Internet Classification Tool

Computer Networks Term Project

Fatima Hasan
L17-4020
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore, Pakistan
l174020@lhr.nu.edu.pk

Nuzha Khaled
L17-4162
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore, Pakistan
l174162@lhr.nu.edu.pk

Samra Fakhar
L17-4031
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore, Pakistan
l174031@lhr.nu.edu.pk

Shanzay Gauhar
L17-4236
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore, Pakistan
l174236@lhr.nu.edu.pk

***Abstract*—The Internet is an ever-evolving technology which has made it complex to comprehend the structure of the network. To understand the structure of networks and how protocols work in the real world, internet classification tools were the need of the time. These tools assist in breaking up the traffic into categories by various parameters such as port number, protocol etc. As the internet is widely used and fast growing it poses a challenge for us to understand or characterize the traffic without the assistance of such classification tools. In this project, we identify and quantize different types of traffic/packets such as DNS, VoIP, HTTP/S, RTP, FTP, etc. from live packet captures obtained through windump and then used the pcap file created as a result to perform classification steps. The report presents the Network Traffic Classification based on Port Numbers identification. The application is developed on a Pycharm application using python language. Inbuilt libraries are used to develop graphical user interfaces that are used to show percentage breakdown of traffic for each application over a given time. There has already been a lot of work done in this field and most of them involve machine learning concepts. However, in this project we are using Pyshark - a wrapper for Tshark to analyze a capture file - pcap. Basically, Tshark is a command line interface for the Wireshark, hence we are using its functionality to drop out unnecessary fields while analyzing a certain packet. The report concludes as many protocols use fixed port numbers it is an efficient and effective way to classify traffic on the network. However, the results for applications that do not use well-known ports are not as accurate as the former..**

Keywords—Internet Traffic, Packet Classification, PyShark, WinDump, Live Packet Capture, Data Visualization

I. INTRODUCTION

Network traffic classification is a process which categorizes computer network traffic according to either port number or protocol into a number of traffic classes [3]. It is an important problem in computer networks. The performance of a network highly depends on efficient resource management which in turn relies on analyzing network trends and planning and designing a network based on that analysis. Classification of network traffic packets is very important for Internet Service Providers (ISPs) for determining which type of network applications flow in the network. Moreover, it is also used in a multitude of applications including firewalls, intrusion detection systems and status reports [1].

The internet has been growing exponentially over the years and it has become very important for us to understand and even predict the type of traffic that is flowing, to increase our utilization and performance. There are many different types of traffic including, but not limited to, DNS, HTTP/S, VoIP, RTP, FTP, SMTP, TELNET, SNMP and much more.

There are different methods for the identification of the type of traffic. One of these is port number based, and this is the approach that we will be adopting.

A port number is used to identify the specific process that a network message is to be sent to. Both TCP and UDP use port numbers as an endpoint to a logical connection. When a client message is sent to a server, the port number identifies the specific server program that it is to be forwarded to. The range of port numbers is from 0 to 65536, but the port numbers from 0 to 1023 are reserved for privileged services. For example, port 80 is reserved for HTTP. The Internet Assigned Numbers Authority (IANA) is responsible for maintaining the assignments of these port numbers. Some other reserved port numbers are shown in the table below:

Port Number	Description
20	FTP -- Data
21	FTP -- Control
23	Telnet
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name System (DNS)
109	POP2

110	POP3
115	Simple File Transfer Protocol (SFTP)
118	SQL Services
161	SNMP
179	Border Gateway Protocol (BGP)
443	HTTPS
546	DHCP Client
547	DHCP Server

Table 1. Some Common Protocols and their Port Numbers

In this project, we will be focusing on the identification and quantization of different protocols from live captures obtained through windump. The method we will be following to detect the type of packets is identifying the port number of each packet, since each protocol uses a fixed, pre-defined port number. By analyzing the packets and identifying the source and destination port numbers, it will be easy to classify which protocol is being used. This is the fastest and simplest way to classify the traffic and keeping in view our resources, it is the best option. It is also quite an accurate method since these port numbers are fixed for the different protocols. The percentage breakdown of each type of traffic will be depicted through the use of graphs, on a user-friendly GUI. The complete method and evaluation are described in this report.

II. Background

According to many reports, there are many methods of successfully classifying network traffic. These include: port-based, payload-based, flow-statistics based and behavior based. We will be implementing port-based classification, but it is useful to look at the other methods as well.

Payload-based classification uses Deep Packet Inspection (DPI) to identify the type of traffic. Well-known patterns exist inside packets of the same application. This approach identifies and stores these patterns inside a database and then accesses the DB to detect the type of packet by looking at the contents. This method performs well for many types of applications like HTTP, FTP and SMTP. It is used in many software products as well as open source projects, one of which can be viewed here (http://tstat.tlc.polito.it/). Moreover, it can be used to detect network anomalies and is used in intrusion detection systems [11].

But it is not without its problems. One issue is that it does not work well with the applications that do not have well known patterns such as P2P applications and multimedia [1]. Moreover, it has an additional cost of maintaining the database with up-to-date patterns. Additional cost of string and regular expression matching cannot be ignored either. It is also getting increasingly difficult to access the raw payload since most of the transmitted data is being encrypted [2]. A disadvantage of this method is that it invades the privacy policies of the users and this data may be used against them.

The other method of identifying the internet traffic is by using flow-based statistics. It uses the information that is available in packet headers for example length of the packet, TCP window size, packet interarrival time etc. This information is then used by different machine learning techniques to predict the type of traffic. The machine learning methods include naive Bayes, K-nearest neighbors, artificial neural networks, clustering, regression and support vector machines [1]. These methods can either be supervised or unsupervised. The supervised methods classify the input according to pre-defined outputs whereas the unsupervised methods such as clustering only separate the samples according to their similarities. This method is better than the payload-based method, since it does not rely on the complete contents of the packet and hence, is not affected by encryption to that extent.

The flow-based classification methods have been increasing in popularity since the rise of machine learning and artificial intelligence. There have been many works that apply neural networks but are all different from our approach here.

In [4], they have used a multi-layer perceptron (MLP) with zero/one hidden layer and they use this to apply a Bayesian analysis. The accuracy that they have received for 10 labels is above 99%.

The use of MLP has been quite popular among researchers as it is the primary classification method in both [5] and [6] as well. The accuracy of these have been above 90% and they have suggested further improvements too. In [5], error correcting output codes have been used with MLP classifiers and the accuracy they achieved was 93.8 for 5 labels. As opposed to this, a particle swarm optimization algorithm is used in [6] to classify 6 labels with an accuracy of about 96%.

The most accurate out of all these researches has been [2]. They have employed a combination of Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). It provides the best detection results as the accuracy is around 99.59% for 15 labels. By only using the high-level header data, not including the IP addresses since they are confidential, great accuracy has been achieved in the machine learning area.

Moving on, the behavioral classification method looks at the entire traffic received by an endpoint in the network. The traffic patterns are then examined to identify the application running on the target host. Different applications have different patterns for example P2P applications contact a number of peers using the same port for each host, whereas a Web server is contacted by clients with multiple parallel connections.

[7,8] and other related works employ different heuristics like number of contacted ports or transport layer protocols used to characterize the traffic patterns at different detail levels such as functional, application and social. [9,10] have shown in their study that P2P and client-server applications have extremely different behavioral patterns which can be used to classify these classes even in the network core.

Coming towards the port-based method, according to some articles, the port-based method might not provide very reliable results in the near future since the services of the

internet are growing and the new applications may not have a well-known port number assigned by IANA. It would become difficult to classify such traffic using this method. However, for the applications that do have reserved port numbers, this method would provide 100% accurate results, and in some cases, it may be important to only classify some applications, and not all. In such circumstances, this method would provide very good results.

III. Design And Methodology

We have developed this system for windows. It is developed in python. The Integrated Development Environment (IDE) used for development is PyCharm. Moreover, it uses windump as a supplementary program for data capture. A .pcap file is generated upon running the tool. This contains the captured data packets.

Our tool uses port numbers of the packets to classify them on the basis of their application layer protocol. A port number is used to identify the specific process that a network message is to be sent to. Both TCP and UDP use port numbers as an endpoint to a logical connection. When a client message is sent to a server, the port number identifies the specific server program that it is to be forwarded to. The range of port numbers is from 0 to 65536, but the port numbers from 0 to 1023 are reserved for privileged services. For example, port 80 is reserved for HTTP. Almost all application layer protocols use fixed port numbers to exchange data.

We will be focusing on the identification and quantization of different protocols from live captures obtained through windump. Using windump live data ais captured and kept in a file. quantization of different protocols from live captures obtained through windump. The method we will be following to detect the type of packets is identifying the port number of each packet, since each protocol uses a fixed, pre-defined port number. By analyzing the packets and identifying the source and destination port numbers, it will be easy to classify which protocol is being used. The file containing captured packets is then read and used to visualize the data in form of a table depicting all the packets captured as well as a bar chart and pie chart.

We have developed a user-friendly GUI. Using this user can start capturing the data packets at a given point in time on pressing “Start Capture” button. Windump captures as long as the user does not specify it to stop. This is achieved through the user pressing the “stop capture” button. Once stopped, the table is viewed. On this very screen depicting table, two buttons provide the option to display the data graphically.

A. Python Libraries

We have used PyShark for the analysis and classification of data packets captured. PyShark is a Python wrapper for TShark, allowing python packet parsing using Wireshark dissectors. TShark is a network protocol analyzer. It allows capturing packet data from a live network, or reading packets from a previously saved file. TShark's native capture file format is pcap format. The file generated by windump is in the same format.

We have employed PyQt5 for the development of GUI. QtWidgets and QtGui are imported from PyQt5. QtWidgets used include QApplication, and QMainWindow.

Matplotlib and Numpy are used for data visualization. Matplotlib generates bar graph, piechart as well as the table in our tool.

Libraries like subprocess, time, signal, sys, os are imported to write a script which uses command prompt to run windump.

B. Windump

WinDump is the Windows version of tcpdump, the command line network analyzer for UNIX. WinDump is fully compatible with tcpdump and can be used to watch, diagnose and save to disk network traffic according to various complex rules.

WinDump captures using the WinPcap library and drivers. WinDump supports 802.11b/g wireless capture and troubleshooting through the Riverbed AirPcap adapter.

Command “Windump -i 2 -w file.pcap” captures the data and stores them to a file “file.pcap”. if the file exists already, it overwrites that. Else, it creates the file and stores packets data to it.

C. Packet Classification

Our tool uses port numbers of the packets to classify them on the basis of their application layer protocol. A port number is used to identify the specific process that a network message is to be sent to.

When a client message is sent to a server, the port number identifies the specific server program that it is to be forwarded to. The range of port numbers is from 0 to 65536, but the port numbers from 0 to 1023 are reserved for privileged services. For example, port 80 is reserved for HTTP. Almost all application layer protocols use fixed port numbers to exchange data.

PyShark performs the tasks necessary to identify and classify the data which is then visualized.

D. Representation

The captured data packets can be visualized in the following manner:

a) *Table:* The table contains source IP address of packet, destination IP address of packet, source port number of the packet, destination port number of the packet, length of the packet and name of the application layer protocol of the packet.

b) *Bar Graph:* The bar graph depicts the exact count of packets captured. Each bar corresponds to a protocol name along the x-axis and the length of bar depicting the number of packets.

c) *Pie Chart:* Pie chart represents their percentage of each individual protocol in complete capture. Each segment in a unique color shows the percentage and the name of protocol. Legend also provides information pertaining to color and the protocol represented by it

IV. Testing and Evaluation

As we mentioned before, we have displayed results in 3 different formats, namely, a table, a pie chart and a bar graph. All of these were used in the testing of our tool. We captured packets for around 10-15 seconds and the percentage breakdown has been displayed in the pie chart and bar graph as shown below:

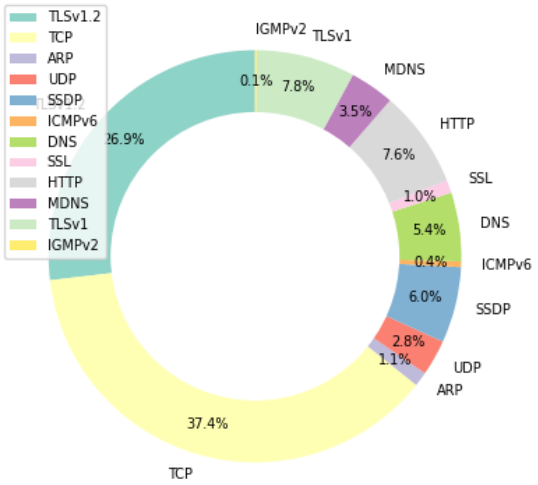

Figure 1. Pie chart representation of Captured Data Packets

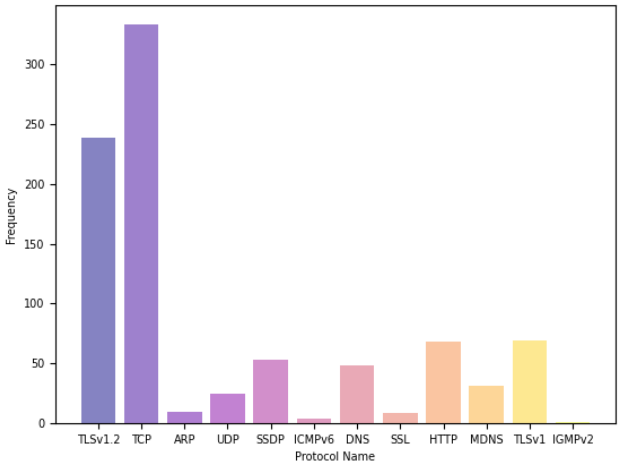

Figure 2. Bar Graph representation of Captured Data Packets

To check that our results were correct, we used the same .pcap file in WireShark and inspected all the packets and protocols identified there. Our tool showed 100% accuracy in identifying the protocols. Following is a screenshot of the table our application created, along with the WireShark packet classification.

Pie Chart | Bar Graph

1	2	3	4
Source	Destination	Length	Protocol
64.233.167.189	192.168.100.15	107	TLSv1.2
192.168.100.15	64.233.167.189	54	TCP
3c:f8:08:4a:3f:1f	d8:fc:93:55:83:63	42	ARP
d8:fc:93:55:83:63	3c:f8:08:4a:3f:1f	42	ARP
192.168.100.15	185.21.216.142	145	UDP
185.21.216.142	192.168.100.15	319	UDP
192.168.100.7	239.255.255.250	167	SSDP
b8:d7:af:79:c6:f1	ff:ff:ff:ff:ff:ff	42	ARP
192.168.100.7	239.255.255.250	167	SSDP
192.168.100.7	239.255.255.250	167	SSDP
192.168.100.8	239.255.255.250	167	SSDP
192.168.100.8	239.255.255.250	167	SSDP
192.168.100.8	239.255.255.250	167	SSDP
fe80::49fe:2f9e:...	ff02::1:ff00:1	86	ICMPv6
fe80::1	fe80::49fe:2f9e:...	86	ICMPv6
fe80::49fe:2f9e:...	fe80::1	102	DNS
192.168.100.15	192.168.100.1	82	DNS

Figure 3. representation of Captured Data Packets in Table

Source	Destination	Protocol	Length
64.233.167.189	192.168.100.15	TLSv1.2	107
192.168.100.15	64.233.167.189	TCP	54
HuaweiTe_4a:3f:1f	IntelCor_55:83:63	ARP	42
IntelCor_55:83:63	HuaweiTe_4a:3f:1f	ARP	42
192.168.100.15	185.21.216.142	UDP	145
185.21.216.142	192.168.100.15	UDP	319
192.168.100.7	239.255.255.250	SSDP	167
MurataMa_79:c6:f1	Broadcast	ARP	42
192.168.100.7	239.255.255.250	SSDP	167
192.168.100.7	239.255.255.250	SSDP	167
192.168.100.8	239.255.255.250	SSDP	167
192.168.100.8	239.255.255.250	SSDP	167
192.168.100.8	239.255.255.250	SSDP	167
fe80::49fe:2f9e:919…	ff02::1:ff00:1	ICMPv6	86
fe80::1	fe80::49fe:2f9e:919…	ICMPv6	86
fe80::49fe:2f9e:919…	fe80::1	DNS	102
192.168.100.15	192.168.100.1	DNS	82
fe80::1	fe80::49fe:2f9e:919…	DNS	146

Figure 4. .pcap file in Wireshark

One small issue that we faced is that when there are a lot of categories of packets, and some of them have very small percentages, the labels of the pie chart tend to overlap and it is difficult to determine the percentage of each. To overcome this, we created a key which depicts the colour of each protocol so that it is easier to identify.

We tested our tool on 4 different PCs, with different time intervals for capturing packets each time and evaluated the results. The protocols were identified correctly each time. However, due to the limited time and resources, we were not able to test it as extensively as we would have liked. There may be some packets which remain unclassified, but overall, the accuracy of the application would remain greater than 95% at least.

V. Conclusion

The report presents the Network Traffic Classification based on Port Numbers identification. Different techniques and advanced classification models have been introduced in the field of networks to classify the real-time, interactive, and bulk traffic, essential for the management of Internet Service Providers (ISPs). These techniques have their benefits and limitations as discussed. We proposed a detailed analysis on the designed methodology for detection of the protocols being used by the packets. This detection was based on their port numbers. The approach used is one of the simplest ways of analyzing packets and proves to be not only effective but efficient as well for many network traffic due to the fixed port numbers associated with various protocols. Accurate traffic classification is possible for well-known port numbers in the range between 1 and 1023 port numbers. However, identifying the packets associated with the applications that do not use well-known port numbers can be challenging in this regard. An example of this are P2P Applications. These do not use well-known port numbers, and if the purpose is to identify P2P applications, then using machine learning to classify protocols would be a better solution.

Following the methodology that we presented, capturing and analyzing live data was done using Pyshark. The developed application uses the input data capture file to classify different packets according to their port numbers and display the results. The evaluation of our tool showed 100% accurate classification of traffic captured. The promising results suggest the reliable packet classification for applications using well known port numbers.

References

[1] M Dashevskiy, Z Luo. Conformal Prediction for Reliable Machine Learning, Morgan Kaufmann Publishers Inc, 2004.

[2] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things", IEEE Access, vol. 5, pp. 18042-18050, 2017.

[3] "Traffic classification", En.wikipedia.org, 2020. Available: https://en.wikipedia.org/wiki/Traffic_classification.

[4] T. Auld, A. W. Moore, S. F. Gull, "Bayesian neural networks for Internet traffic classification", IEEE Trans. Neural Netw., vol. 18, no. 1, pp. 223-239, Jan. 2007

[5] X. Xie, B. Yang, Y. Chen, L. Wang, Z. Chen, "Network traffic classification based on error-correcting output codes and NN ensemble", Proc. 6th Int. Conf. Fuzzy Syst. Knowl. Discovery, pp. 475-479, Aug. 2009

[6] Z. Chen et al., "Improving neural network classification using further division of recognition space", Int. J. Innov. Comput. Inf. Control, vol. 5, no. 2, pp. 301-310, 2009.

[7] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of P2P traffic. In 4th ACM SIGCOMM Internet Measurement Conference (IMC'04), Taormina, IT, October 2004.

[8] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. ACM SIGCOMM Comput. Commun. Rev., 35(4):169– 180, 2005

[9] Marios Iliofotou, Prashanth Pappu, Michalis Faloutsos, Michael Mitzenmacher, Sumeet Singh, and George Varghese. Network monitoring using traffic dispersion graphs (tdgs). In Proc. of IMC '07, San Diego, California, USA, 2007.

[10] Yu Jin, Nick Duffield, Patrick Haffner, Subhabrata Sen, and Zhi-Li Zhang. Inferring applications at the network layer using collective traffic statistics. SIGMETRICS Perform. Eval. Rev., 38, June 2010