# ECE 8813. Individual Assignment 1: Basic Network Traffic Analysis

## Prerequisites

- Familiarity with network protocols (e.g., the Open Systems Interconnection model).
- Familiarity with Python programming. Ability to perform rapid prototyping.

## Resources

- **Pyshark library:** Pyshark is a Python library that is a wrapper for the terminal version of Wireshark tshark. If you aren't familiar with packet processing you may also want to install and get familiar with the Wireshark network protocol analyzer to help you inspect traces manually. Pyshark requires `tshark` and `libxml`
- **Packet capture file** (traffic-analysis-exercise-1.pcap). The `sha1sum` is:

  `5f7b154880f5a30e3b54e2edd81ed0d928f6615c`

**WARNING: Do NOT visit any of the domains / URLs extracted from the trace unless you are in a sandbox. In fact, there is NO need to visit any of those sites to complete the assignment.**

## OBJECTIVE

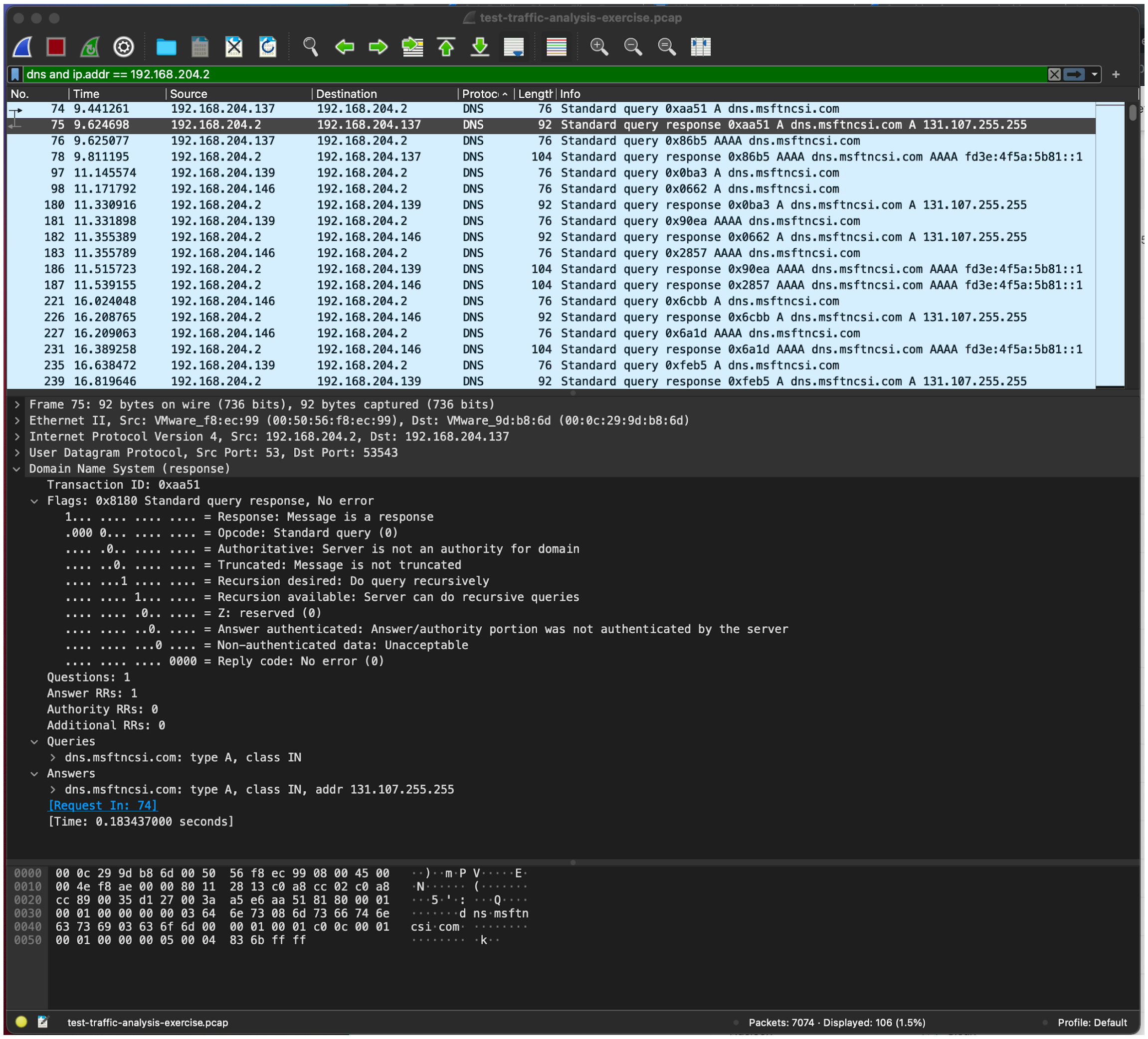
The objective is to get exposed to basics of network forensics via packet analysis. That experience will be helpful to properly assess some of the assigned readings in the course. The traffic capture file we will be examining is called `traffic-analysis-exercise-1.pcap`. You must work on this assignment on your own and be prepared to demonstrate your solution and discuss your findings in class. You have 1 week for this assignment. The specific tasks are:

1. Programmatically parse a packet capture (pcap) file.
2. Compute various traffic statistics for improved network situational awareness.
3. Perform a rudimentary forensic analysis on the collected network data.
4. Find indicators of compromise and explain how a breach may have happened.

When we discuss your solutions in class, a different pcap file (i.e, where some of your assumptions may not hold) will be provided for an in-class active learning task to see how well the learning objectives are met. So be sure to document your assumptions. More on that below.

## Part 1 - Gathering Basic Network Statistics (15 points)

The first step is to gather statistics about the network traffic and use that information to look for any anomalies or suspicious behavior. If you have not performed such analyses in prior courses before, you will want to install and get familiar with `Wireshark`. You can use `Wireshark` to, for example, search for DNS packets from a specific host.



test-traffic-analysis-exercise.pcap

dns and ip.addr == 192.168.204.2

| No. | Time | Source | Destination | Protoc | Length | Info |
|---|---|---|---|---|---|---|
| 74 | 9.441261 | 192.168.204.137 | 192.168.204.2 | DNS | 76 | Standard query 0xaa51 A dns.msftncsi.com |
| 75 | 9.624698 | 192.168.204.2 | 192.168.204.137 | DNS | 92 | Standard query response 0xaa51 A dns.msftncsi.com A 131.107.255.255 |
| 76 | 9.625077 | 192.168.204.137 | 192.168.204.2 | DNS | 76 | Standard query 0x86b5 AAAA dns.msftncsi.com |
| 78 | 9.811195 | 192.168.204.2 | 192.168.204.137 | DNS | 104 | Standard query response 0x86b5 AAAA dns.msftncsi.com AAAA fd3e:4f5a:5b81::1 |
| 97 | 11.145574 | 192.168.204.139 | 192.168.204.2 | DNS | 76 | Standard query 0x0ba3 A dns.msftncsi.com |
| 98 | 11.171792 | 192.168.204.146 | 192.168.204.2 | DNS | 76 | Standard query 0x0662 A dns.msftncsi.com |
| 180 | 11.330916 | 192.168.204.2 | 192.168.204.139 | DNS | 92 | Standard query response 0x0ba3 A dns.msftncsi.com A 131.107.255.255 |
| 181 | 11.331898 | 192.168.204.139 | 192.168.204.2 | DNS | 76 | Standard query 0x90ea AAAA dns.msftncsi.com |
| 182 | 11.355389 | 192.168.204.2 | 192.168.204.146 | DNS | 92 | Standard query response 0x0662 A dns.msftncsi.com A 131.107.255.255 |
| 183 | 11.355789 | 192.168.204.146 | 192.168.204.2 | DNS | 76 | Standard query 0x2857 AAAA dns.msftncsi.com |
| 186 | 11.515723 | 192.168.204.2 | 192.168.204.139 | DNS | 104 | Standard query response 0x90ea AAAA dns.msftncsi.com AAAA fd3e:4f5a:5b81::1 |
| 187 | 11.539155 | 192.168.204.2 | 192.168.204.146 | DNS | 104 | Standard query response 0x2857 AAAA dns.msftncsi.com AAAA fd3e:4f5a:5b81::1 |
| 221 | 16.024048 | 192.168.204.146 | 192.168.204.2 | DNS | 76 | Standard query 0x6cbb A dns.msftncsi.com |
| 226 | 16.208765 | 192.168.204.2 | 192.168.204.146 | DNS | 92 | Standard query response 0x6cbb A dns.msftncsi.com A 131.107.255.255 |
| 227 | 16.209063 | 192.168.204.146 | 192.168.204.2 | DNS | 76 | Standard query 0x6a1d AAAA dns.msftncsi.com |
| 231 | 16.389258 | 192.168.204.2 | 192.168.204.146 | DNS | 104 | Standard query response 0x6a1d AAAA dns.msftncsi.com AAAA fd3e:4f5a:5b81::1 |
| 235 | 16.638472 | 192.168.204.139 | 192.168.204.2 | DNS | 76 | Standard query 0xfeb5 A dns.msftncsi.com |
| 239 | 16.819646 | 192.168.204.2 | 192.168.204.139 | DNS | 92 | Standard query response 0xfeb5 A dns.msftncsi.com A 131.107.255.255 |

```
> Frame 75: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
> Ethernet II, Src: VMware_f8:ec:99 (00:50:56:f8:ec:99), Dst: VMware_9d:b8:6d (00:0c:29:9d:b8:6d)
> Internet Protocol Version 4, Src: 192.168.204.2, Dst: 192.168.204.137
> User Datagram Protocol, Src Port: 53, Dst Port: 53543
v Domain Name System (response)
    Transaction ID: 0xaa51
  v Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > dns.msftncsi.com: type A, class IN
  v Answers
    > dns.msftncsi.com: type A, class IN, addr 131.107.255.255
    [Request In: 74]
    [Time: 0.183437000 seconds]
```

```
0000  00 0c 29 9d b8 6d 00 50  56 f8 ec 99 08 00 45 00   ··)··m·P V·····E·
0010  00 4e f8 ae 00 00 80 11  28 13 c0 a8 cc 02 c0 a8   ·N······ (·······
0020  cc 89 00 35 d1 27 00 3a  a5 e6 aa 51 81 80 00 01   ···5·'·: ···Q····
0030  00 01 00 00 00 00 03 64  6e 73 08 6d 73 66 74 6e   ·······d ns·msftn
0040  63 73 69 03 63 6f 6d 00  00 01 00 01 c0 0c 00 01   csi·com· ········
0050  00 01 00 00 00 05 00 04  83 6b ff ff               ········ ·k··
```

test-traffic-analysis-exercise.pcap

Packets: 7074 · Displayed: 106 (1.5%)

Profile: Default

Because traffic traces can be large, you must automate the process. Thus, write a `python3` program that leverages the pyshark module to provide the information below. Specify the pcap file with a `--file` argument (e.g., `python3 analyzer.py --file traffic.pcap`). Other arguments should be specified as below:

1. [`-n <int> --dests`] Provide a ranked list of the top *n* destination addresses based on the number of packets received. E.g, your script might produce output similar to that below for the top *n* =10 destination addresses.

```
============================
   Top 10 Source Addresses
============================
IP Address         Rx Packets
192.168.204.137    1482
192.168.204.139    1394
192.168.204.146    1289
213.186.33.19       620
50.57.227.160       490
213.146.191.132     279
74.125.232.26       214
74.125.232.25       126
98.138.250.88       103
192.168.204.2        92
--------------------------
```

2. [`-n <int> --sources`] Provide a ranked list of the top *n* source addresses based on the number of packets transferred by each address. For example:

```
============================
   Top 10 Source Addresses
============================
IP Address         Tx Packets
213.186.33.19      1260
192.168.204.146    1207
50.57.227.160       885
192.168.204.139     868
192.168.204.137     815
213.146.191.132     307
74.125.232.26       257
98.138.250.88       220
74.125.232.25       134
168.235.69.248      128
--------------------------
```

3. [`--protocols`] Provide a ranked list of the network protocols at each level of the OSI model based on their frequency in the traffic, e.g.:

```
=========================
   Ranked Protocol List
==========================
Layer         Protocol   Rank
Application   HTTP       1
              DNS        2

Transport     TCP        1

Network       IP         1
--------------------------
```

4. [--contx] Output an estimate of the number of **successful** TCP connections observed in the trace specified by the --file flag. Be sure to describe the heuristic you used to determine if a connection was successful or not.

## Part 2 - Who's Talking to Whom (25 points)

Now that you have a basic understanding about the communication patterns on the network, your task is to identify some key information that will be needed to answer questions in Part 3.

- [--talkers] **Return the Top Talkers**
    - Here, top talkers are defined as the source and destination pairs that are responsible for the majority of communication. The designation of a top talker in this scenario can be based on the number of packets exchanged between a source and a destination IP address. To identify the top talkers, you can use the information gathered about the top source and destination addresses from part 1. Sort your results by the number of connections between hosts.

- [--evasive] **Return the list of suspicious connections**. For each protocol in the trace, list any traffic for that protocol that is on a non-standard port. The well-known ports can be found here. For now, you may limit yourself to the most frequent protocols observed in the trace. As an example, you might flag HTTP/HTTPS traffic that is not on the standard port 80/443:

```
=================================================
            HTTP Traffic Statistics
=================================================
==> Source IP Address: 192.168.206.2
==> Destination IP Address: 168.235.69.34
==> URL http://getpayload.eu:12345/file.js
==> Host getpayload.eu:12345
==> On Port: 12345
==> Browser Mozilla
==> Content: application/javascript
```

- [`--windows`] Provide the hostname, IP address, MAC address, operating system and workgroup of the Windows hosts in the trace. For example:

```
==========================
       Windows Hosts
==========================
Host Name:      XXXXXXXXX
Mac Address:    00:0c:22:8d:65:98
Work Group:     XXXXXXXXX
IP Address:     192.168.0.1
OS:             Windows XP
#####
Host Name:      YYYYYYYYY
Mac Address:    00:0c:fd:84:65:2b
Work Group:     YYYYYYYYY
IP Address:     192.168.0.2
OS:             Windows XP
#####
Host Name:      ZZZZZZZZZ
Mac Address:    00:0a:22:84:65:bc
Work Group:     ZZZZZZZZZ
IP Address:     192.168.0.3
OS:             Windows 7
#####
```

## Part 3: Forensic Report (60 points)

Great, we can now turn to answering some specific questions and document your answers. The only context we have about the packet capture is that the analyst that collected it suspected that one of the websites in the trace was actually compromised by some type of “drive-by-download” (i.e., wherein the vulnerable machine(s) interacting with that site are in turn infected by an exploit kit). To test that conjecture, your task is to determine if any of the Windows hosts were infected, and if so, to provide the indicators of compromise. Your report must specify:

1. [10 points] The machine(s) you believe were infected.

2. [20 points] A list of domain name(s) and IP address(es) of the compromised website(s) that the infected machine interacted with.

3. [15 points] An explanation of the sequence of events that led to the machine(s) being infected. Justify your answer.

4. [15 points] A proposal for how you would rate the "riskiness" of each website found in a network trace. If there are specific characteristics you would use in your rating algorithm, describe them. Similarly, if during your explorations of how you would design such an algorithm, you discovered APIs that you could use when designing such an algorithm, please document them (as those APIs might come in handy for a future assignment 😁).

In answering the above questions, you must document the specific assumptions you made. **Come prepared to demonstrate your solution and present your findings in class.** We are also interested in any techniques you developed or tools you used to visualize your findings (e.g., for answering question 3).

## Part 4 - Extra Credit

- What type of exploit was used by the exploit kit(s) observed in the pcap? Justify your answer with evidence from the pcap.

- What URL(s) acted as a redirect between the compromised website(s) and the exploit kit(s)?

## Use of External Resources

You MAY use online materials to assist you in your coding endeavors. However, any code you use must be properly cited with a link to where you found it. Some resources that you may find useful for this exercise:

- An introduction to pyshark
- Wireshark basics

## How To Submit Your Solution

You have 1 week to complete this assignment. Your solution is due **TBD**. For this assignment you are required to submit a PDF with the answers to the questions in Part 3. The PDF should also include a link to a compressed archive of your code. The archive must include a readme file that explains how to run your code (e.g., any custom flags). Happy Hunting!