Data extraction using Scapy and Pandas

Scapy

Scapy is an interactive packet manipulation library written in Python able to decode packets from many protocols.

- Directly reads PCAP files
- Extracts all relevant flow-level and packet-level features (
- Supports filtering by protocol or activity type, enabling precise analysis.
- Can be combined with ML pipelines by providing structured feature data.

Pandas

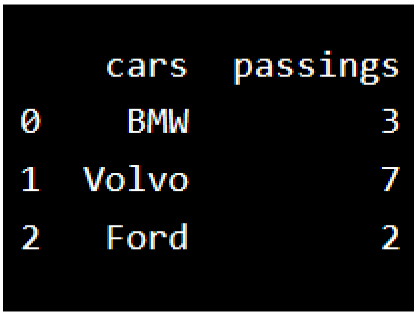
Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

- Allows for importing and exporting tabular data in various formats
- Aids in data manipulation operations
- Aids in data cleaning
- Expertly handles missing data
- Supports data visualization
- Flexible reshaping
- Fully documented

Pandas example:

The code below creates and outputs a simple dataset that shows how data can be stored and visualised using Pandas.

```
mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pandas.DataFrame(mydataset)

print(myvar)
```



```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

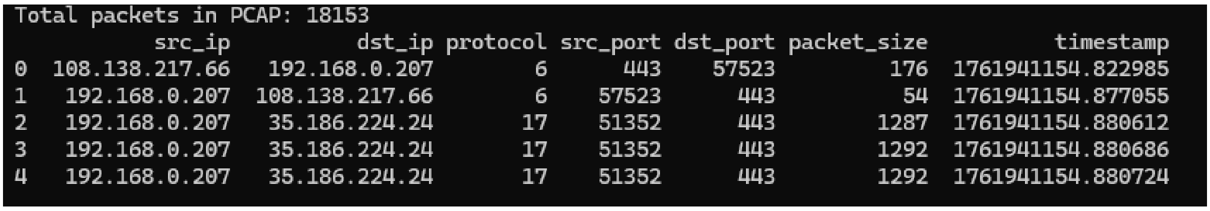
Scapy example

The following code reads a pcap file and outputs each individual packet's source IP, destination IP and length. It shows how incredibly easy Scapy makes PCAP file analysis.

Code	Output
from scapy.all import rdpcap	208.21.2.184 10.1.1.99 1659
	208.21.2.184 10.1.1.99 4232
packets = rdpcap("example.pcap")	208.21.2.184 10.1.1.99 2997
for pkt in packets:	208.21.2.184 10.1.1.99 1685
if pkt.haslayer("IP"):	208.21.2.184 10.1.1.99 4487
print(pkt["IP"].src, pkt["IP"].dst, len(pkt))	208.21.2.184 10.1.1.99 4513

Using Scapy and Pandas together

Using the example Scapy and Pandas code as our foundation, we can abstract further details from the file and output them in a readable way.

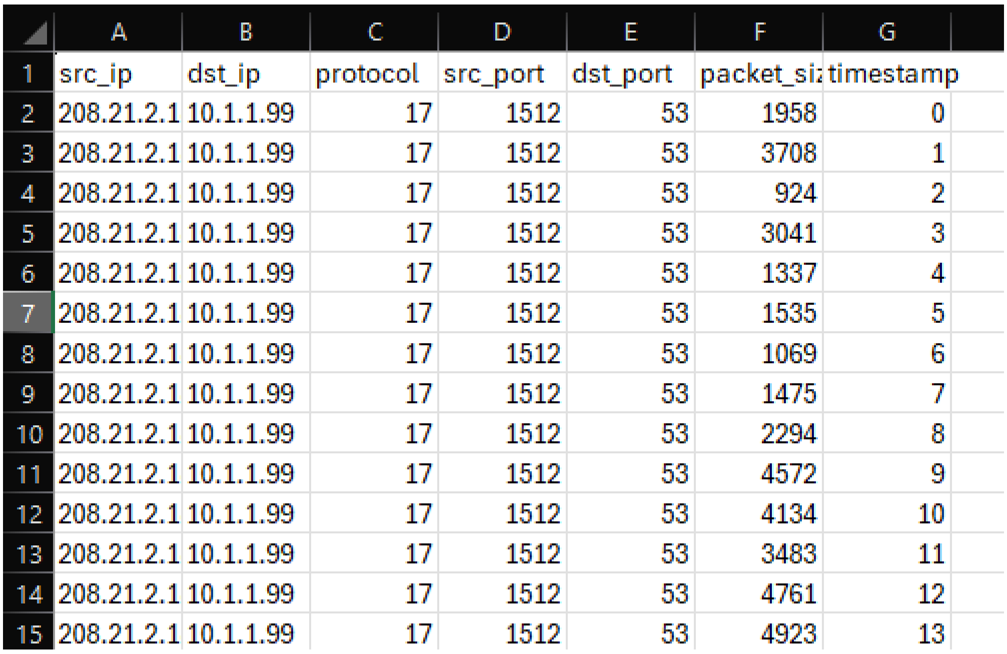


```
Total packets in PCAP: 18153
           src_ip          dst_ip protocol src_port dst_port packet_size          timestamp
0  108.138.217.66   192.168.0.207        6      443    57523         176  1761941154.822985
1   192.168.0.207  108.138.217.66        6    57523      443          54  1761941154.877055
2   192.168.0.207   35.186.224.24       17    51352      443        1287  1761941154.880612
3   192.168.0.207   35.186.224.24       17    51352      443        1292  1761941154.880686
4   192.168.0.207   35.186.224.24       17    51352      443        1292  1761941154.880724
```

```
# Load the PCAP file
packets = rdpcap("test2.pcapng")

print(f"Total packets in PCAP: {len(packets)}")

# Create empty DataFrame with desired columns
columns = ["src_ip", "dst_ip", "protocol", "src_port", "dst_port",
        "packet_size", "timestamp"]
df = pd.DataFrame(columns=columns)

# Iterate packets and append rows
for pkt in packets:
   if "IP" in pkt:
      src = pkt["IP"].src
      dst = pkt["IP"].dst
      proto = pkt["IP"].proto
      size = len(pkt)
      ts = pkt.time

      # TCP/UDP ports if available
      src_port = pkt["TCP"].sport if pkt.haslayer("TCP") else (pkt["UDP"].sport if pkt.haslayer("UDP") else None)
      dst_port = pkt["TCP"].dport if pkt.haslayer("TCP") else (pkt["UDP"].dport if pkt.haslayer("UDP") else None)

      # Append row directly
      df = pd.concat([df, pd.DataFrame([[src, dst, proto, src_port, dst_port, size, ts]], columns=columns)],
ignore_index=True)

print(df.head())
```

Data output

Pandas makes it very easy to output data to files. Adding the following line to the code above ‘df.to_csv("network_flows.csv", index=False)’ will yield A CSV file:



	A	B	C	D	E	F	G
1	src_ip	dst_ip	protocol	src_port	dst_port	packet_siz	timestamp
2	208.21.2.1	10.1.1.99	17	1512	53	1958	0
3	208.21.2.1	10.1.1.99	17	1512	53	3708	1
4	208.21.2.1	10.1.1.99	17	1512	53	924	2
5	208.21.2.1	10.1.1.99	17	1512	53	3041	3
6	208.21.2.1	10.1.1.99	17	1512	53	1337	4
7	208.21.2.1	10.1.1.99	17	1512	53	1535	5
8	208.21.2.1	10.1.1.99	17	1512	53	1069	6
9	208.21.2.1	10.1.1.99	17	1512	53	1475	7
10	208.21.2.1	10.1.1.99	17	1512	53	2294	8
11	208.21.2.1	10.1.1.99	17	1512	53	4572	9
12	208.21.2.1	10.1.1.99	17	1512	53	4134	10
13	208.21.2.1	10.1.1.99	17	1512	53	3483	11
14	208.21.2.1	10.1.1.99	17	1512	53	4761	12
15	208.21.2.1	10.1.1.99	17	1512	53	4923	13

References:

https://pandas.pydata.org/

https://www.nvidia.com/en-gb/glossary/pandas-python/

https://www.w3schools.com/python/pandas/pandas_getting_started.asp