Network Traffic Analysis PROJECT

Computer Science L3,
Network & Algorithms

Professors:

Rabih AMHAZ

Javid Khalilov

Members:

Mikayil Shahtakhtinski

Khumar Huseynova

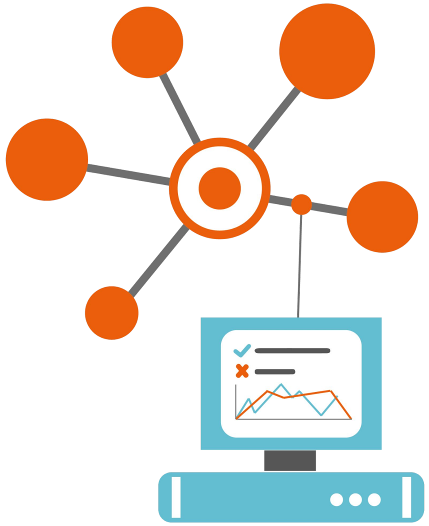
Arif Ahmadli

INTRODUCTION / PROBLEM AND GOAL

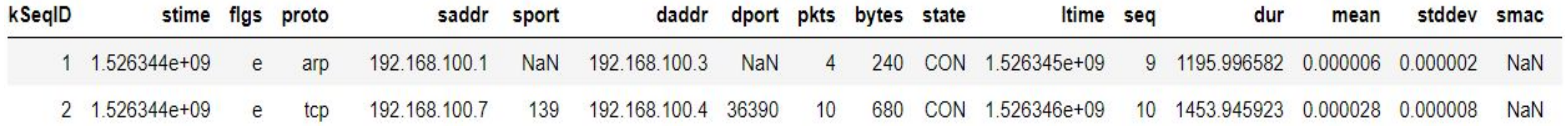
The main problem that we aimed to solve in this project was to gain a better understanding of the behavior and topology of a network based on its event data.

By analyzing this data and creating visual representations of it using NetworkX, we hoped to identify patterns and relationships that could help us gain insights into how the network operates.

The goal of the project was to analyze a dataset using Python libraries such as **Pandas**, **Matplotlib**, **Networkx**, and **Seaborn** and represent the data in a graph format using different approaches.



The dataset used in this project contains network traffic data with various attributes such as source and destination IP addresses, protocol types, and timestamps.



kSeqID	stime	flgs	proto	saddr	sport	daddr	dport	pkts	bytes	state	ltime	seq	dur	mean	stddev	smac
1	1.526344e+09	e	arp	192.168.100.1	NaN	192.168.100.3	NaN	4	240	CON	1.526345e+09	9	1195.996582	0.000006	0.000002	NaN
2	1.526344e+09	e	tcp	192.168.100.7	139	192.168.100.4	36390	10	680	CON	1.526346e+09	10	1453.945923	0.000028	0.000008	NaN

OBJECTIVES

Our objectives in this project were to:

- Read and describe the dataset in detail, including the relationships between different attributes
- Use statistical tools to represent the data using charts and other visualizations
- Create different graph representations of the data using NetworkX, focusing on different attributes
- Automate the process of creating these graph representations so that users can easily choose which attribute to use as nodes and edges
- Create a function to find paths between different nodes in the network
- Use graph analysis tools to calculate degree connectivity, closeness centrality, betweenness centrality, network density, network diameter, and network average path length.

PART 1 - AGENDA

- Read the dataset file from the CSV files
- Describe each attribute in the dataset in the simplest way.
- Describe the relations between the attributes.
- Use statistical tools to represent the data using charts and other tools.

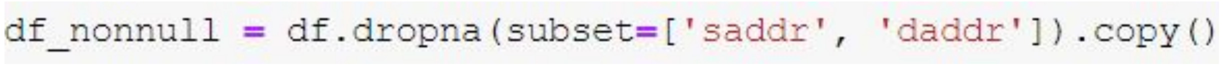
DATA LOADING | DATA MANIPULATION

We combined **three separate datasets** into **one dataframe** to simplify our analysis and allow us to easily explore the relationships between different attributes in the data.

To clean data, create graphs, do manipulations on it, we imported several Python libraries:

- ***Pandas*** was used to analyze and upload data in CSV format.
- ***NetworkX*** was used to create, manipulate, and analyze complex networks.
- ***Matplotlib*** was used to create static, interactive, and animated visualizations.
- ***NumPy*** was used for numerical computing.
- ***Seaborn*** was used for data visualization.
- ***LabelEncoder*** from scikit-learn was used to transform categorical values to numerical values for correlation analysis (**heatmap**).

To clean our data, especially to remove the rows where the IP source and destination IP is absent. There is no event where no IP source and no destination IP available



```
df_nonnull = df.dropna(subset=['saddr', 'daddr']).copy()
```

By analysing the dataset the list of important columns was configured.

MAIN FEATURES IN DATASET

There are 34 main features in our dataset. Each feature from feature.xlxs was analysed and added to our dataset.

- **pkSeqID:** Row Identifier
- **Stime:** Record start time
- **Flgs:** Flow state flags seen in transactions
- **flgs_number:** Numerical representation of feature flags
- **Proto:** Textual representation of transaction protocols present in network flow
- **Saddr:** Source IP address
- **Sport:** Source port number
- **Daddr:** Destination IP address
- **Dport:** Destination port number
- **Pkts:** Total count of packets in transaction
- **Bytes:** Total number of bytes in transaction
- **State:** Transaction state
- **Sum:** Total duration of aggregated records
- **Ltime:** Record last time
- **Seq:** Argus sequence number
- **Sum:** Total duration of aggregated records
- **Min**: Minimum duration of aggregated records
- **Max**: Maximum duration of aggregated records
- **Spkts**: Source-to-destination packet count
- **Dpkts**: Destination-to-source packet count
- **Sbytes**: Source-to-destination byte count
- **Dbytes**: Destination-to-source byte count
- **Rate**: Total packets per second in transaction
- **Srate**: Source-to-destination packets per second
- **Drate**: Destination-to-source packets per second
- **Attack**: Class label: 0 for Normal traffic, 1 for Attack Traffic
- **Category**: Traffic category
- **Subcategory**: Traffic subcategory

DATA ANALYSIS
RELATIONSHIP BETWEEN FEATURES

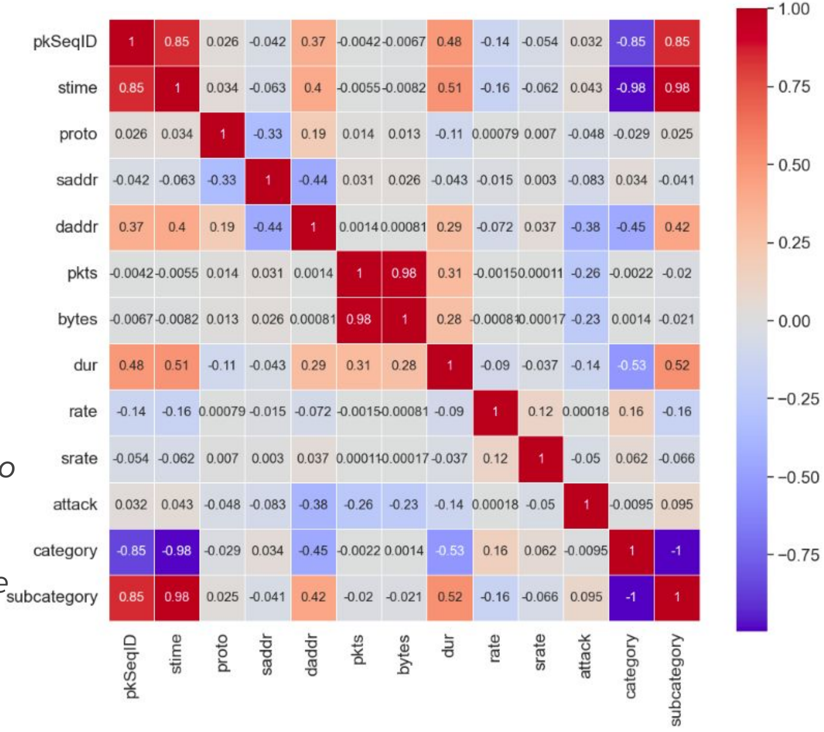
We used a heatmap to visualize the relationships between different attributes in our dataset.

Example of analysis

- **a high positive correlation between "stime" and "category" (0.98) as well as "stime" and "subcategory" (0.98)**
- **a moderate positive correlation between "stime" and "duration" (0.51) and "dur(Record total duration)" and "subcategory" (0.52)**

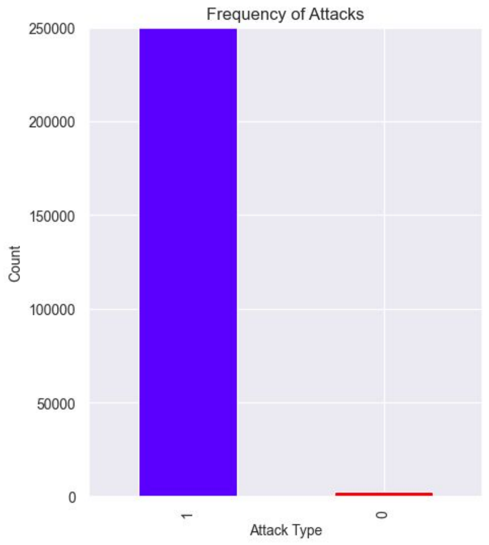
This means that certain types of attacks or subcategories are more likely to occur at specific times or time ranges.

Moreover it means that the duration of a communication can provide some information about the type of attack or subcategory, but the relationship is not as strong as the relationship between start time and category/subcategory.



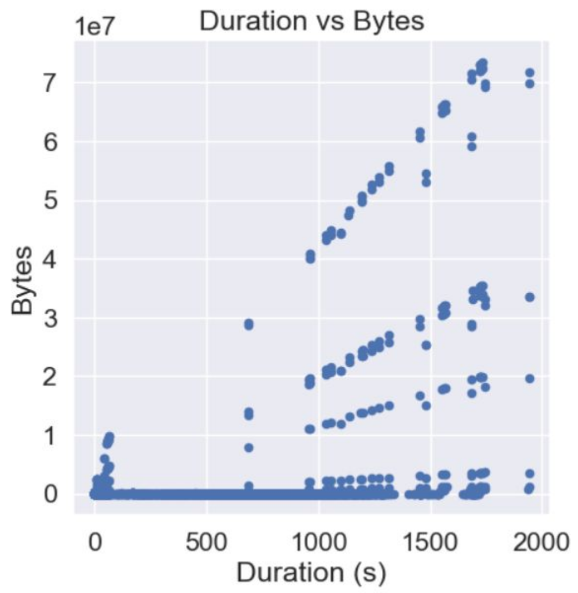
	pkSeqID	stime	proto	saddr	daddr	pkts	bytes	dur	rate	srate	attack	category	subcategory
pkSeqID	1	0.85	0.026	-0.042	0.37	-0.0042	-0.0067	0.48	-0.14	-0.054	0.032	-0.85	0.85
stime	0.85	1	0.034	-0.063	0.4	-0.0055	-0.0082	0.51	-0.16	-0.062	0.043	-0.98	0.98
proto	0.026	0.034	1	-0.33	0.19	0.014	0.013	-0.11	0.00079	0.007	-0.048	-0.029	0.025
saddr	-0.042	-0.063	-0.33	1	-0.44	0.031	0.026	-0.043	-0.015	0.003	-0.083	0.034	-0.041
daddr	0.37	0.4	0.19	-0.44	1	0.0014	0.00081	0.29	-0.072	0.037	-0.38	-0.45	0.42
pkts	-0.0042	-0.0055	0.014	0.031	0.0014	1	0.98	0.31	-0.0015	0.00011	-0.26	-0.0022	-0.02
bytes	-0.0067	-0.0082	0.013	0.026	0.00081	0.98	1	0.28	-0.00081	0.00017	-0.23	0.0014	-0.021
dur	0.48	0.51	-0.11	-0.043	0.29	0.31	0.28	1	-0.09	-0.037	-0.14	-0.53	0.52
rate	-0.14	-0.16	0.00079	-0.015	-0.072	-0.0015	-0.00081	-0.09	1	0.12	0.00018	0.16	-0.16
srate	-0.054	-0.062	0.007	0.003	0.037	0.00011	0.00017	-0.037	0.12	1	-0.05	0.062	-0.066
attack	0.032	0.043	-0.048	-0.083	-0.38	-0.26	-0.23	-0.14	0.00018	-0.05	1	-0.0095	0.095
category	-0.85	-0.98	-0.029	0.034	-0.45	-0.0022	0.0014	-0.53	0.16	0.062	-0.0095	1	-1
subcategory	0.85	0.98	0.025	-0.041	0.42	-0.02	-0.021	0.52	-0.16	-0.066	0.095	-1	1

STATISTICAL ANALYSIS



Attack 1 has the highest frequency, with a bar that reaches the top of the graph.

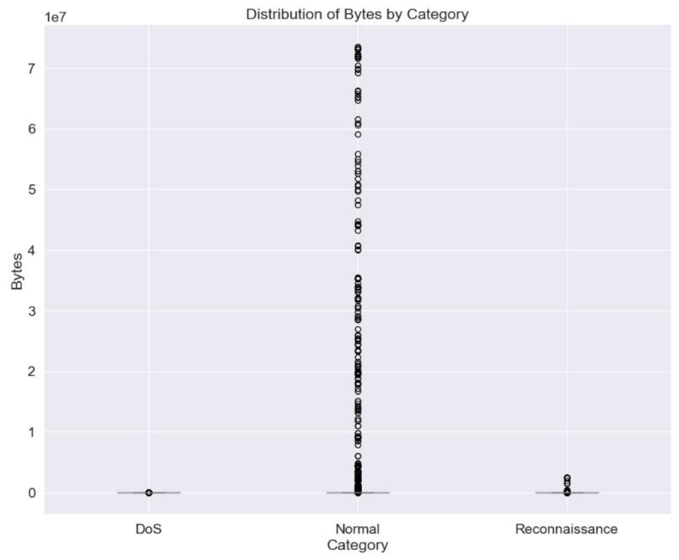
The red bar, representing another type of attack, is very small.



There is positive correlation between two variables

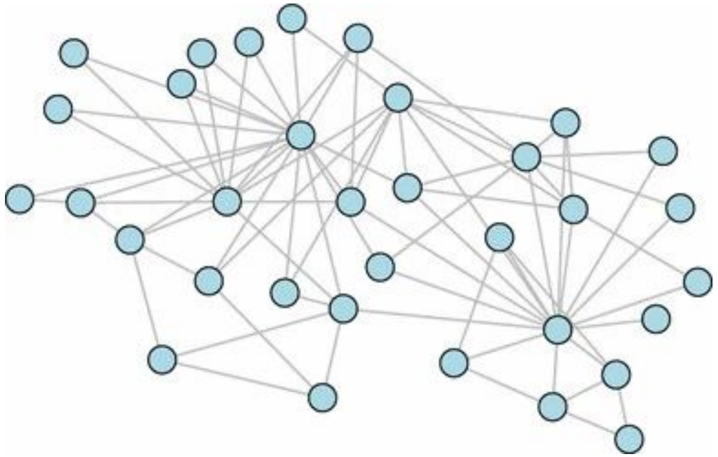
As the duration of transmission increases, so does the number of bytes sent

Linear relationship => a consistent data transfer rate between the nodes in the network



The normal category seems to have the highest median byte count and the widest range of byte counts, with many outliers beyond the whiskers.

PART 2 - AGENDA



- Using your experience from the lab session and the lab networkx, present the data in a graph format using:

  a. First representation uses “IP” as a node and protocol as edges.

  b. Second representation uses event (row) as a node.

  c. Third representation uses any attribute that you decide as a node.

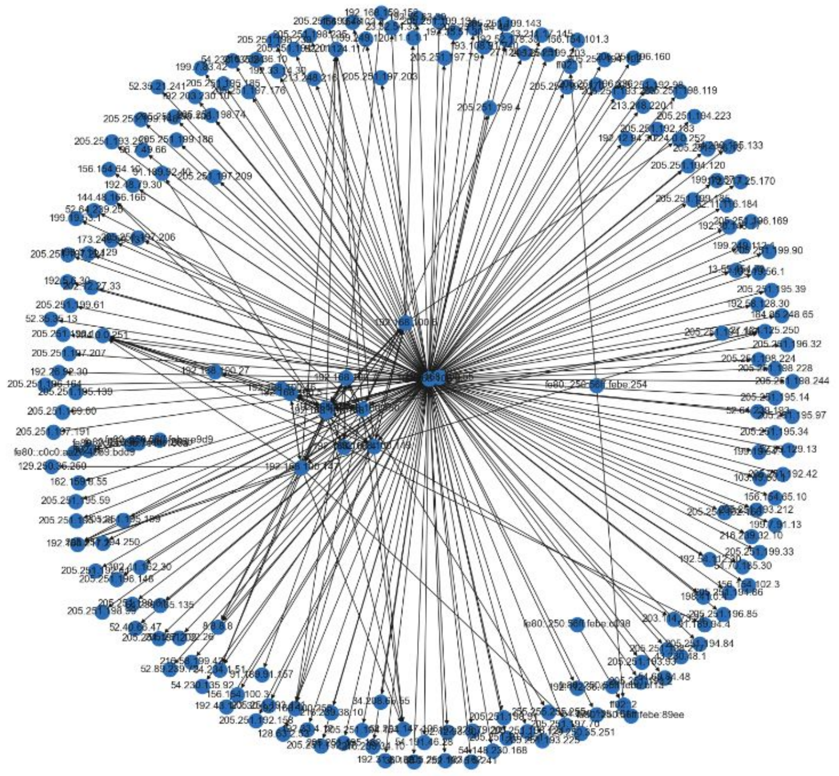
- Automate the process of representation using graphs to let users decide which attribute as node and which attribute as edges.

“IP” as a node and a protocol as an edge

The graph has the **source IP addresses** and **destination IP addresses as nodes** and the **protocols** as **edges.**

Here is the algorithm of our work

1. Create an empty directed graph object using NetworkX library.
2. Extract all the **unique** source IP addresses and destination IP addresses from the dataset and add them as nodes to the graph using the ***add_nodes_from*** function.
3. Create edges in the graph between source IP and destination IP pairs for **each unique combination** of protocol used by them. This is done by iterating through the rows of the dataset and adding edges for each unique combination of source IP, destination IP, and protocol.
4. Plot the graph using the **spring_layout** function.



Event (row) as a node

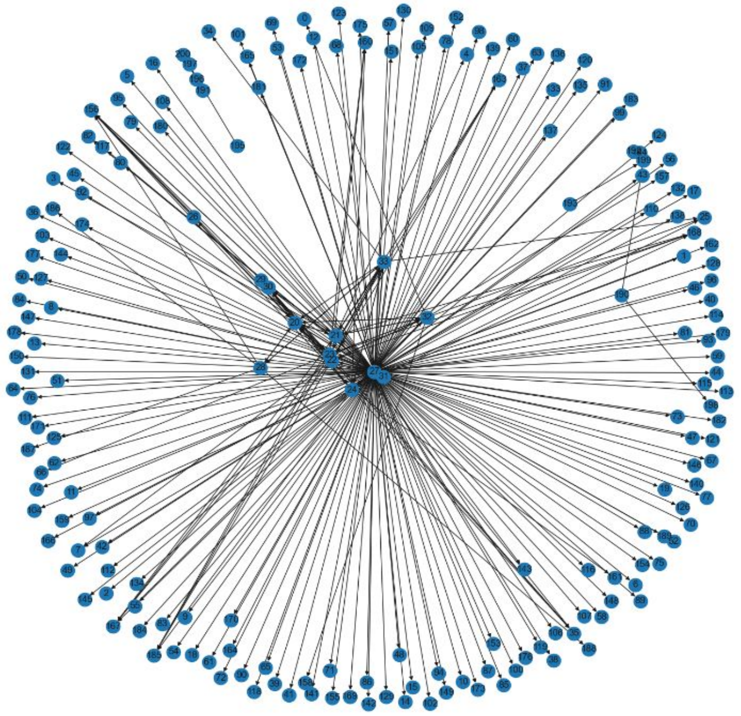
1. Each row in the network traffic data represents a **unique event** .These events can be thought of as nodes in a graph, where each node represents a specific event.

2. To create the graph, we first create an **empty directed graph object** using the **NetworkX** library. We then extract all the **unique** source IP addresses and destination IP addresses from the dataset and add them as nodes to the graph using the **add_nodes_from** function.

3. Next, we create edges between nodes based on the source and destination addresses of each network event. To do this, we iterate over each row in the dataset and create an edge between the source and destination nodes represented by that row.

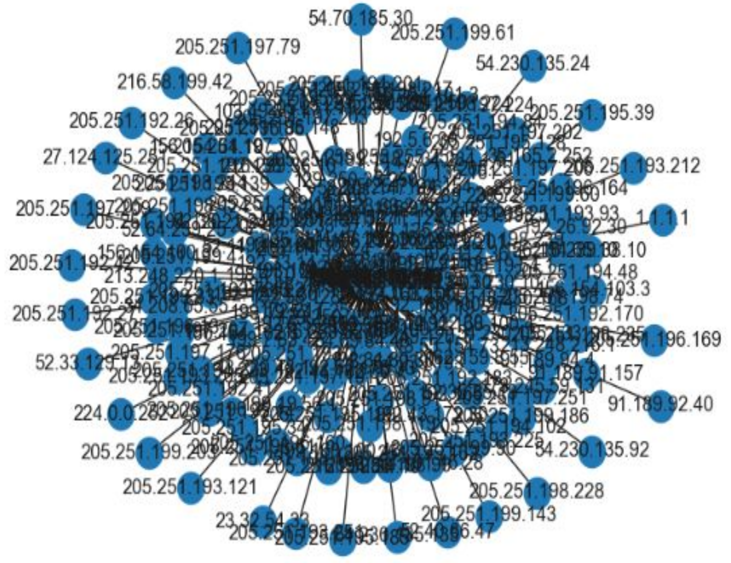
4. Finally, we plot the resulting graph using the **spring_layout** function. This graph shows the interactions between network events, where each edge represents a communication between two events.

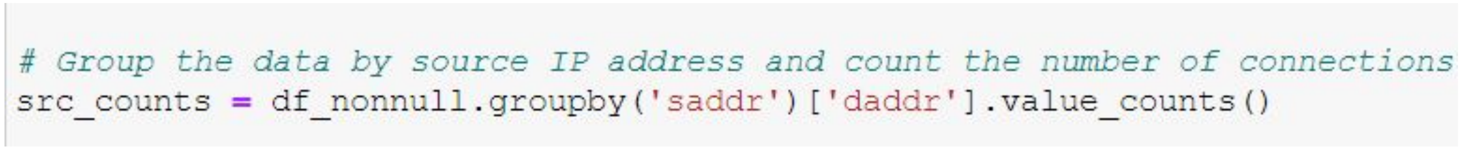
We can use this graph to identify devices that are communicating frequently with each other or devices that are communicating with a large number of other devices.



ANY ATTRIBUTE THAT YOU DECIDE AS A NODE

For the third representation, we can choose any attribute as a node. Let's say we want to create a graph where each node represents a unique source IP address and the edges represent the number of connections between them. To create this graph, we can use the groupby function to group the data by source IP address and count the number of connections between each pair of source IP addresses.

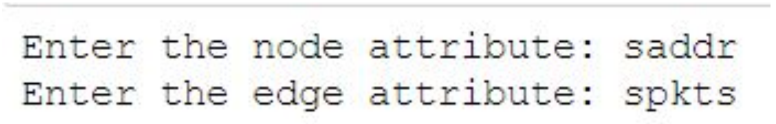




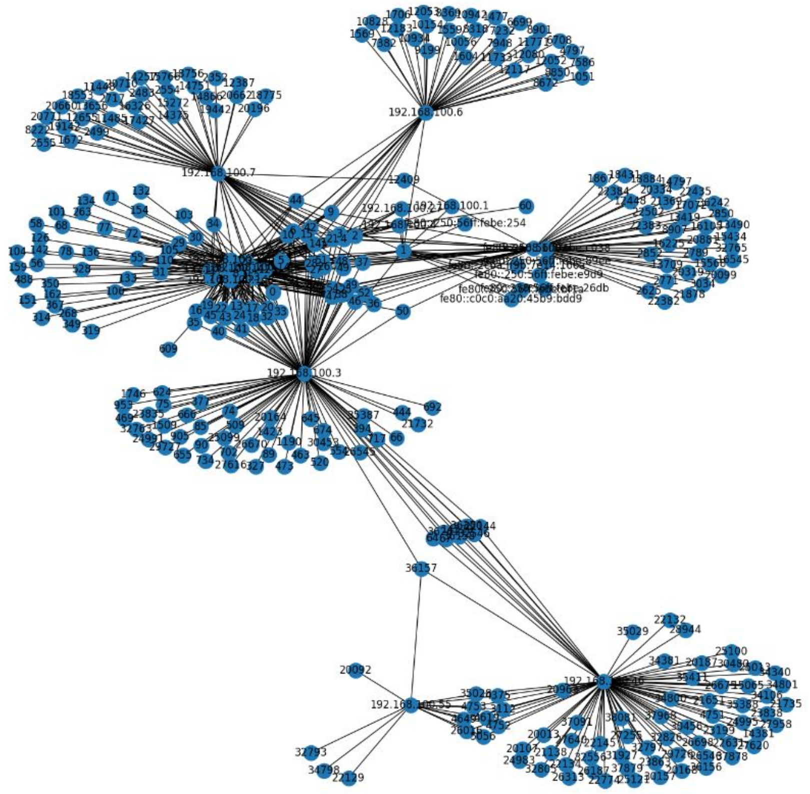
```
# Group the data by source IP address and count the number of connections
src_counts = df_nonnull.groupby('saddr')['daddr'].value_counts()
```

AUTOMATED PROCESS OF DECISION

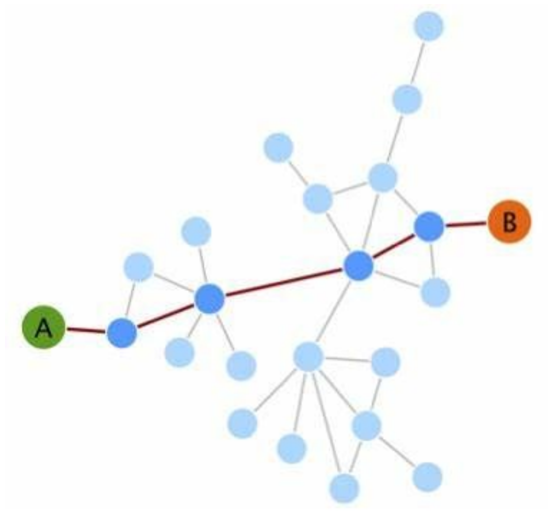
In this section we allowed the user to decide what will be taken as a node and what will be considered as an edge. Likewise other algorithms **empty directed graph object** is created using the **NetworkX** library and then nodes are added to the graph using **add_nodes_from** method, whereas edges are added using **add_edge.** In the end of this process the graph is plotted using spring layout. To allow the user to choose nodes and edges we specified the input option.



```
Enter the node attribute: saddr
Enter the edge attribute: spkts
```



PART 3 - AGENDA



- Create a function to find anyone's path from a given origin to a given destination

- Create a function to evaluate how many packets were sent using a path between IPs

FIND PATH

It uses depth first search to find a path between the starting and ending nodes in the graph.

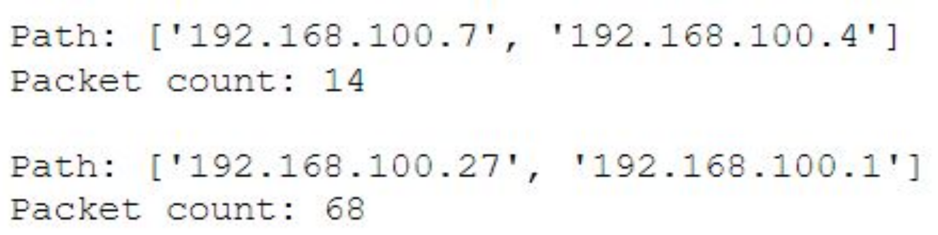
The function recursively explores the graph by visiting each neighbor of the current node and checking if it leads to the destination node.

- **If a path is found**, the function returns the path as a list of nodes.
- **If no path is found**, it returns None.

COUNT PACKETS

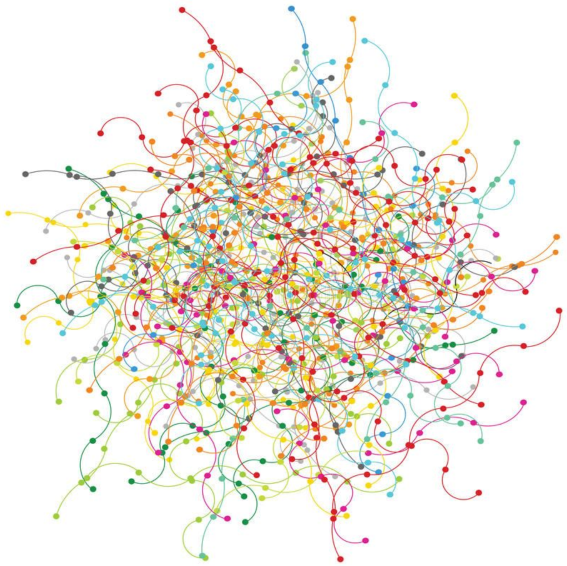
The **count_packets()** function takes in a dataframe that represents packet transmissions between nodes and a path between two nodes represented as a list of nodes.

- creates a mask that filters the dataframe to only include rows where the source and destination nodes match current pair of nodes in the path, respectively.
- iteratively adds more conditions to the mask to ensure that each subsequent node in the path is connected to the previous node through a transmission in the dataframe.



```
Path: ['192.168.100.7', '192.168.100.4']
Packet count: 14

Path: ['192.168.100.27', '192.168.100.1']
Packet count: 68
```

PART 4 - AGENDA

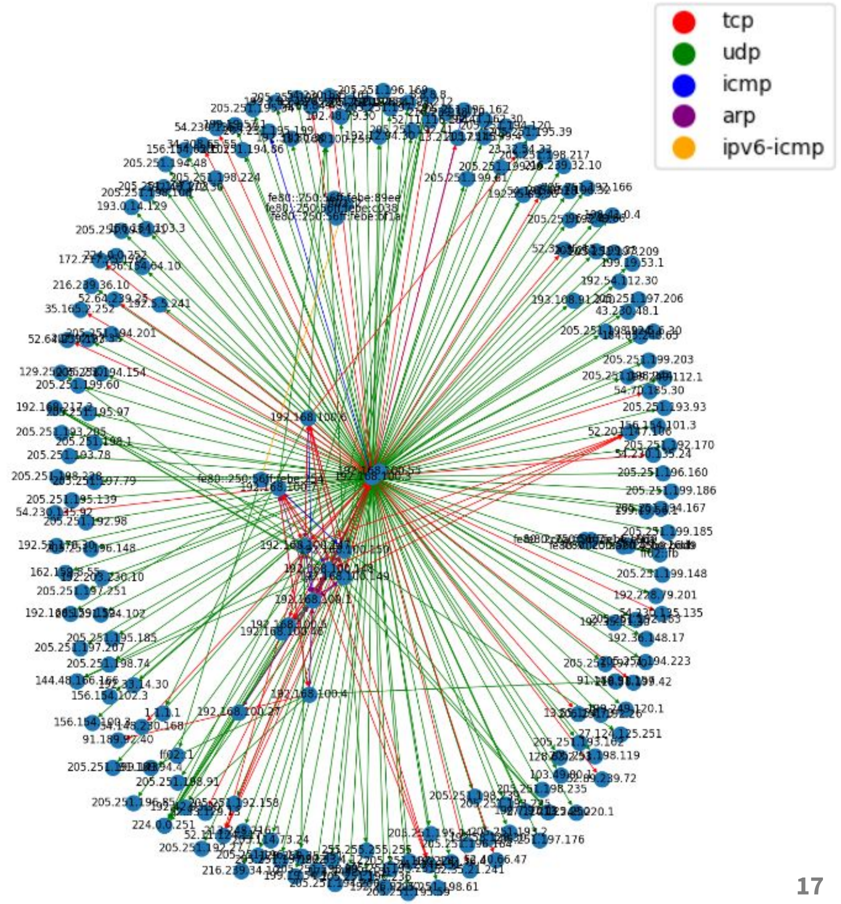


- Use colors to the plot to represent differences between protocols
- Automate the coloring so the user can choose the attribute and then the representation color based on the difference between values.

SOURCE IP ADDRESS AS A NODE, PROTOCOL AS AN EDGE

ADDING COLORS TO DIFFERENTIATE PROTOCOLS

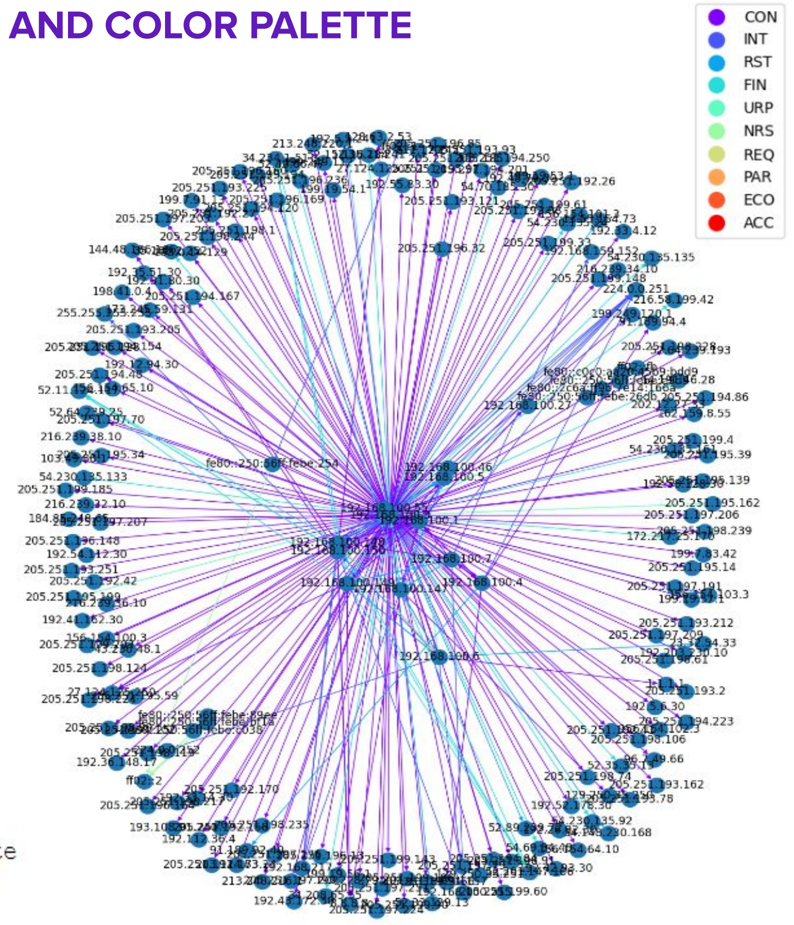
Nodes are added to the graph for each source and destination IP address in the data, and edges are added for each communication between nodes, with the protocol type as an edge attribute. The edges are colored based on their protocol type using a custom color dictionary, and a legend is added to the plot to show the colors for each protocol.



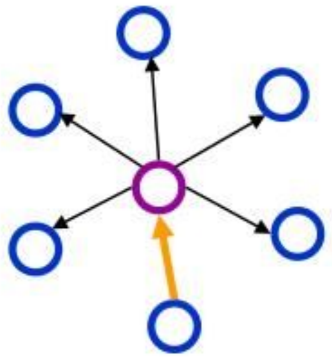
AUTOMATE THE COLORING | USER CHOSEN EDGE AND COLOR PALETTE

The graph generated by this code represents a network of communication between nodes, where each node is either a source or a destination IP address. The edges between the nodes represent the communication paths between them, and the color of each edge represents a chosen attribute of the communication, such as the size of data transmitted or the state of the connection. By visualizing this information, the graph can help to identify patterns or anomalies in the communication patterns and gain insights into the behavior of the network.

```
Enter the attribute to color the graph(proto, bytes, state...) by: state
Enter the color scale (e.g., viridis,plasma,magma,rainbow...): rainbow
```



PART 5 - AGENDA



- Using the graphs tools give the values or the matrices of values for :

  a. Degree Connectivity

  b. Closeness Centrality

  c. Betweenness Centrality

  d. Network Density

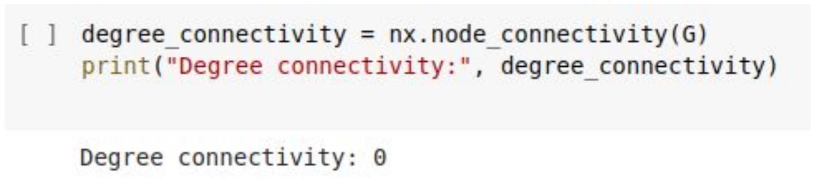
  e. Network Diameter

  f. Network Average Path Length

Analysis were done on a graph, where IP source and destination are nodes and Protocols are edges

DEGREE CONNECTIVITY

Degree connectivity refers to the number of edges that a node must remove in order to disconnect a graph.

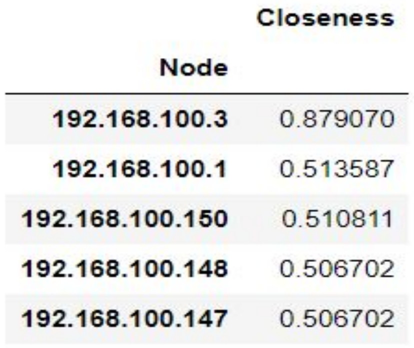


```
[ ] degree_connectivity = nx.node_connectivity(G)
    print("Degree connectivity:", degree_connectivity)

    Degree connectivity: 0
```

Because **connectivity is zero**, the graph is a connected graph, meaning that there is a path between any two nodes in the graph.

CLOSENESS CENTRALITY

It measures how quickly a node can reach other nodes in the graph. A node with high closeness centrality is one that is located close to many other nodes and can quickly access them. **The nodes with the highest closeness centrality values are '192.168.100.1' and '192.168.100.3'.** These nodes are likely central to the network and are important for the overall connectivity of the graph.



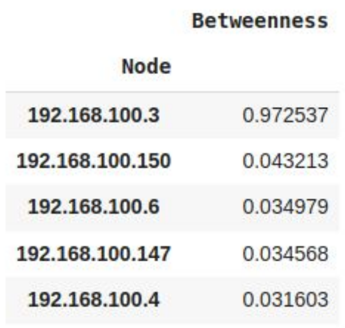
Node	Closeness
192.168.100.3	0.879070
192.168.100.1	0.513587
192.168.100.150	0.510811
192.168.100.148	0.506702
192.168.100.147	0.506702

Analysis were done on a graph, where IP source and destination are nodes and Protocols are edges

BETWEENNESS CENTRALITY

It is a measure used in graph theory that quantifies the importance of a node within a network. It is based on the idea that nodes that occur on many shortest paths between other nodes have higher importance in the network.

For instance, the node with **IP address 192.168.100.3 has a high betweenness centrality of 0.972537**, indicating that it is an important node in the network.



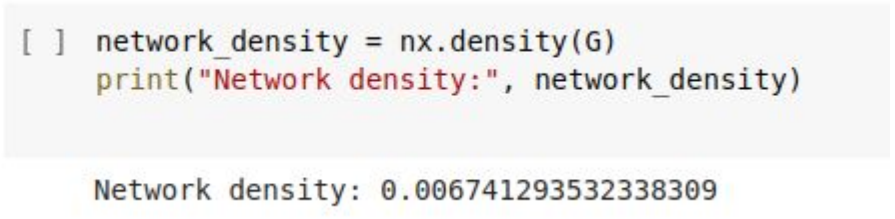
Node	Betweenness
192.168.100.3	0.972537
192.168.100.150	0.043213
192.168.100.6	0.034979
192.168.100.147	0.034568
192.168.100.4	0.031603

NETWORK DENSITY

It is a measure of how well-connected a network is. Specifically, it measures the proportion of all possible connections between nodes that actually exist in the network.

The network density is 0.006741293532338309, which indicates that only a small fraction of all possible connections between nodes in the network actually exist.

Overall, a network with low density may be less efficient at transmitting information and less robust to node failures



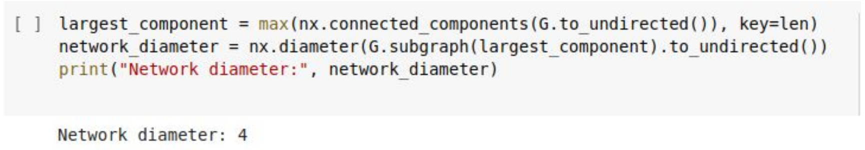
```
[ ] network_density = nx.density(G)
    print("Network density:", network_density)

    Network density: 0.006741293532338309
```

Analysis were done on a graph, where IP source and destination are nodes and Protocols are edges

NETWORK DIAMETER

It is a measure of the longest path between any two nodes in a network, represented by the maximum number of edges that must be traversed to get from one node to another node.

A network diameter of 4 indicates that the longest path between any two nodes in the network is at most 4 edges long. This suggests that the network is relatively well-connected and efficient for transmitting information or other flows between its nodes.

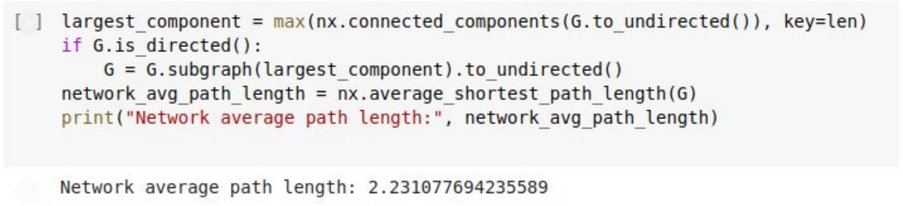


```
[ ] largest_component = max(nx.connected_components(G.to_undirected()), key=len)
    network_diameter = nx.diameter(G.subgraph(largest_component).to_undirected())
    print("Network diameter:", network_diameter)

    Network diameter: 4
```

NETWORK AVRG. PATH LENGTH

It is a measure of the average number of edges that must be traversed to get from one node to another node in a network.

The network average path length of 2.231077694235589 means that on average, it takes a little over two edges to connect any two nodes in the largest connected component of the network. This is a relatively small average path length, suggesting that the network is relatively well-connected and efficient for transmitting information or other flows between its nodes.



```
[ ] largest_component = max(nx.connected_components(G.to_undirected()), key=len)
    if G.is_directed():
        G = G.subgraph(largest_component).to_undirected()
    network_avg_path_length = nx.average_shortest_path_length(G)
    print("Network average path length:", network_avg_path_length)

    Network average path length: 2.231077694235589
```

CONCLUSION

Moving forward to the conclusion, throughout the project we worked on analysis of the network dataset and at end end used different network data measurement to analyse one of the network graphs created namely as a representation of the network traffic data using IP addresses as nodes and Protocols as edges.

Analysis of Correlation between Features and analysis of Network Data:

Beginning with the heatmap, it shows that there is a high positive correlation between the beginning of the communication and the type of attack/subcategory. This implies that the start time of a communication can be used as an indicator in terms of identifying the type of attack/subcategory. Besides the start time communication, the duration of it can also be used as an indicator, but the only concern is that it will not be as efficient.

Moving forward to bar chart, it represents the frequency of appearance of various attacks in the dataset. The analysis shows that Attack 1 was the most common comparing to other attacks. Additionally, the majority of packets have a duration of 0 to 500 seconds, with a single bar in the histogram reaching the maximum frequency value.

CONCLUSION

In the graph representation of the network traffic data, it is found that the nodes with **the highest closeness centrality values are '192.168.100.3' and '192.168.100.1'**. These nodes are likely central to the network and are important for the overall connectivity of the graph. Additionally, the node with I**P address 192.168.100.3 has a high betweenness centrality of 0.9725**, indicating that it is an important node in the network. The other nodes with **high betweenness centrality** values are likely also important in the network.

Finally, the analysis shows that the network has a **low degree connectivity,** which means that it is **not easy to disconnect the graph** by removing nodes. The network density is also relatively low, which suggests that there are not many connections between nodes in the network.

In summary, the analysis provides insights into the correlation between different features in the dataset and the network traffic data. The findings can be useful in identifying the most common type of attack in the dataset and the nodes that are central to the network and important for its overall connectivity. The low degree connectivity and network density suggest that the network may not be very robust and may be vulnerable to attacks or disruptions.

THANKS FOR ATTENTION