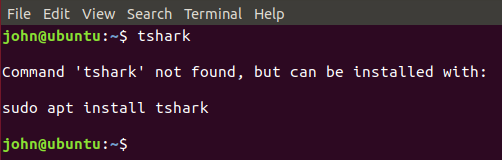
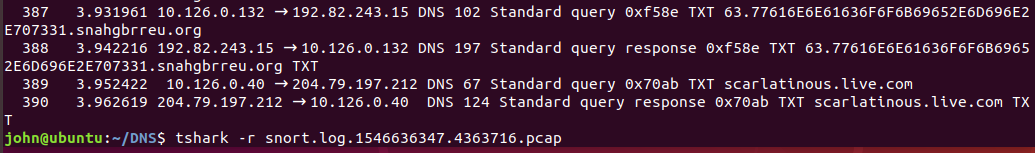
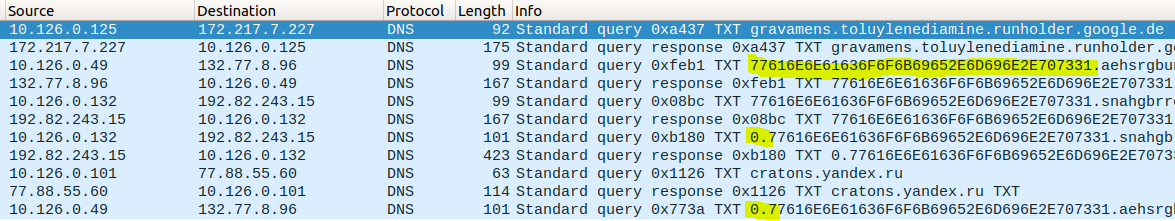
Terminal--Snort Challenge (Part 4)

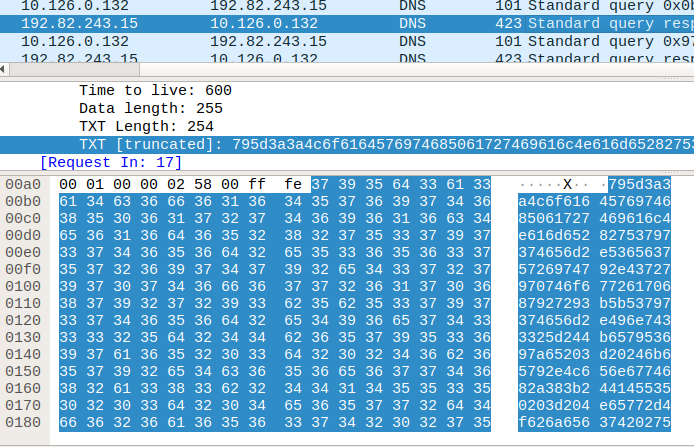
# A deeper look using tshark

The combination of Wireshark and tshark is very powerful for examining packet capture files. Wireshark can help you get the “lay of the land” and help find display filters and field names. Then tshark can extract fields to be analyzed in bulk. Of course, it helps to have tshark installed.  


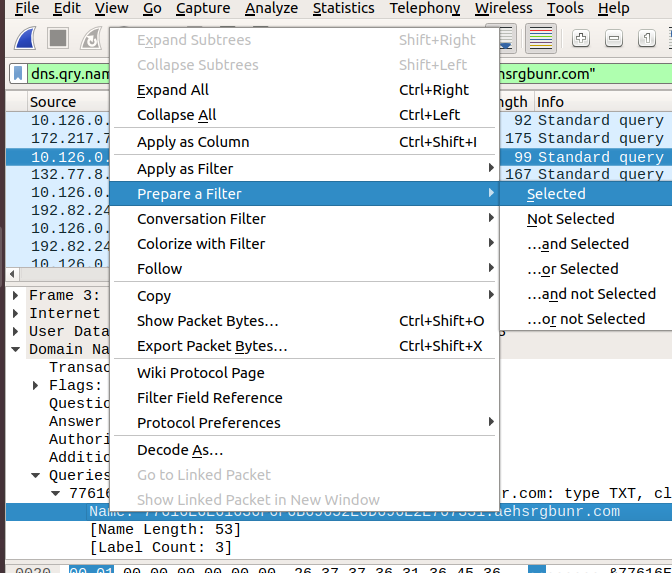
The commands in this lesson can generate a lot of output. To make it easier to display, I am taking screenshots at the end of the output and using the up arrow to show the command. In this case, tshark is just reading the capture file that Alabaster gave to us.  


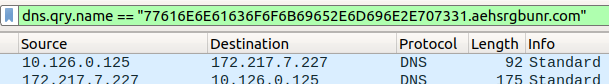
There are two items of interest in the packet capture file. The first is, does the hex in the query have meaning, and are there different hex strings in use?  


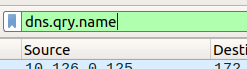
Also, the responses to the TXT queries all have long hex strings in the answer. What is going on there?

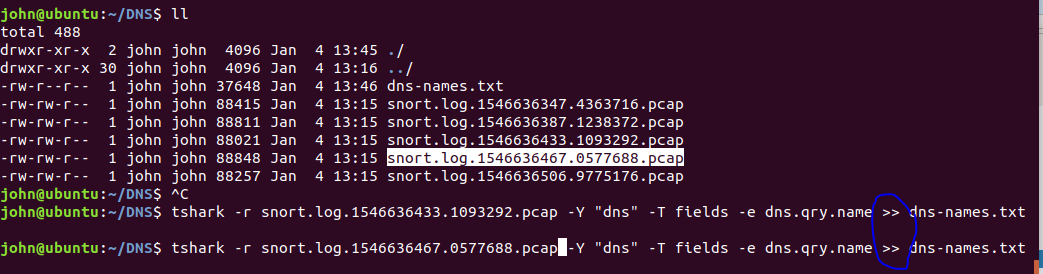


# Question 1. Does the hex have meaning, or different values?

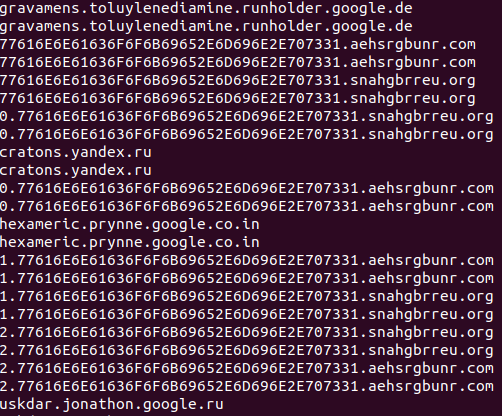
To answer the first question, we can find a field name and put that into tshark to dump all the DNS queries that were made. Right-click on the Name in the Wireshark data pane and select Prepare a Filter > Selected.  


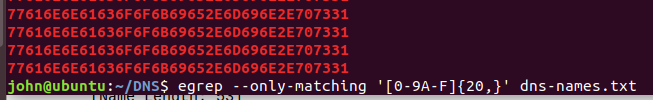
Wireshark creates a display filter that will find that field and packet.  


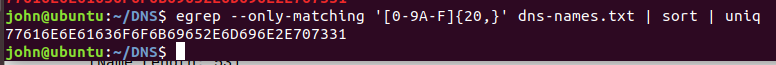
In our case, we just want to use the field name, so we can extract it from all packets.  


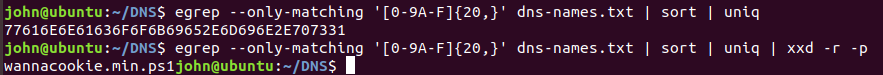
I added -Y "dns" to my tshark command, which gives a display filter for the DNS protocol. All the packets are DNS, but I just couldn’t stand not having some sort of filter. The important additions are   
-T fields and -e dns.qry.name. The -T just tells tshark we are going to extract fields. The  
 -e tells tshark what fields we want. There could be several fields, but we just need one.  


I downloaded several snort.log.xxxxxx.xxxx.pcap files from <http://snortsensor1.kringlecastle.com> (elf, onashelf) to get as much data a possible. Then I ran them through tshark to extract the DNS query names. Note that I’m using the >> so that I append to the file instead of overwriting it.

This is part of the file that was created by the tshark command.  


The same regular expression we used in the Snort rule will weed out the non-malware requests. The switch --only-matching causes egrep to only print the part of the line that matched the expression instead of the entire line.  


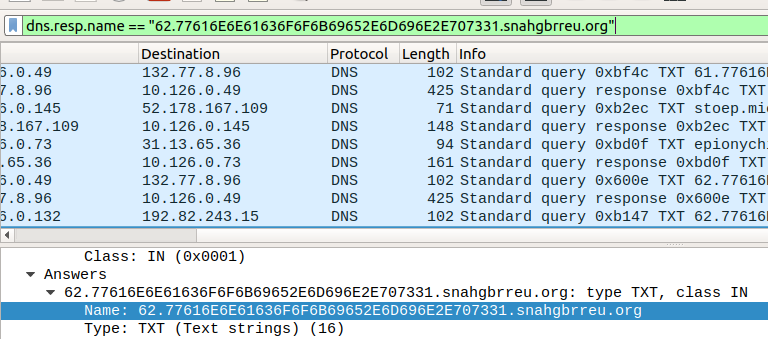
That is working, so we can pipe into sort and uniq to see how many different hex strings are present.  


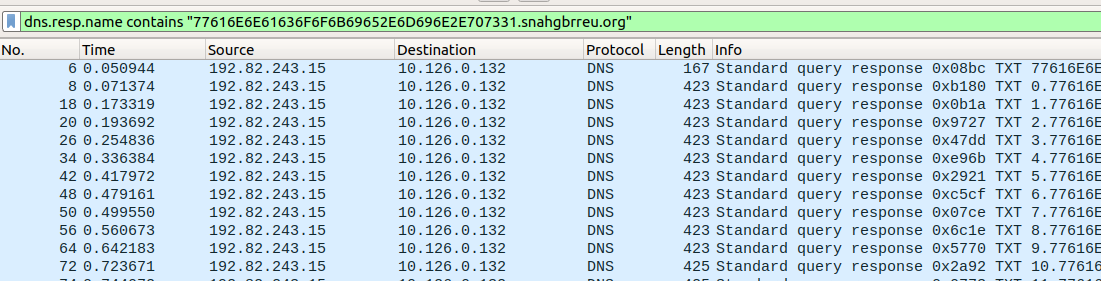
There is only one string! Our Snort rule could have been much simpler. Also, the hex string looks a lot like the numbers in the ASCII range. We can [pipe into xxd](https://stackoverflow.com/questions/13160309/conversion-hex-string-into-ascii-in-bash-command-line) to see if the numbers convert to ASCII.

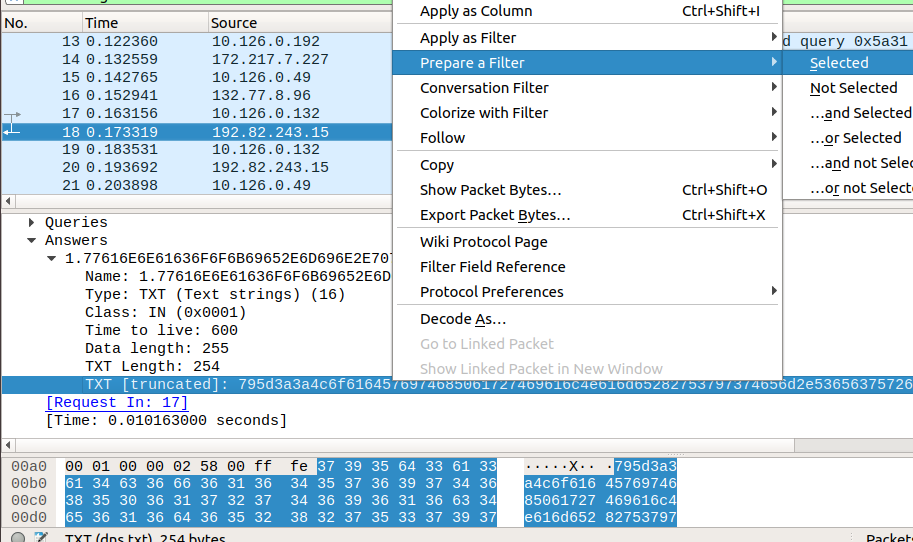
That’s interesting, wannacookie.min.ps1…it appears to be the name of a PowerShell file.

# Question 2. What is the text that is returned in the responses?

This time we do want a display filter for tshark so that we can show just one entire sequence. The right-click Prepare a Filter > Selected trick comes to our aid. Notice that we are making the selection in the Answers section of the packet, so we will omit queries; we only want the responses.

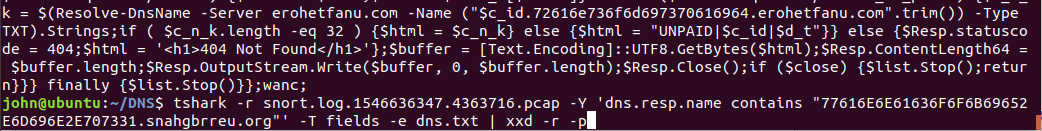


. However, if we use the filter as it is, we will only select one packet. Each request has two or three digits at the beginning of the request, most likely an identifier. We can fix that by removing the “62.” from the beginning of the filter, and by changing == to contains.  


We need the field name for the TXT data in the packet. Once again, Prepare a Filter comes to our aid. The field we want is dns.txt.  




Paste the values back into the tshark command. The display fiter to select just one exchange is  
-Y ‘dns.resp.name contains “77616E6E61636F6F6B69652E6D696E2E707331.snahgbrreu.org”’  
(If you use a different packet capture file, your filter will be different.)

Again, we are only extracting one field.  
-T fields -e dns.txt  


Since the hex digits were all in the ASCII range again, I piped into xxd -r -p to see what they meant. That is probably the malware code in PowerShell, so it would be wise to keep a copy of it.  


# Up next

Alabaster wants us to tell him where the malware is coming from. [Chris Davis’ talk](http://www.youtube.com/watch?v=wd12XRq2DNk) is essential for the challenge we will face.