Capturing Local Network Traffic

Group members: Josh Alanguilan

Goal of the Project: To capture network traffic for a specified duration using the "Scapy" library in Python. The Python script captures packets traversing the network interface and processes them to extract specific information such as source IP, destination IP, protocol, website names (extracted from URLs), and packet length. The script records the network traffic information and formats it into a CSV file. The CSV file can be used for network analysis, troubleshooting, or further data processing and analysis.

Details of Code: To run the code, the file "NetworkTraffic.py" will need to be extracted from the zip file into a directory. There are multiple ways to run the script; right click the file and open with "Python 3.10", this works 50% of the time I do not recommend running the script with this method. If Git Bash is installed, you can right click in the directory the file is in and click "Git Bash Here", then run the script with the command "python NetworkTraffic.py". You can also run the script in Command Prompt by navigating to the directory containing the script using the cd command, then in the directory use the command "python NetworkTraffic.py".
You will need Python installed, preferably Python 3.10. The libraries you will need is BeautifulSoup4, Requests, Scapy and Pandas. The Pandas library includes functions that perform cleaning tasks on the captured data such as removing duplicates and values with 'Unknown'. These cleaning tasks have been added to the script, without them the CSV file will have thousands of captured packets. The script and an example CSV file can be found on the GitHub page:
https://github.com/JAlanguilan/Network-Traffic-Data.git

Methodology: The question that project tries to answer is, how can we monitor network traffic and record necessary data in an efficient manner? A brute force answer to this question would be to capture packets with Wireshark for a time and manually pick out specified data such as IP addresses or destination ports. To avoid recording the data manually I decided to create a Python script that would record network traffic with Wireshark and format the data into a CSV file. To capture network packets I decided to use the Scapy library, using the 'sniff()' function for a defined duration. I needed a method to store the packets and format the data. I found the 'Pandas' library, which creates a DataFrame and extracts packet information and structures the data. After running the script multiple times, the script had sniffed thousands of packets including duplicates and unknown values. I wanted to clean up the recorded data, so I used the cleaning operations that are included with the Pandas library. I included the BeautifulSoup4 and requests libraries that will attempt to find the website/domain name.

Evaluation results:

Time:

The first column of the DataFrame shows the timestamps of the packets that are captured. The amount of packets captured each second varied due to the websites visited. For example, at 2:19:55 there was only 3 packets captured. There were only 2 websites open at this time. Compared to time 2:20:05, which captured 189 packets in that second. At this time of capture I had opened a YouTube video, which is the result of the large number of packets.

Source and Destination IP:

The second and third column of the DataFrame show the Source and Destination IP. The values of the source and destination Ips show a clear pattern of back-and-forth communication between my IP address and a servers IP address. We can see how a “conversation” between my device and a server starts with a request sent to the server from my IP address. The following packet is the response packet sent from the server. By analyzing the pattern of IP addresses, it can be inferred that some packets being sent are too large and are broken down into multiple packets. We can see this with packets 262-264, the source IP 142.250.68.10 send 3 large packets in succession. The source IP 142.250.68.10 is owned by YouTube. With this information we can assume that these 3 large packets contained the data for the video that was being loaded at the time.

Protocol:

The fourth column of the DataFrame shows the protocol of the packets. The script was designed to parse the captured packets from Wireshark and only record the IP protocol packets. This is why IP is the only protocol that is recorded for this section. There a few packets were the script records my IP address as the protocol. This is small issue where the script does not handle certain packets or protocols correctly.

Domain/Website Name:

The fifth and sixth column of the DataFrame is supposed to show the domain and website name of the packets. Due to the limitations of the libraries used in the script, it is not able to extract the domain or website names from websites that use other protocols or encrypted connections.

Length:

The last column of the DataFrame displays the length of the packet captured. The larger packet sizes indicate higher bandwidth utilization as they require more data to be transmitted. The length of the packet can show if larger packets were fragmented due to network constraints.

	A	B	C	D	E	F	G
1	Timestamp	Source IP	Destination IP	Protocol	Website Name	Domain Name	Length
2	12/12/2023 2:19	192.168.50.211	104.99.48.162	IP	not a valid url	Not a valid URL	55
3	12/12/2023 2:19	104.99.48.162	192.168.50.211	IP	not a valid url	Not a valid URL	66
4	12/12/2023 2:19	192.168.50.211	67.220.240.31	IP	not a valid url	Not a valid URL	55
5	12/12/2023 2:19	67.220.240.31	192.168.50.211	IP	not a valid url	Not a valid URL	54
6	12/12/2023 2:19	192.168.50.211	224.0.0.22	192.168.50.211	not a valid url	Not a valid URL	54
7	12/12/2023 2:19	192.168.50.211	3.232.144.130	IP	not a valid url	Not a valid URL	127
8	12/12/2023 2:19	3.232.144.130	192.168.50.211	IP	not a valid url	Not a valid URL	145
9	12/12/2023 2:19	192.168.50.211	3.232.144.130	IP	not a valid url	Not a valid URL	54
10	12/12/2023 2:19	3.232.144.130	192.168.50.211	IP	not a valid url	Not a valid URL	145
11	12/12/2023 2:19	192.168.50.211	3.232.144.130	IP	not a valid url	Not a valid URL	66

Table of the first 10 packets captured.

Workload: I am the only member of my team.

Tools: I used Wireshark to capture network packets. I used Visual Studio Code to write the Python script. I used Git Bash to run the script. The libraries used are Scapy, Pandas, Requests and BeautifulSoup4. The results were stored in a CSV file which can be viewed in Excel.

Challenges: A challenge I faced was extracting the website name or domain name. The script relies on HTTP/HTTPS traffic and DNS queries to identify websites. If the captured traffic predominantly uses other protocols or encrypted connections, extracting website names fails. With each attempt at running the script, I could never extract the website names. Due to the limitations of the libraries, the script will only write the website names for the websites that allow it. Since most websites structures are complex or dynamically generated, the script will most likely not return a valid website name.

Future Directions: I am pursuing a career in cybersecurity which I will gain networking skills and knowledge that I can later apply to this project. I can improve this project by including operations that capture traffic from other devices from my local network, recording information from packets to learn more about the sender/receiver and use machine learning to discover patterns in the captured packets. If I was not working a full-time job alongside college classes and had more time, I would have created a comprehensive program instead of a simple Python script. The program would include an interface that would allow the user to choose what data is recorded and how it is formatted. The program would be a simpler version of the Security Information and Even Management (SIEM) tools that are used to capture and record network traffic. Some interesting project topics could be Optimizing traffic routing algorithms or improving network congestion control algorithms.