Tutorial: Converting CSV Datasets (e.g., NSL-KDD) into PCAPs for Wireshark

◇ Prerequisites

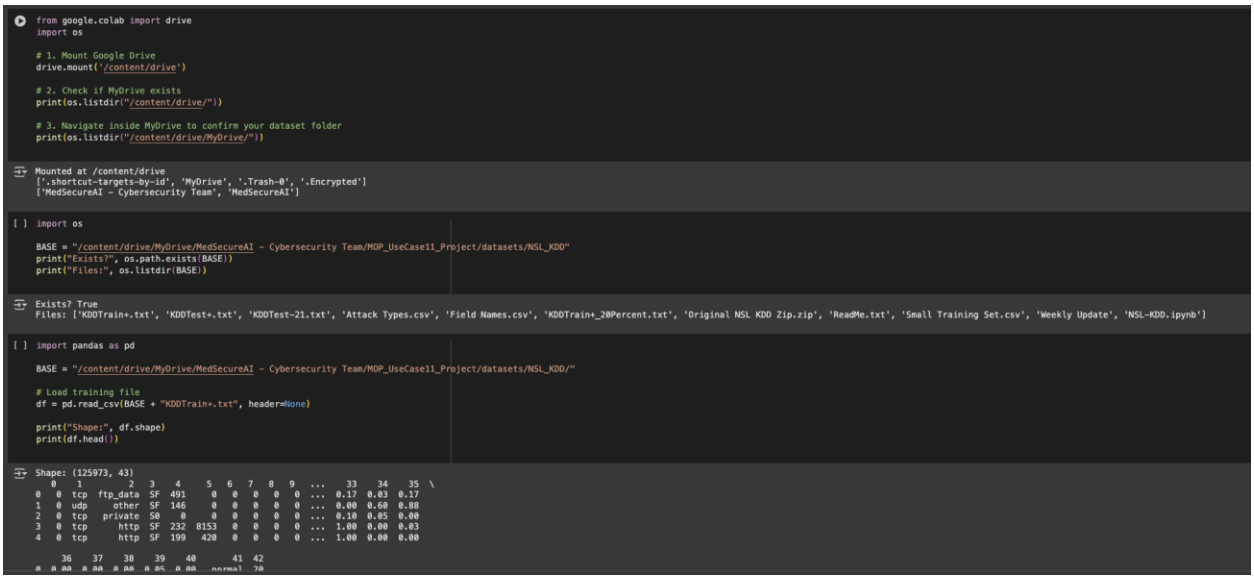
- **Google Colab** or Jupyter Notebook
- Python 3.x
- Libraries: pandas, scapy

pip install pandas scapy

◇ Step 1: Mount Google Drive (if dataset is stored in Drive)

```
from google.colab import drive
drive.mount('/content/drive')
```

Confirm your dataset path, e.g.:
/content/drive/MyDrive/MedSecureAI - Cybersecurity Team/MOP_UseCase11_Project/datasets/NSL_KDD/



```
from google.colab import drive
import os

# 1. Mount Google Drive
drive.mount('/content/drive')

# 2. Check if MyDrive exists
print(os.listdir("/content/drive/"))

# 3. Navigate inside MyDrive to confirm your dataset folder
print(os.listdir("/content/drive/MyDrive/"))
```

```
Mounted at /content/drive
['.shortcut-targets-by-id', 'MyDrive', '.Trash-0', '.Encrypted']
['MedSecureAI - Cybersecurity Team', 'MedSecureAI']
```

```
import os

BASE = "/content/drive/MyDrive/MedSecureAI - Cybersecurity Team/MOP_UseCase11_Project/datasets/NSL_KDD"
print("Exists?", os.path.exists(BASE))
print("Files:", os.listdir(BASE))
```

```
Exists? True
Files: ['KDDTrain+.txt', 'KDDTest+.txt', 'KDDTest-21.txt', 'Attack Types.csv', 'Field Names.csv', 'KDDTrain+_20Percent.txt', 'Original NSL KDD Zip.zip', 'ReadMe.txt', 'Small Training Set.csv', 'Weekly Update', 'NSL-KDD.ipynb']
```

```
import pandas as pd

BASE = "/content/drive/MyDrive/MedSecureAI - Cybersecurity Team/MOP_UseCase11_Project/datasets/NSL_KDD/"

# Load training file
df = pd.read_csv(BASE + "KDDTrain+.txt", header=None)

print("Shape:", df.shape)
print(df.head())
```

```
Shape: (125973, 43)
   0   1        2   3    4     5   6   7   8   9  ...    33    34    35  \
0  0  tcp  ftp_data  SF  491     0   0   0   0   0  ...  0.17  0.03  0.17
1  0  udp     other  SF  146     0   0   0   0   0  ...  0.00  0.60  0.88
2  0  tcp   private  S0    0     0   0   0   0   0  ...  0.10  0.05  0.00
3  0  tcp      http  SF  232  8153   0   0   0   0  ...  1.00  0.00  0.03
4  0  tcp      http  SF  199   420   0   0   0   0  ...  1.00  0.00  0.00

     36    37    38    39    40          41  42
0  0.00  0.00  0.00  0.00  0.05  0.00  normal  20
```

◇ Step 2: Load the CSV Dataset

import pandas as pd
import os

# Path to dataset
BASE = "/content/drive/MyDrive/MedSecureAI - Cybersecurity Team/MOP_UseCase11_Project/datasets/NSL_KDD"
df = pd.read_csv(os.path.join(BASE, "KDDTrain+.txt"), header=None)

print(df.shape)

print(df.head())

```
[ ]  import pandas as pd

     BASE = "/content/drive/MyDrive/MedSecureAI - Cybersecurity Team/MOP_UseCase11_Project/datasets/NSL_KDD/"

     # Load training file
     df = pd.read_csv(BASE + "KDDTrain+.txt", header=None)

     print("Shape:", df.shape)
     print(df.head())
```

```
Shape: (125973, 43)
   0    1         2   3    4     5   6   7   8   9   ...    33    34    35  \
0   0  tcp  ftp_data  SF  491     0   0   0   0   0  ...  0.17  0.03  0.17
1   0  udp     other  SF  146     0   0   0   0   0  ...  0.00  0.60  0.88
2   0  tcp   private  S0    0     0   0   0   0   0  ...  0.10  0.05  0.00
3   0  tcp      http  SF  232  8153   0   0   0   0  ...  1.00  0.00  0.03
4   0  tcp      http  SF  199   420   0   0   0   0  ...  1.00  0.00  0.00

     36    37    38    39    40       41  42
0  0.00  0.00  0.00  0.05  0.00   normal  20
1  0.00  0.00  0.00  0.00  0.00   normal  15
2  0.00  1.00  1.00  0.00  0.00  neptune  19
3  0.04  0.03  0.01  0.00  0.01   normal  21
4  0.00  0.00  0.00  0.00  0.00   normal  21

[5 rows x 43 columns]
```

◇ Step 3: Add Column Names

NSL-KDD has **43 features**. Example:

columns = [
"duration","protocol_type","service","flag","src_bytes","dst_bytes","land",
"wrong_fragment","urgent","hot","num_failed_logins","logged_in","num_compromised",
"root_shell","su_attempted","num_root","num_file_creations","num_shells",
"num_access_files","num_outbound_cmds","is_host_login","is_guest_login",
"count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate",

"same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count
",
"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","label","difficulty_level"
]

df.columns = columns

```
columns = [
    "duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment",
    "urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted",
    "num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login",
    "is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate",
    "same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate",
    "dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate",
    "label", "difficulty_level"
]
df.columns = columns
```

```
[ ] # Select only useful columns
    df_small = df[["protocol_type", "service", "src_bytes", "dst_bytes", "label"]]

    print(df_small.head())
    print(df_small.shape)
```

```
  protocol_type   service  src_bytes  dst_bytes    label
0           tcp  ftp_data        491          0   normal
1           udp     other        146          0   normal
2           tcp   private          0          0  neptune
3           tcp      http        232       8153   normal
4           tcp      http        199        420   normal
(125973, 5)
```

◇ Step 4: Extract Only Relevant Fields

For packet simulation we only need:

- Protocol (protocol_type)
- Source Bytes (src_bytes)
- Destination Bytes (dst_bytes)
- Label (label)

df_small = df[["protocol_type","src_bytes","dst_bytes","label"]]
print(df_small.head())

```
columns = [
    "duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment",
    "urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted",
    "num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login",
    "is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate",
    "same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate",
    "dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate",
    "label", "difficulty_level"
]
df.columns = columns
```

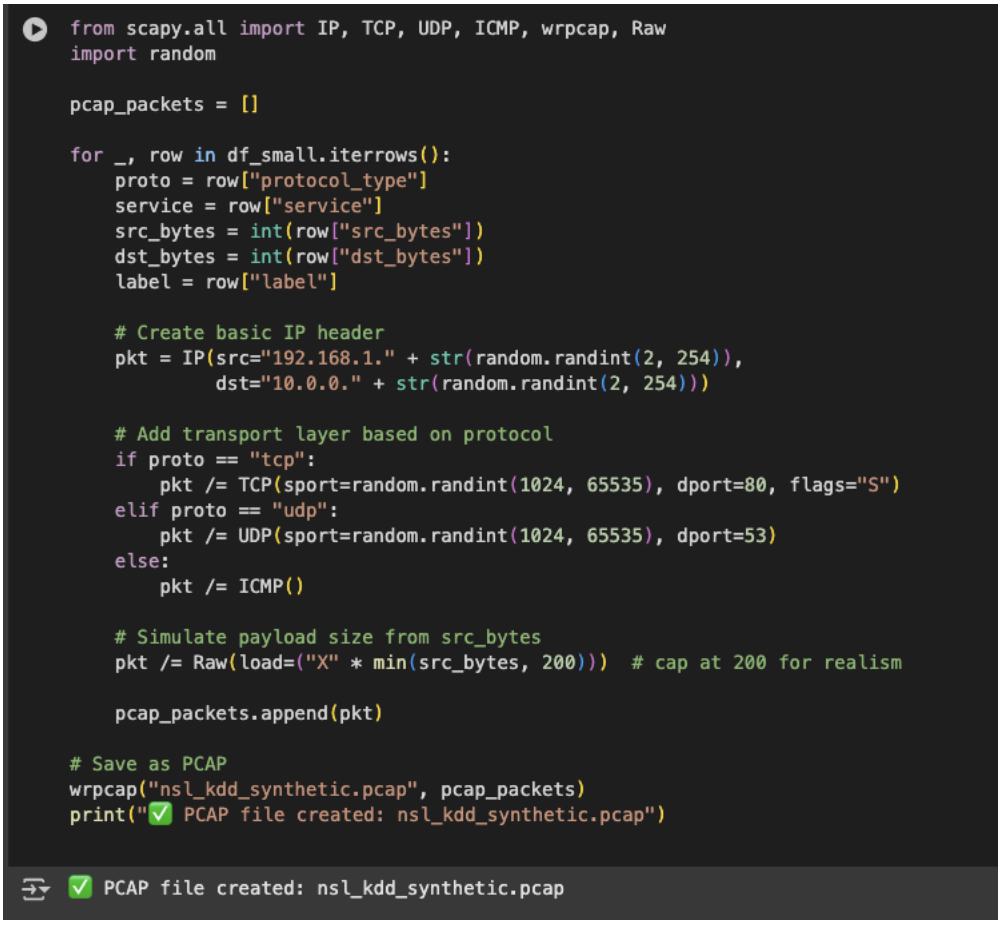
```
[ ] # Select only useful columns
    df_small = df[["protocol_type", "service", "src_bytes", "dst_bytes", "label"]]

    print(df_small.head())
    print(df_small.shape)
```

```
  protocol_type   service  src_bytes  dst_bytes    label
0           tcp  ftp_data        491          0   normal
1           udp     other        146          0   normal
2           tcp   private          0          0  neptune
3           tcp      http        232       8153   normal
4           tcp      http        199        420   normal
(125973, 5)
```

◇ Step 5: Generate Synthetic Packets with Scapy

```
from scapy.all import IP, TCP, UDP, ICMP, wrpcap, Raw
import random

pcap_packets = []

for _, row in df_small.iterrows():
    proto = row["protocol_type"]
    src_bytes = int(row["src_bytes"])

    # Random IPs
    pkt = IP(src="192.168.1." + str(random.randint(2, 254)),
             dst="10.0.0." + str(random.randint(2, 254)))

    # Protocol mapping
    if proto == "tcp":
        pkt /= TCP(sport=random.randint(1024,65535), dport=80, flags="S")
    elif proto == "udp":
```

```
    pkt /= UDP(sport=random.randint(1024,65535), dport=53)
  else:
    pkt /= ICMP()

  # Payload simulation
  pkt /= Raw(load="X" * min(src_bytes, 200))

  pcap_packets.append(pkt)

# Save PCAP
wrpcap("nsl_kdd_synthetic.pcap", pcap_packets)
```



◇ Step 6: Download PCAP

from google.colab import files
files.download("nsl_kdd_synthetic.pcap")

```
[ ] import os
    print(os.listdir())

['.config', 'drive', 'nsl_kdd_synthetic.pcap', 'sample_data']

[ ] from google.colab import files
    files.download("nsl_kdd_synthetic.pcap")
```

◇ Step 7: Open in Wireshark

1. Launch Wireshark
2. File → Open → Select nsl_kdd_synthetic.pcap
3. Analyze traffic (filter by TCP, UDP, ICMP, etc.)