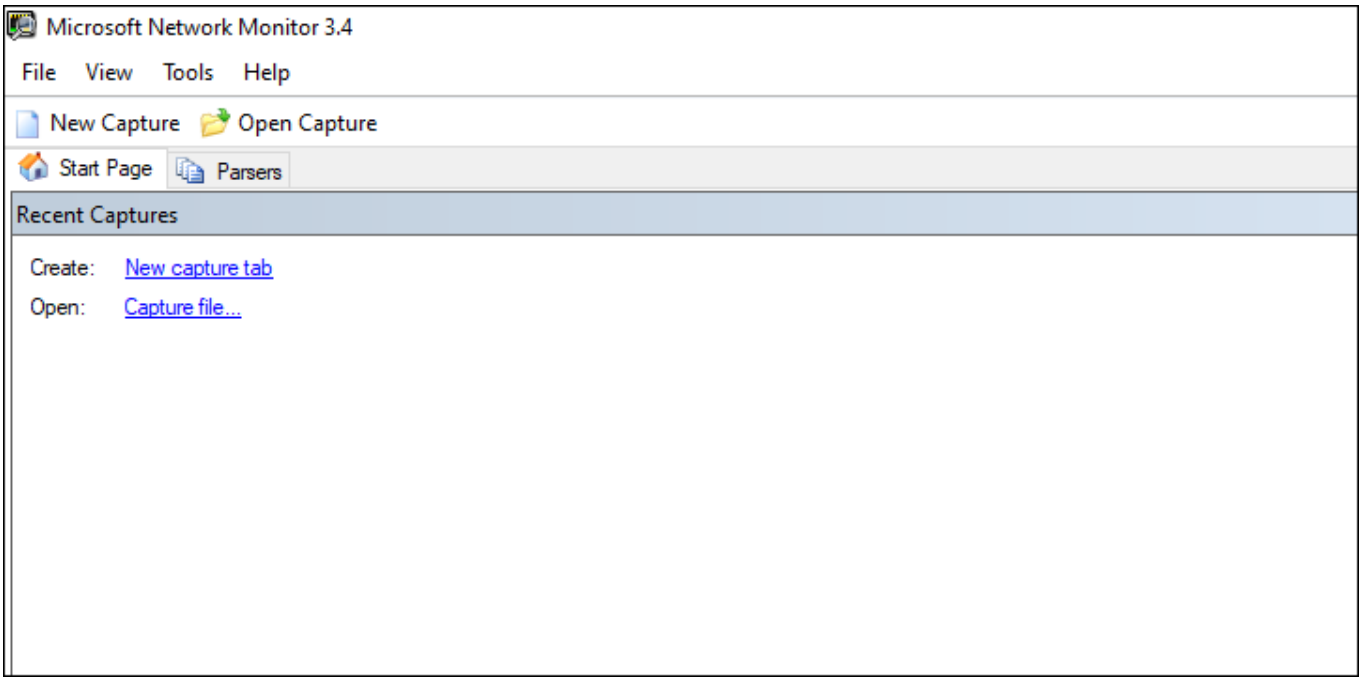
While troubleshooting tricky connection or application issues, it can be very helpful to see what is being transmitted across the network. Microsoft originally offered the Microsoft Network Monitor which was succeeded by the Microsoft Message Analyzer. Unfortunately, Microsoft has discontinued the Microsoft Message Analyzer and removed its download links. Currently, only the older Microsoft Network Monitor is available.

Of course, you can use third-party tools for performing network captures, such as WireShark. Though some third-party tools may offer a better experience Microsoft Network Monitor still holds its own. In this article, we are going to see how to capture and inspect packets using the last available version of Microsoft Network Monitor, one of the most popular tools out there.

Although I could have used WireShark, I have found that the interface and usability of Microsoft Network Monitor, out of the box, is far easier to use. Much of the same can be accomplished in WireShark, but you may have to do far more configuration in the interface.

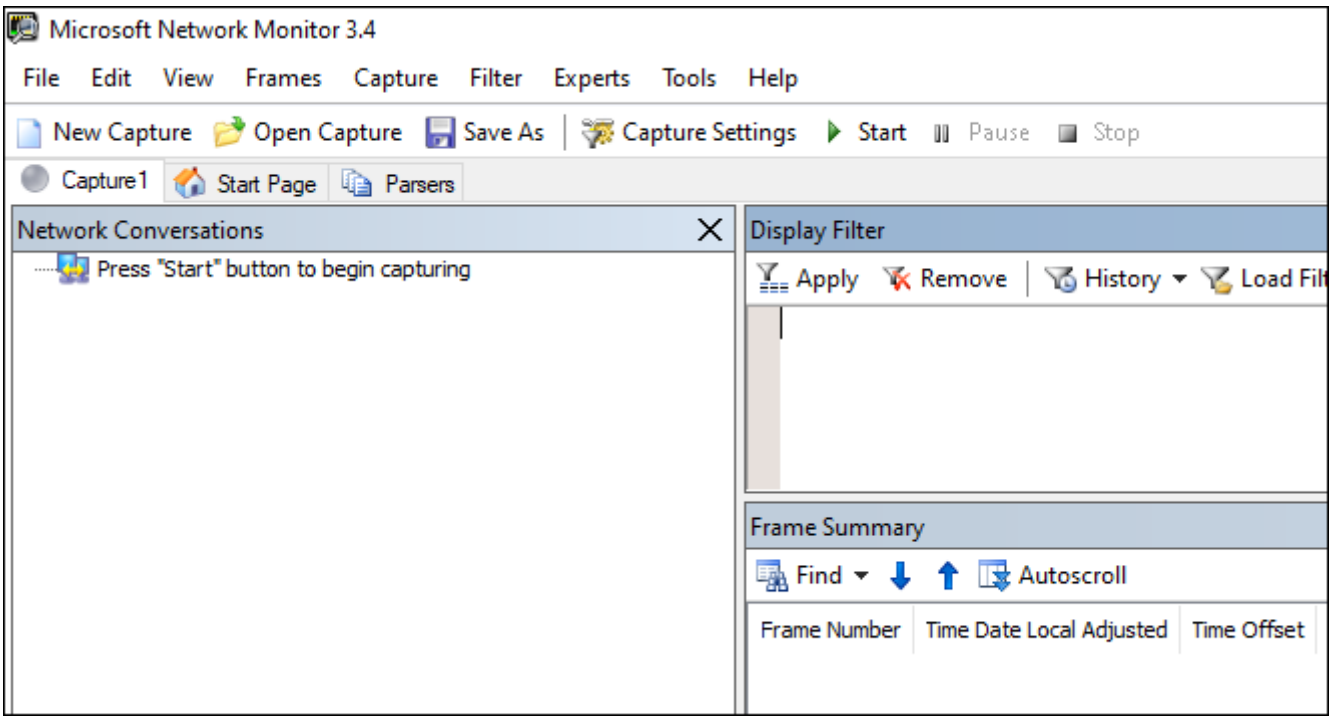
Capturing Packets Using Microsoft Network Monitor

First, we need to install Microsoft Network Monitor, you can locate the download here and then proceed to install it. Once you have Microsoft Network Monitor installed, go ahead and launch the program. Once launched, you will click on New Capture.



Viewing the Start Page

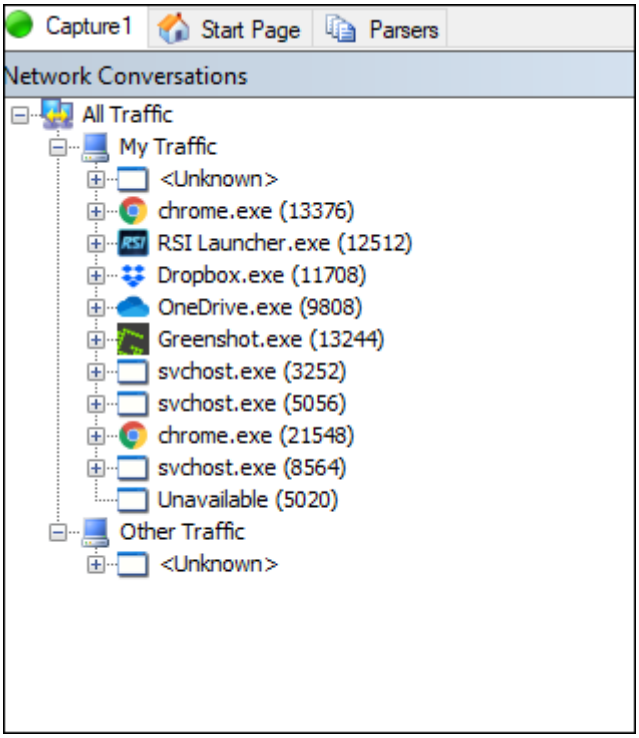
Next, you will want to start the monitoring by clicking on the Start button. This will instantly start the capture and you will see conversations starting to show up on the left-hand side.



Viewing a New Capture screen before it has started capturing

If you find that you get an error message saying no adapters are bound, then you should run Microsoft Network Monitor as an Administrator. Additionally, if you have just installed this, you may need to reboot.

One of the great benefits of using Microsoft Network Monitor is that it groups your network conversations very easily on the left-hand side. This makes looking at specific processes much easier to find and then dive into.

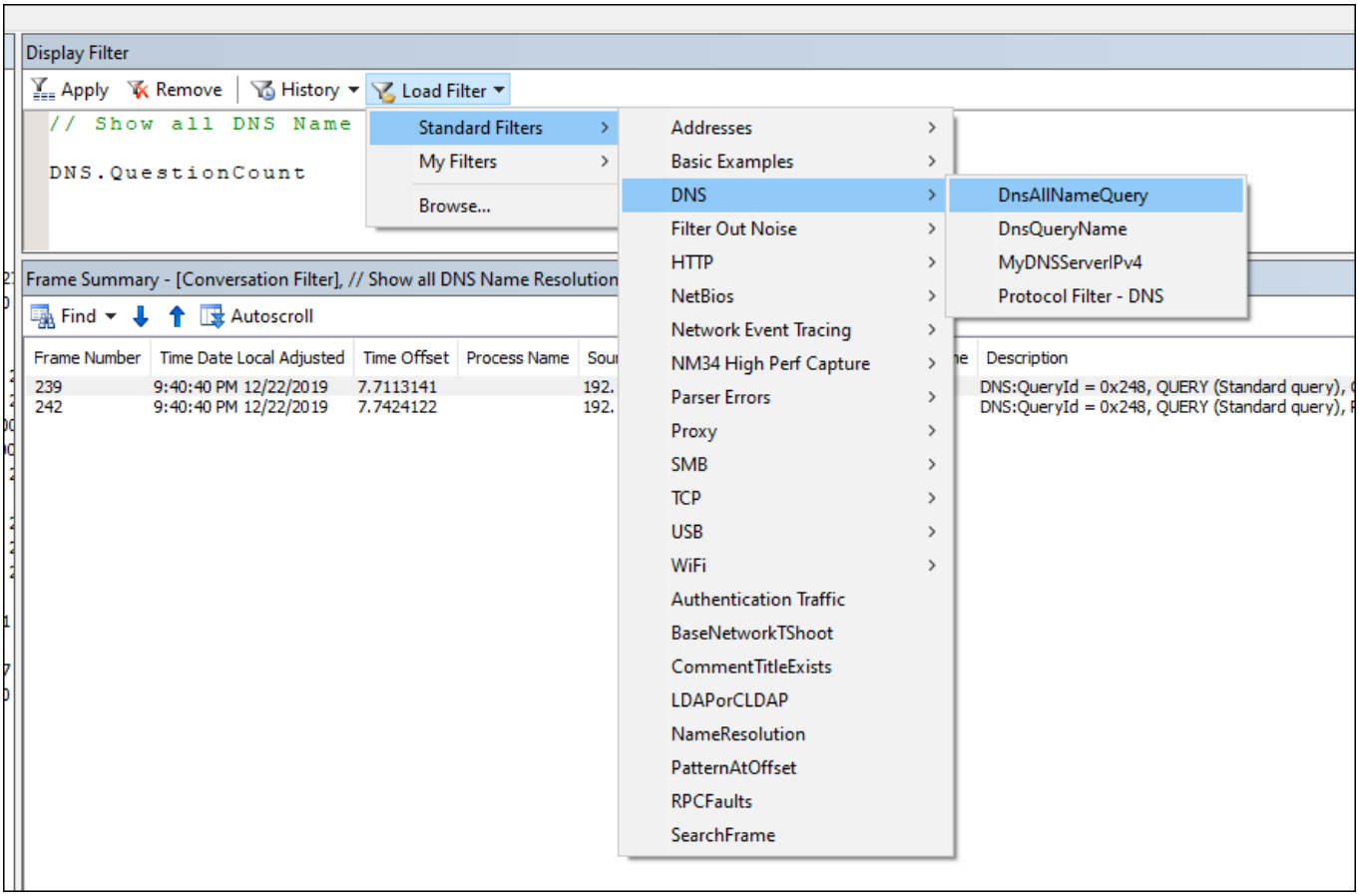


Viewing Network Conversations

Expanding any one of the plus signs will show you the specific set of “conversations” that the network monitor may have captured and grouped underneath a process.

Filtering Traffic

You will quickly find that with all of this data coming in, you will need to more easily filter out noise. One example of using a filter, is the `DnsAllNameQuery`, under the DNS section of Standard Filters. By adding this line to the display filter section and clicking on Apply, then you will be able to only display those packets that are DNS queries, such as below.



Viewing the DnsAllNameQuery Filter

Building Filters

Creating filters, or modifying the built-in filters, is very easy. Within the Display Filter field, there are several ways to construct filters. By entering in a Protocol Name and following that by a `.` (period), you will see an auto-complete of possible field values to compare. Using the standard comparison operator of `==` we can see if certain values are equal. We can even create multi-expressions using logic operators such as `and` and `or`. An example of what this looks like is below.

```
DNS.QuestionCount AND
DNS.ARecord.TimeToLive == 14
```

There are a few methods as well that are available such as `contains()` and `UINT8()`. You can see using the contains method below to filter out just DNS records that contain `[google.com](http://google.com)` and a TimeToLive of `14`.

```
DNS.QuestionCount AND
DNS.ARecord.TimeToLive == 14 AND
DNS.QRecord.QuestionName.contains("google.com")
```

As you might be able to tell, there are a number of ways to combine filters to make them useful and convenient to use. This is a great way to only return the data that you are interested in, especially since packet capture can become quite big. In the next section, we take a look at some more useful examples.

Example Filters

Some practical examples, beyond what the default built-in ones are, go a long way to helping you understand how to get to just the useful data that you need.

Filtering by Port Number

Though it's possible to use the HTTP protocol to filter by, using the following method allows you to account for custom ports, such as `8080` or `8443`, which is especially useful when troubleshooting.

```
// Filter by TCP Port Number
tcp.port == 80 OR Payloadheader.LowerProtocol.port == 80
tcp.port == 443 OR Payloadheader.LowerProtocol.port == 443
```

TCP frames that have been fragmented are reassembled and inserted into a new frame in the trace that contains a special header named, `Payloadheader`. By looking for both, we can make sure we are getting all of the data we are looking for here.

Find SSL Negotiation Frames

While troubleshooting, you may need to understand what SSL connections are attempted to be negotiated. Though you may not be able to decrypt the internal traffic, this will help find what servers the connection is attempting to use.

```
// Filter by SSL Handshake
TLS.TlsRecLayer.TlsRecordLayer.SSLHandshake.HandShake.HandShakeType == 0x1
```

Find TCP Retransmits and SYN Retransmits

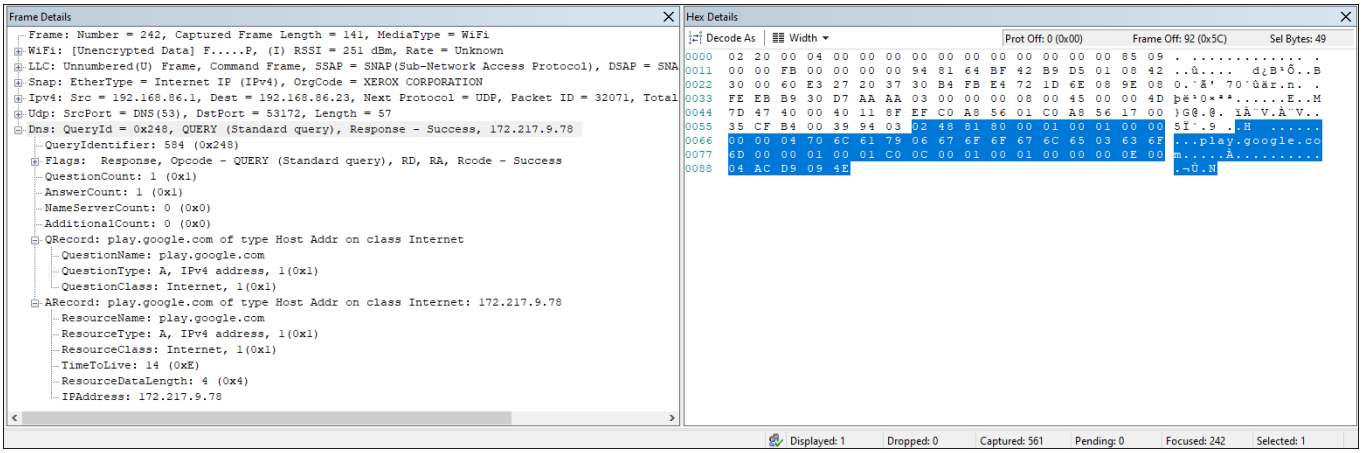
To troubleshoot file upload and download problems, you can look to see if many retransmissions are occurring that could be impacting performance.

```
Property.TCPRetransmit == 1 || Property.TCPSynRetransmit == 1
```


Make sure you have conversations turned on, this filter depends on that functionality.

Reading Frames and Hex Data

By default, the window layout has two bottom panes dedicated to Frame Details and Hex Details. Within the Frame Details is each packet broken up into its component parts. On the opposite side is the Hex Details which are the raw bytes and decoding. As you select a different section within the Frame details, the same section within the Hex code will be highlighted as well.



Viewing Frame Details and the raw Hex Data

Conclusion

Performing network traces is very easy with the latest version of Windows. Though Microsoft has opted to discontinue or deprecate their internally created tools, some still thrive. There are plenty of others, such as WireShark, but Microsoft Network Monitor still makes it quite easy to parse and understand the packet information that is captured.

ADAM BERTRAM

Adam Bertram is a 20+ year veteran of IT and an experienced online business professional. He's a consultant, Microsoft MVP, blogger, trainer, published author and content marketer for multiple

technology companies. Catch up on Adam's articles at , connect on , or follow him on Twitter at . **READ FULL BIO »**