A FEATURE TAXONOMY FOR NETWORK TRAFFIC

by

Arshiya Khan

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Cybersecurity

Summer 2019

© Arshiya Khan
All Rights Reserved

A FEATURE TAXONOMY FOR NETWORK TRAFFIC

by

Arshiya Khan

Approved: _____
Chase Cotton, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____
Kenneth E. Barner, Ph.D.
Chair of the Department of Electrical and Computer Engineering

Approved: _____
Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: _____
Douglas J. Doren, Ph.D.
Interim Vice Provost for Graduate and Professional Education and
Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my mentor, philosopher and advisor Dr. Chase Cotton, who has extended his continuous support for my study and research. I thank him for his patience, motivation, supervision, knowledge and enthusiasm. His guidance has been a driving force not only for research, but also beyond that. As a research guide, he was ever assisting and would always improvise research tasks. He provided ample opportunities and illustrious research ideas and allowed me to choose whatever ideas caught my attention. As an advisor, he was always keen on assisting me with the courses that would prove to be beneficial for me. My knowledge of cybersecurity has been broadened significantly over the last two years. I could not have imagined having a better advisor and mentor for my study. Without his supervision, I would have not been where I am today.

I would also like to thank the University of Delaware for giving me the platform and opportunity to accomplish this work.

I extend my vote of thanks to my husband Imran Ahmad who has been my driving force in the two years of my M.S. course. I thank my family and friends who have also been very supportive. Their encouragement has been really valuable.

I also appreciate the valuable inputs from my colleague Ishaani Priyadarshini who is not only a fellow graduate student at the University but is also another researcher of Dr. Cotton.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Even though Artificial Intelligence is still a black box to a lot of people, both experts and non-experts alike, it has become an important tool in current and future technology. Every day, we trust Artificial Intelligence (AI) with our lives, from driving cars to using medical devices. One such important part of life is the Internet which is basically a worldwide exchange of data packets at a very high rate. Security is an integral part of this exchange which encourages enterprises to use Intrusion Detection Systems (IDS)[1,2,3] and Intrusion Prevention Systems (IPS)[4,5] to detect and prevent anomalous activities. As much as we want to use AI to ease our task of anomaly detection, we want to win the trade-off between the true positives, true negatives and false positives, false negatives and ultimately achieve true AI.

Amongst various applications of AI, machine learning (ML) is the most famous. Conceptually, ML is discipline of discovering probabilistic models which use algorithms to learn new things from data like patterns, behaviors, and decision-making capabilities, etc. The higher the accuracy of a model, the better these patterns are learned. These models are developed by training with a large set of data and testing their accuracy on a test dataset. Therefore, we can safely say that the driving engine behind ML is data. If we want ML to make decisions like a human brain, we need to train it on the best possible version of the data we have.

Trusting a probabilistic mechanism to make the right decision might be mathematically acceptable but that is not the case in a dynamic environment like a network where multiple devices like computers, routers, switches and servers, etc. are

communicating and thousands of packets are exchanged every minute and passing through multiple devices. The volume of data seen or collected at a node in a network is enormous, even in a short span of time.

Our goal in this thesis is to collect a dataset of different types of packets that arrive at a node, for long durations of time, in order to facilitate the identification of unusual traffic which might indicate a system error or a possible attack. All the data in a packet does not contribute to identification of the anomaly so we don't use the raw data downloaded directly into a machine learning model. We study the data collected and filter it in a way that useful information is retained, and noise and repetitive data is removed. We also can perform feature engineering[6,7] on the traffic. We compute several derived characteristics of the data by performing several statistical computations like mean and variance on the numeric data and quantitative flow records[8,9] to find derived characteristics which might play an important role in spotting specific behaviors. As a result, we are able to develop an extensive taxonomy of packet data which may be useful in our goal of detecting anomalies.

This taxonomy is a collection of information both taken directly from the packet files as well as derived from the information collected. It defines a tree-like structure which represents all the features of a packet as well as the traffic flow. The branches of the tree are the categories and subcategories and the leaves of the tree represent the final features which can be directly used as estimators of a machine learning model.

Chapter 1

INTRODUCTION

1.1 Cybersecurity

IoT sensors networks are exponentially expanding. Millions of gigabytes of data travel through them every day in the form of data packets. Amongst a lot of things, these sensors include data exchanges from government and other organizations like defense bodies, space research institutes, online money transfer, data from healthcare organizations like Electronic Health Records (EHR) and medical devices like pacemakers, etc. These data packets carry highly sensitive and confidential information which makes it critical to protect these devices from adversary attacks by hackers and terrorist organizations. If delivered into wrong hands, they can have a direct impact on human life. The sensitive nature of these exchanges forces us to administer the highest level of security measures to detect and prevent anomalous activities. The practice of protecting data from malicious attacks is called Computer and Network Security or Cybersecurity.

As the attacks surface is increasing, panic around cybersecurity is also high. To help private sectors deal with malicious activities, the government of the United States has provided several guidelines to approach cybersecurity. The National Institute of Standards and Technology or NIST cybersecurity framework designed in 2014 is one of the most approachable and easy to implement guide. It classifies cybersecurity practices into five phases namely, Identify, Protect, Detect, Respond, and Recover. The below figure from NIST.gov depicts these functions.

Figure 1.1 NIST Cybersecurity Framework

During the "Identify" phase, a clear understanding is developed by an organization by identifying all its IoT and other computer and network assets, the data held by these assets, the processes and procedures followed in the organization for communication between devices, and the capabilities as well as the limitation of the organization with respect to its assets. This will help in setting a clear picture of the organization, specifically portraying needed areas of improvement. The second phase recognized by the NIST framework is "Protect" where an organization needs to ensure that it delivers its services by developing information security schemes and implementing them to safeguard their data and maintain availability. The next phase is called "Detect". Detection of malicious activity should be done quickly, and its implications must be understood. The "Respond" phase follows detection where in case of a cybersecurity event like breach, an organization should be ready with appropriate plans and resources to provide first response, mitigate the effects of the attack and stop the attack, if possible. The organization should perform root cause analysis on the attacked assets to find the source of the attack. The final phase of the framework is called recover in which an organization learns from the cyberattacks and frames security policies and implements protocols to make their systems more resilient

to future similar attacks and breaches. It also tries to restore the damaged assets and lost data from backup.

It is a widely accepted fact that the breaches are inevitable, therefore detection and understanding of a cybersecurity event is the key to resolution and loss prevention. History shows that the protect phase can never be 100% effective, therefore, it is undeniable that we must focus on the "detect" phase of the NIST cybersecurity framework as our final defense. As the name suggest, we will try to detect a cybersecurity event and gather information about it in this phase. The process of detection should answer the following questions:

a. Which cybersecurity event was carried out? The answer that we look for here is which attack was carried out on the organization. This will also help us realize the hidden ultimate motives of the attacker. We can study similar cyberattacks in the history and find the best possible way for save ourselves from it.

b. What information has been compromised in the attack? This is the area where a cyberattack hits the most. The motivation behind the majority of cyberattacks is to hack sensitive data. Therefore, it is important to know how much data has been stolen or even changed as we don't want the latter to compromise the integrity of our databases. This will help us to make decisions if we can save some data and if we have backup.

c. When the cyber-attack happened. The timing of the cyberattack is also an important information. We want to know if the attackers had been in the system for months or years or a few days. Increasingly the attacked organization is informed of the breach by another organization.

d. How the attack happened? It is critical to know how the attack was carried out. With the help of all the information mentioned above, we must find the cause of the breach. The breach could be a result of flaws in the application, network or hardware. If the breach was due to errors in application, we need to change the code and if it is in the network, we need to find the protocol or service where the leak happened. If the breach came from hardware, we might have to replace our current hardware and buy new. Some breaches are a combination of all; application, hardware and network.

e. What are the impacts of the attack? Answer to this question will help us in reconciliation. It will indicate to us how to restore damaged assets. It will also help us get prepared for future. It will point out the gaps in the cybersecurity plan of the enterprise and may illustrate gaps that had not been seen before.

After all these questions are properly answered and accurately recorded, it is the duty of the information security team of the organization to perform root cause analysis and investigate to help the organization prevent any malicious activity in the future. It is also their responsibility to inform the clients.

1.2 Machine Learning

Whenever a cybersecurity event is encountered, we investigate the network traffic that arrived at the device to find where the attack came from, what steps it took, and which service was responsible for the breach. In addition to that, we would also want to collectively analyze the behavior of the flow of data packets and the device's reaction over time. Every single data packet contains an enormous amount of information from which we can find evidence which contributed to the breach in security. Collective study of all the dimensions of this information including the flows

can be scary as it is scattered around in millions of packets. Even after this manual labor, there is still chance of a miss finding the attack or issue. One fairly new, innovative and reliable way to find anomalies in traffic is by using artificial intelligence.

AI is a technology which encourages computers to act like a human brain. The human capabilities which AI can display are playing a strategic game, implementing a decision system like the trolley dilemma, understanding different languages, deducing signals/information from noisy data and remembering behaviors, etc. The ultimate goal of AI is to achieve wisdom. To enable a computer to portray human like intelligence, we implement the AI technology into applications like machine learning.

Given the amount of work done in the field of machine learning, the proportion of people having a clear understanding of it is very low. Machine learning is like a black box to a many people. It is a secret sauce which takes in numbers and gives hopefully accurate results. It is a sauce but it's not secret. A lot of implementations of AI require very specific coding to perform computations explicitly for an application, for example robotic arms or industrial robots found in warehouses. Machine learning is an idea which states that we don't need a lot of custom coding for an algorithm to learn interesting facts about our data, instead we can just provide a lot of data and the algorithm recognizes patterns and displays facts. In most cases, it gives accurate results which is the ultimately the goal. By using machine learning, we can solve problems like true or false classification, image recognition, value prediction like stock prices and customer churn data, future cost of houses, etc. Other applications are clustering unknown data into groups with similar behavior, audio analysis, text

generation and reinforcement learning. More complex applications of machine learning are generative adversarial networks, deep learning models, etc.

As mentioned above, we can use a generic algorithm, just feed it a large amount of data and our machine learning model is ready to use. So, it is fair to say that dataset plays a decisive role in a machine learning model to reach semantically correct decisions with the help of these algorithms. Machine learning will only work if the dataset we have matches the needs of the problems that it should solve. It means that the dataset should be a true representative of the task a model is supposed to do. We can measure the quality of the dataset by the features present in it. These features are fine identifiers of anomalous and non-anomalous situation. Later, when the model is trained to predict classes, these will be computed into the estimators of the model.

Chapter 2

RELATED WORK

A variety of research work has been conducted in this field of anomaly detection in a network using machine learning[10,11]. Some researches use features to combat adversarial machine learning[12,13] while other use features to create anomaly detection models. But there has been limited work on actually identifying a comprehensive set of features to help AI reach accurate solutions.

Limthong[30] has shown how machine learning approaches can be used to detect anomaly by experimenting on a dataset with malicious data in them collected over a period of around 8 weeks. Their dataset contained back, ipsweep, neptune, portsweep and smurf attacks with a maximum anomaly detection percentage of 1.25, 5.3, 7.38, 1.19, and 2.22 respectively. The features used were number of packets, sum of packet size, number of flows, number of source addresses, number of destination addresses, number of source ports, number of destination ports, difference between source and destination addresses, and difference between source and destination ports. Several machine learning algorithms were used like Naïve Bayes algorithm[14,15] and K-nearest neighbor[16,17]. Although, the accuracy was good, but the features taken into consideration were limited.

De Lucia[37] discusses the importance of features in adversarial learning in the context of cybersecurity. The study of features was divided into network packet features and network flow/traffic features. A need of derived features was also expressed to be computed using statistical methods. Data mining approach like

TF/IDF[18,19,20,21,22,23] was used to study the frequency of TLS record sizes[24,25] in a conversation. This study presents good discussion and direction to capture complete set of features for machine learning.

Muehlstein[38] performed classification of HTTP[26] encrypted traffic (HTTPS[27]) using machine learning classifiers to identify user's operating system, browser and application by exploiting passive attack techniques such as sniffing on traffic. Their dataset contains around 2000 labeled online sessions, with Ubuntu, Windows, Linux and OSX operating systems. The users browsed using common applications like Internet Explorer, Safari, Chrome and Firefox. The sites that were used for classification were Facebook, twitter and YouTube. All the traffic that passed through port 443 (for SSL) through was collected. They used support vector machines (SVMs)[28] to perform classification and achieved more than 90% accuracy in all cases. Some base features included number/mean of forward/backward packets, mean packet size/bytes, etc. Some new features were proposed like TCP initial window size, SSL[29] information, peak throughputs, peak interval time differences, the number of keep alive packets. The features collected were meaningful to this experiment but are not able to represent the collective nature of a packet and network flow.

Chapter 3

FEATURES OF TRAFFIC DATA

3.1 Network Traffic Focus

An anomaly in a network traffic is a disruption of the intended behavior of a service. This disruption can act as a loophole to be exploited by cyber attackers and the whole system can be compromised. Despite efforts made to secure all communication channels and devices, mischievous actors find a way to harm the system. The primary contribution of my thesis is to identify characteristics of network traffic that help us differentiate between anomalous behavior from normal.

3.2 IPv4 Packet

Data is sent from source to destination by dividing them into small units of packets (i.e. datagrams). The majority of traffic on modern networks is carried in IPv4 datagrams with sizes ranging from 20 to 65,535 bytes though realistic packet sizes are usually limited by the maximum packet size carried by the underlying network (e.g. Ethernet). An IPv4 packet is divided into header and payload. The payload, as the name suggests, contains the message that needs to be delivered at the destination. The header, the most important part of a packet contains 14 fields and occupies 20 bytes of data which can be extended, if needed. It contains all the information that is required by a packet to reach its destination node. The figure below from Cisco[35] depicts the fields in the packet header and their distribution across the header.

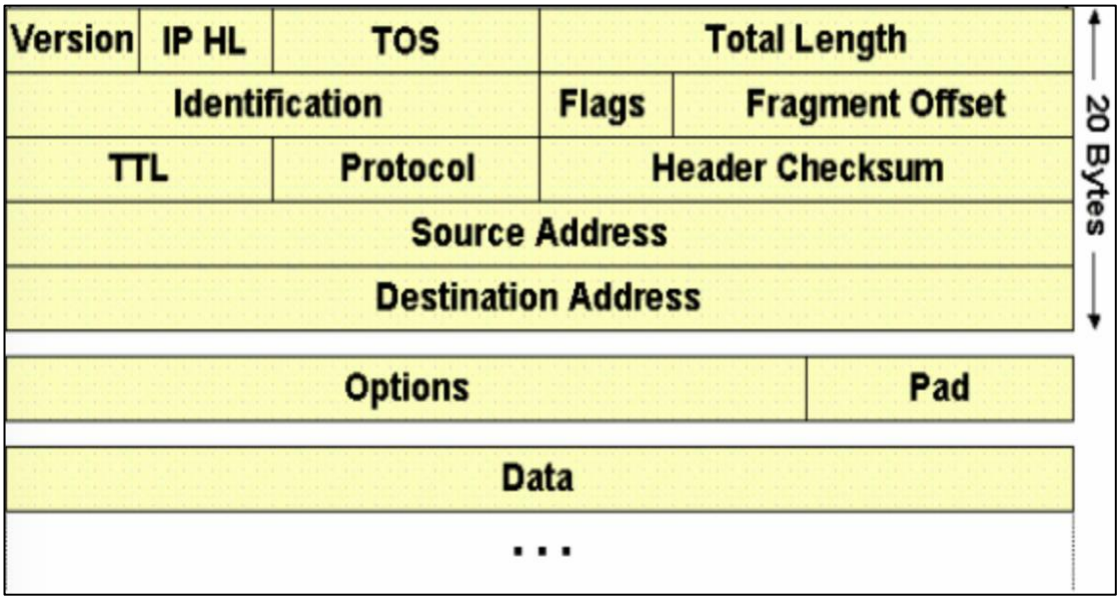

Figure 3.1 IPv4 Packet

3.3 Feature Identification

There have been several discussions of network features that can be used as training data in machine learning. Prior use restricted these features to very limited set of data has been selected from the traffic to differentiate behaviors like source, destination and ports. Although, the results of such studies have been precise, unfortunately there is a chance of overfitting as the decision of declaring a packet that contains an anomaly seems to be incomplete with such limited data.

In order to reach meaningful results, we need to identify all the features that might play a role in disrupting a packet or traffic overall. To achieve this, we need to look beyond just header values. We need to consider the traffic movements and header values at those instances. As tedious as this task sounds, it is necessary that we use it to achieve true AI.

3.4 Tools Used

The tools that we used to achieve all measurement of traffic are Wireshark/tshark and Zeek (previously known as Bro).

3.4.1 Wireshark

Wireshark is a GUI tool used to sniff traffic over an Internet connection. It has a companion command-line tool is called tshark. We use Wireshark to watch all the traffic passing through the node. We can even download the packets seen by Wireshark into a file. The files which hold the packets are called pcap (or pcapng) format files. The tshark tool is a command-line version of Wireshark. By using tshark on any operating system, we can download all the packet information into a pcap file. We can filter out this file to remove unnecessary and repetitive information and use the results as a training dataset.

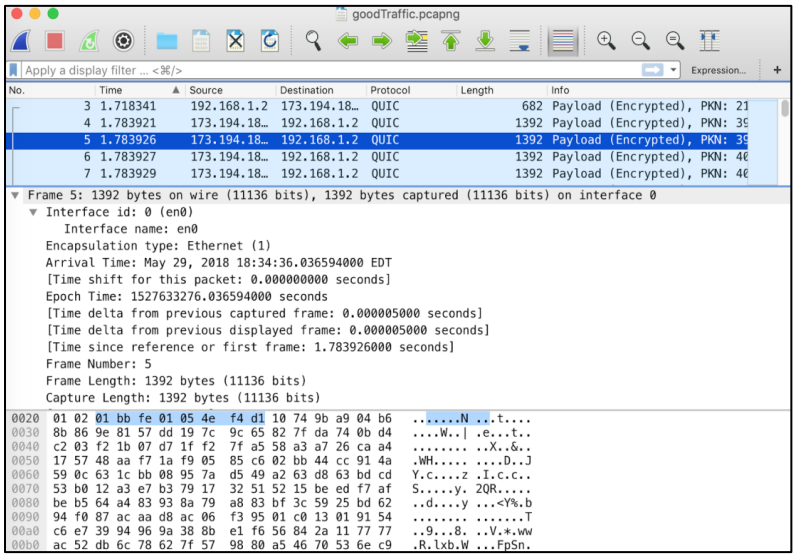

Figure 3.2 Wireshark GUI

3.4.2 Zeek

Zeek (aka Bro) is a network analysis framework which is used as an Intrusion Detection System (IDS). Zeek is an open source architecture which is driven by cybersecurity events and handles security exceptions to stop malicious activity. It monitors traffic over the whole network from a central or multiple point and stores its results in multiple log files at the host. These files can be examined to examine and find a variety of information from logging of connections to signature mismatches, and other exceptions. These files are logged on an hourly basis. This information can be useful to detect behaviors over a network at different instances of time. Changes in the normal behavior can be identified as an anomaly. Some types of files produced by Zeek are:

3.4.2.1 stats.log

Zeek records all the connections This file contains the obvious statistics related the connections of the node. This file contains

3.4.2.2 conn.log

This file logs all the connection information that the node has seen on the network. The connection.log is created on an hourly basis. All the connection from the previous hour are recorded here. It contains the IP, TCP, UDP and ICMP connections. Some of the fields found in this log are: source and destination IP addresses and ports, duration of the connections, protocols used, number of RESP packets, and GeoIP country codes of the packets.

3.4.2.3 weird.log

The weird.log shows the records the events which are not understood by Zeek's protocol analyzers. This information which contains some valid and/or invalid protocols or unexpected events which Zeek is not able parse are placed into this file.

3.4.2.4 signature.log

The signature.log file contains all the signature matches from the signature framework.

3.4.2.5 stderr.log

The standard errors found in the packets when Zeek is started are logged into this file.

3.4.2.6 ssh.log

This file contains information about the SSH handshakes that took place over the network. It has the following fields: ID record of source, destination and ports; status of the login attempt, client and server responses.

3.5 Expansive Feature Taxonomy

Our goal here is to study all the characteristics of a system. To score good analysis results, we need to find all the dependencies and internal and external environments that influence a system. In this development of an expansive taxonomy, we will classify our data based on the source where it came from and what it signifies. We have achieved this by developing a taxonomy tree whose sub-roots are the main categories into which the features have been divided. The features at the leaves of this tree can be used into a machine learning model as estimators of the model.

We have divided the network traffic into the following categories:

a. **Packet field:** These features are directly obtained from the packets. They provide information about the packet and the source and destination of the packets.

b. **Traffic volume:** Volume based features represent the quantitative measurements of the size (e.g. packet length, average packet length, etc.) in a of different groupings (sets) of packets, e.g. a flow, in the channel, etc..

c. **Protocol state:** State based features represent behavior of the packet.

d. **Time series**: Time based features depict the time dependent characteristics of the packets.

e. **Traffic flow:** Characteristics of the data stream are included in this category.

f. **Computed features:** Computed features are not directly observed from the packet stream. They are calculated from the recorded from the packet/stream to determine their behavior and recognize any patterns.

g. **Other:** All other miscellaneous information gathered from either packet or stream which does not fit in the above categories is included here.

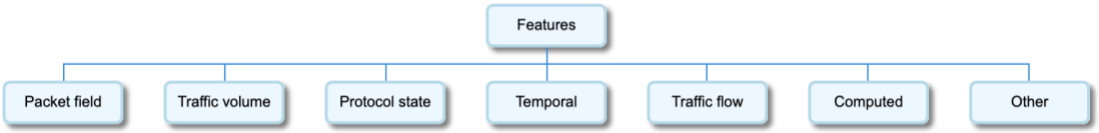

Figure 3.3 Feature Taxonomy

3.5.1 Packet field

These features are directly collected from the defined fields found in any standard communications devices for protocols defined at any layer of the communications stack. Both header fields as well as data fields are included in our

definition. This set of define fields form an almost unlimited set of features to use in traffic classification, clustering, anomaly detection, and any other potential use of statistical or neural machine learning system. The header fields will largely indicate the state activity in the protocol and the data fields may help understand the content of the data being transferred by the protocol.

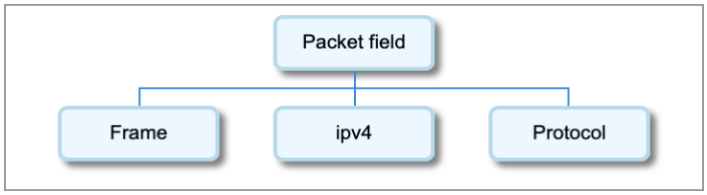

Figure 3.4 Packet Field Branch

Observation of these features is the easiest way of detecting anomaly. We have used Wireshark to define the features for this category as the Wireshark community has invested a significant amount of time defining the fields of commonly seen network traffic.

A lot of important information is retrieved from this field therefore, we try to retain maximum information in this category, except for a few redundant and trivial fields. The packet field-based features are further classified into three categories named after the section of the header they come from:

3.5.1.1 Frame

The frame section of the protocol contains the summary of the packet. It is a collection of metadata about the packet. Therefore, we observe important characteristics about the packet from this section. The notation of frame in Wireshark is “frame”. We are going to directly use this notation to describe our features.

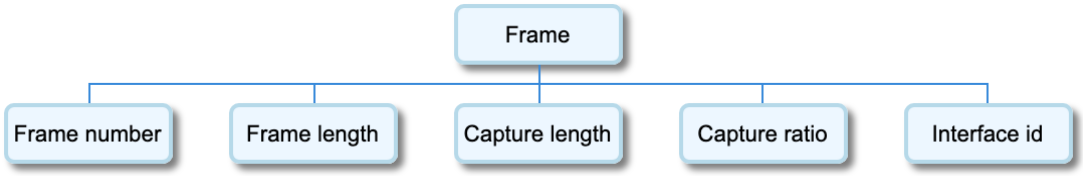

Figure 3.5 Frames features

The features that are important to be noted are:

a. Frame number: It is a record of the sequence in which the frame was downloaded in the file. It is vital that the frame numbers remain in sequence. A missing frame might be a potential sign of malicious activity.

b. Frame length: Length of the frame as seen on the wire. It is recorded in bytes

c. Capture length: Length of the frame captured in the file. It is recorded in bytes.

d. Capture ratio: It is imperative that all the packets seen on the wire be captured in Wireshark for analysis. This means that the ideal value of ratio of capture length over frame length should be 1. If any other value is observed, it should be reported.

Let’s see how these features look like in Wireshark.

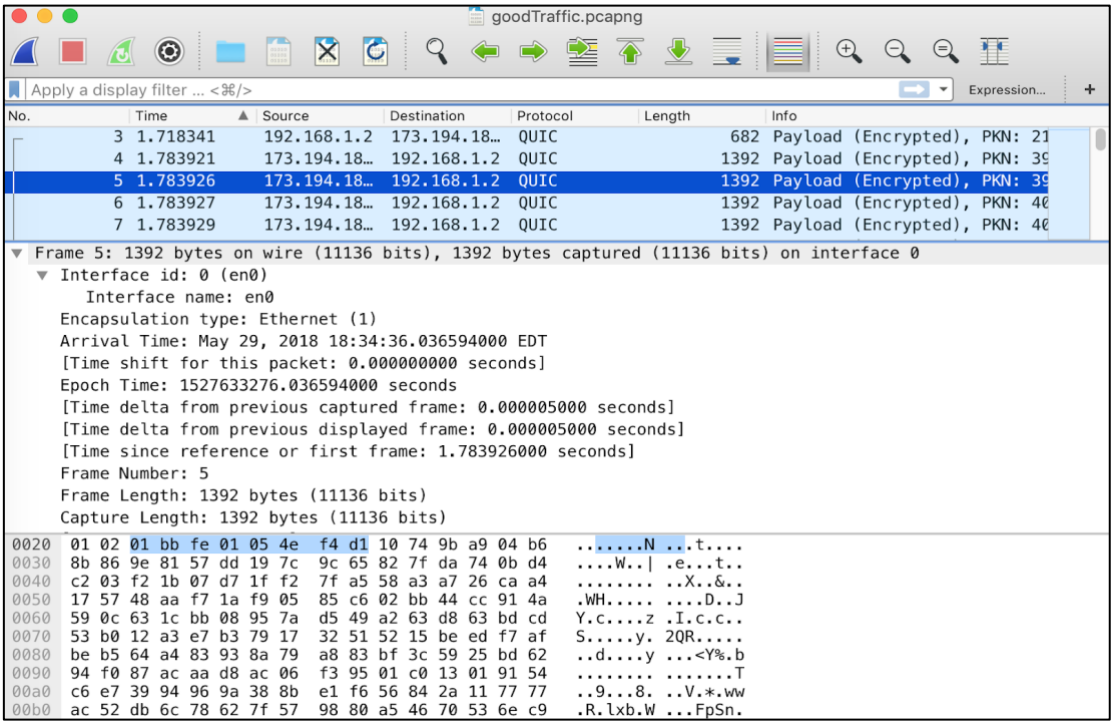

Figure 3.6 Wireshark packet frame

e. Interface id: ID of the interface on which the packet was seen. Interface is the network connection on which the data exchange happens physically. It is being used to capture the packets. It can be used for either wired or wireless connections. Examples of interfaces are en0, en1, lo1 (loopback interface), and fw0 (firewall interface IP), vmnet0 (for virtual machines installed), eth0 (non OSX devices), etc.

The table below is a useful explanation of the nature of the features captured in this section following the Wireshark notation.

Taxonomy	Notation	Type	Size
Frame number	frame.number	Unsigned Integer	4 bytes
Frame length	frame.pckt_len	Unsigned Integer	4 bytes

Capture length	frame.cap_len	Unsigned Integer	4 bytes
Interface id	frame.interface_id	Unsigned Integer	4 bytes

Table 3.1 Feature taxonomy of a frame

3.5.1.2 IPv4

All the features which come from the IPv4 header of the packet are recorded in this section. The basic and the most important information among all the field-based features come from this section. Features like protocol, source, destination, TTL, etc. are observed in this field and a lot of direct anomalies can be deduced from these set of fields. The field name notation for IPv4 section in Wireshark is “ip”. Therefore, the notations for fields inside this section start with “ip”.

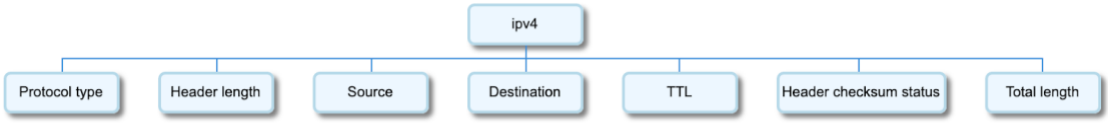

Figure 3.7 IPv4 features

Fields in the above figure shape an IP packet and are defined below:

a. Protocol type: The protocol used in this conversation to exchange information is recorded in the protocol type field. Protocols like TCP, UDP, ICMP can be found in this field.
b. Header length: This is the length of the IPv4 header.
c. Source IP address: This field contains the IP address of the source machine.
d. Destination IP address: This field contains the IP address of the destination machine.

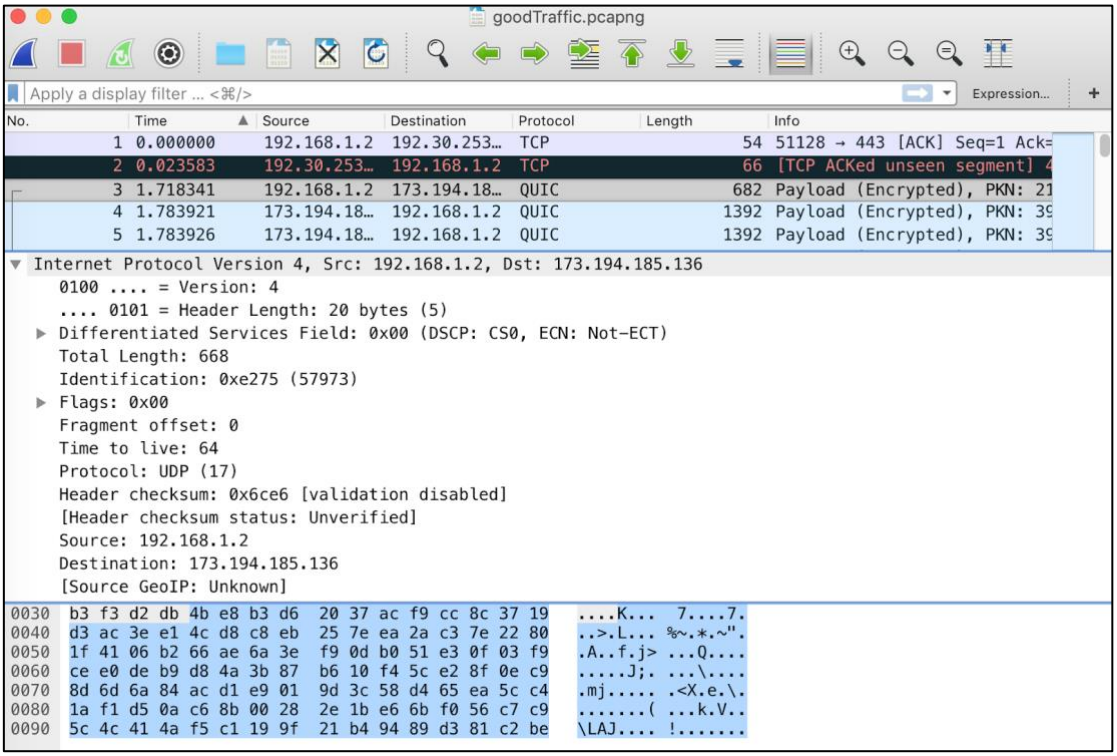

Figure 3.8 IPv4 packet

e. TTL: Time-to-live is a very important feature of a packet. It is the maximum time an IP packet has to reach its destination hopping from one network device to another. Once a packet has reached TTL, it is discarded on the hop it has arrived on, assuming it to be lost. Every time a packet hops to a new device in the network, TTL is reduced by a few units equal to or more than 1.

f. Header checksum status: This field is a description of whether the checksum of the header is verified or not.

g. Total length: It shows the length of the IP header. We can extend this length if need.

Feature name	Taxonomy	Type	Size
Protocol type	ip.proto	Unsigned Integer	1 byte

Header length	ip.hdr_len	Unsigned Integer	1 byte
Source IP address	ip.src	IPv4 address	-
Destination IP address	ip.dst	IPv4 address	-
TTL	ip.ttl	Unsigned Integer	1 byte
Header checksum status	ip.checksum.status	Unsigned Integer	1 byte
Total length	ip.len	Unsigned Integer	2 bytes

Table 3.2 Feature taxonomy of an IPv4 packet

3.5.1.3 Protocol

This section of the packet field-based features contains the protocol that is used to exchange information between the source and destination using the packets. This can be TCP, UDP, SMB etc. For notation in this section, we have recorded the most important fields critical to anomaly detection in any protocol are:

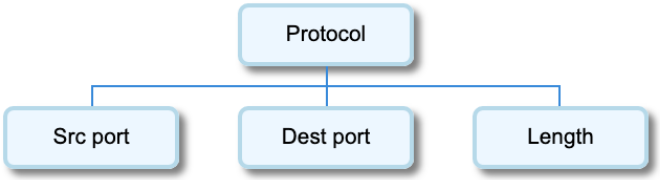

Figure 3.9 Basic protocol features

a. Source port: The port number of the sender's device from where the information is sent on the wire.

b. Destination port: The port number of the destination's

c. Length: This is the length of the protocol header.

Feature name	Taxonomy	Type	Size
Source port	protocol.srcport	Unsigned Integer	2 bytes
Destination port	protocol.dstport	Unsigned Integer	2 bytes
Length	protocol.length	Unsigned Integer	2 bytes

Table 3.3 Feature taxonomy of a packet

Protocols can be of different types in this context. Some of the common protocols found in this section are:

3.5.1.3.1 UDP

User Datagram Protocol(UDP) sends data in an insecure way. It's not the job of UDP to make sure that data arrived at the destination which makes it easier to attack.

Feature name	Taxonomy	Type	Size
Source port	udp.srcport	Unsigned Integer	2 bytes
Destination port	udp.dstport	Unsigned Integer	2 bytes
Length	udp.length	Unsigned Integer	2 bytes
Time since previous frame	udp.time_delta	Time offset	-
Time since first frame	udp.time_relative	Time offset	-
udp.time_delta/udp.time_relative	-	Unsigned Integer	-

Table 3.4 Feature taxonomy of a UDP packet

udp.time_delta/udp.time_relative: Another famous attack displayed in the UDP protocol is DoS flood attack. To detect the attack, we take the ratio of the time features in the above table. udp.time_delta gives the time from the previous frame and udp.time_relative gives the time from the first frame. The ratio will give the speed at which the packets are arriving at the node. If the rate is very high, it can be an indicator of UDP Dos attack.

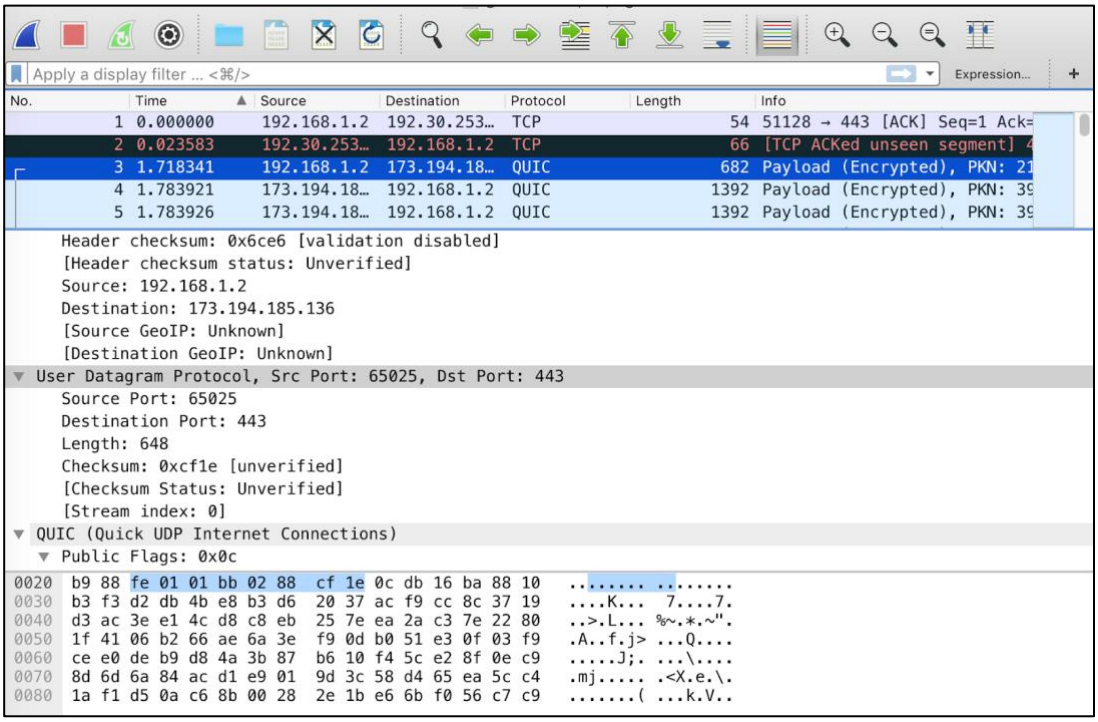

Figure 3.10 UDP features

3.5.1.3.2 SMB

Feature name	Taxonomy	Type	Size
Server Component	smb.server_component	Unsigned Integer	4 bytes
SMB Command	smb.cmd	Unsigned Integer	1 byte
NT Status	smb.nt_status	Unsigned Integer	4 bytes

Flags2	smb.flags2	Unsigned Integer	2 bytes
Security Mode	smb.sm	Unsigned Integer	2 bytes
Word Count (WCT)	smb.wct	Unsigned Integer	1 byte
Byte Count (BCC)	smb.bcc	Unsigned Integer	2 bytes

Table 3.5 Feature taxonomy of a SMB protocol

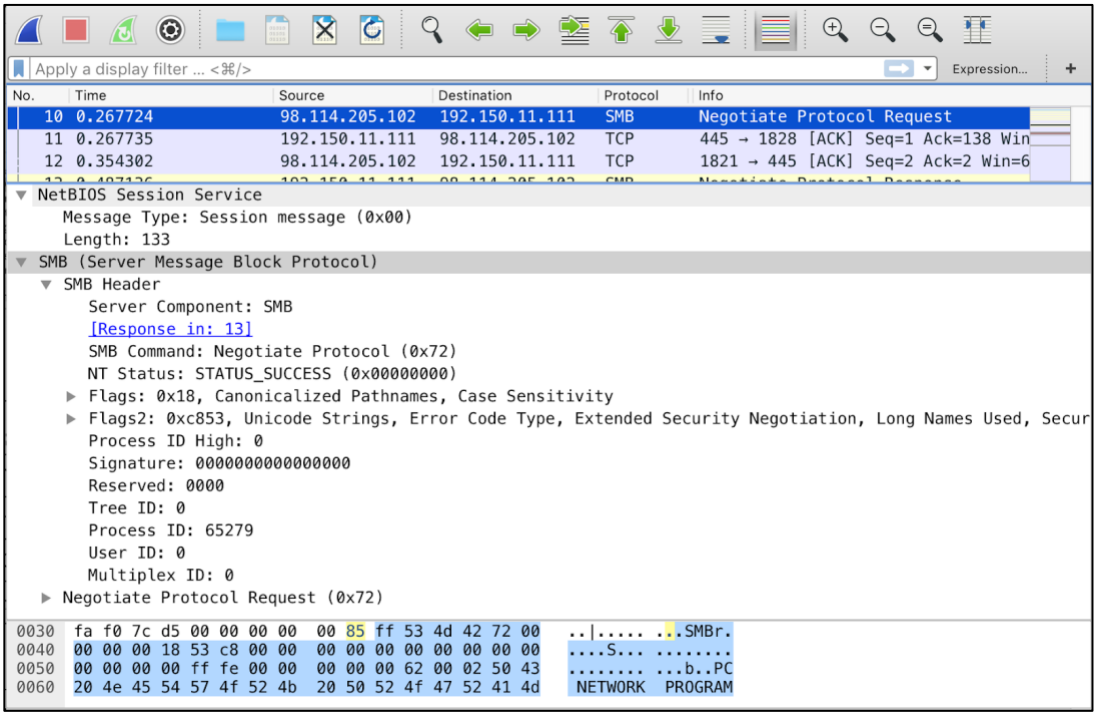

Figure 3.11 SMB features

3.5.1.3.3 TCP

Feature name	Taxonomy	Type	Size
Source port	tcp.srcport	Unsigned Integer	2 bytes
Destination port	tcp.dstport	Unsigned Integer	2 bytes
Header Length	tcp.hdr_len	Unsigned Integer	1 byte

TCP Segment Len	tcp.len	Unsigned Integer	4 bytes
Checksum Status	tcp.checksum.status	Unsigned Integer	1 byte
MSS Value	tcp.options.mss_val	Unsigned Integer	2 bytes
Time since previous frame	tcp.time_delta	Time offset	-
Time since first frame	tcp.time_relative	Time offset	-
Calculated window size	tcp.window_size	Unsigned Integer	4 bytes
Window size value	tcp.window_size_value	Unsigned Integer	2 bytes
ACKed segment that wasn't captured	tcp.analysis.ack_lost_segment	Label	-
ACK to a TCP keep-alive segment	tcp.analysis.keep_alive_ack	Label	-

Table 3.6 Feature taxonomy of TCP protocol

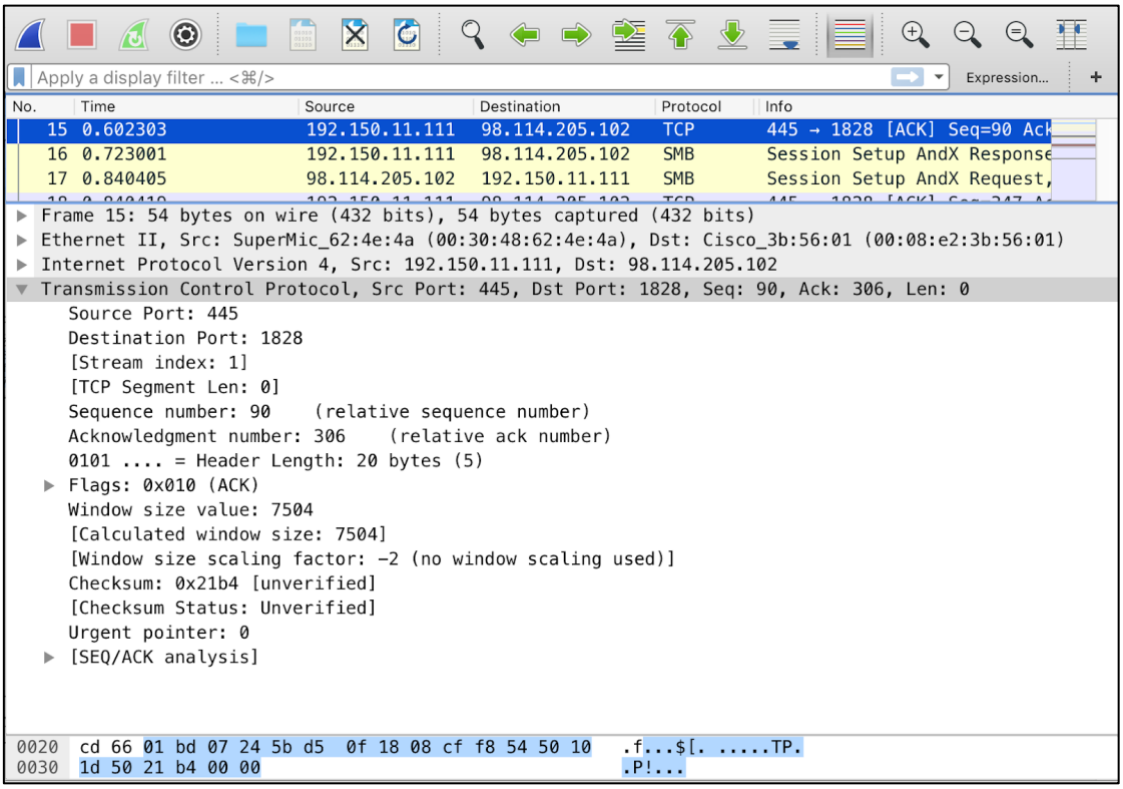

Figure 3.12 TCP features

3.5.1.3.4 SSL

SSL or Secure Socket Layer is used to encrypt the messages using some common encryption techniques like Elliptical Curve Cryptography.

Feature name	Taxonomy	Type	Size
Handshake Protocol	ssl.handshake	Label	-
Alert Message	ssl.alert_message	Label	-
Content Type	ssl.record.content_type	Unsigned Integer	1 byte
Length	ssl.record.length	Unsigned Integer	2 bytes
Session ID Length	ssl.handshake.session_id_length	Unsigned	2 bytes

		Integer	
Cipher Suites Length	ssl.handshake.cipher_suites_length	Unsigned Integer	2 bytes
Compression Methods	ssl.handshake.comp_method	Unsigned Integer	1 byte
Public key length	ssl.handshake.server_point_len	Unsigned Integer	1 byte

Table 3.7 Feature taxonomy of SSL protocol

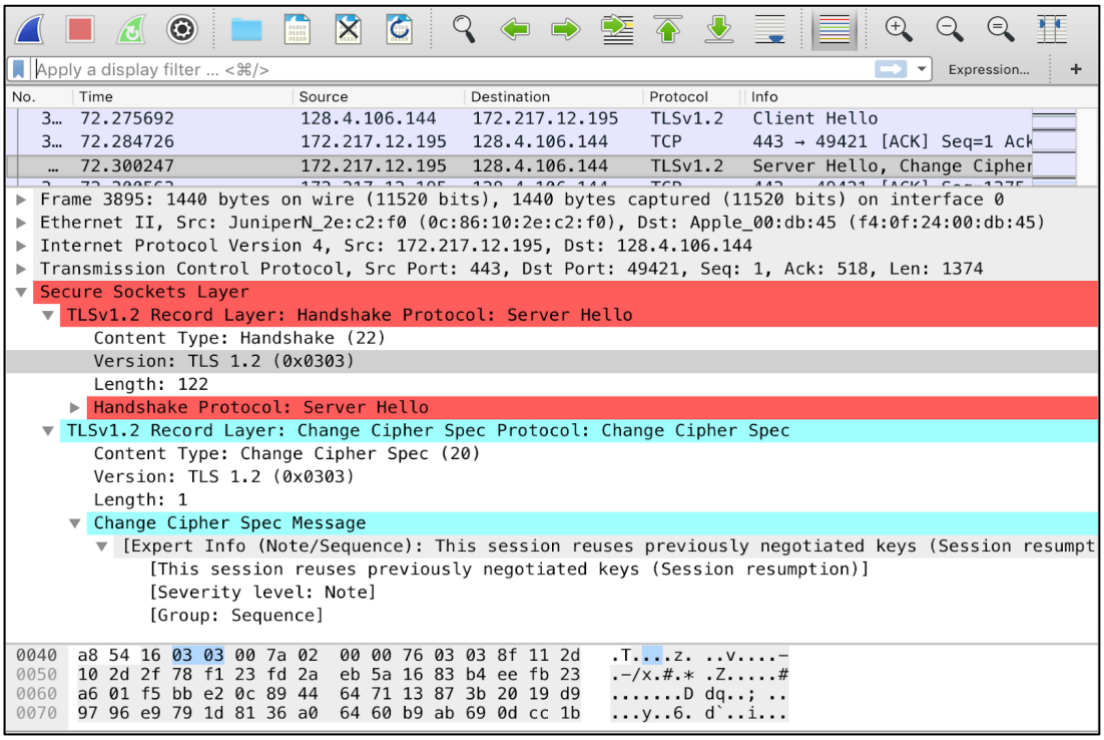

Figure 3.13 SSL features

3.5.1.3.5 DNS

DNS stands for Domain Name Server. This protocol hunts for IP addresses for the given domain names when using a web page. DNS protocol has request and

response queries where a node sends a DNS request to the root domain servers and response contains the IP address of the webserver.

Feature name	Taxonomy	Type	Size
Transaction ID	dns.id	Unsigned Integer	2 bytes
Questions	dns.count.queries	Unsigned Integer	2 bytes
Answer RRs	dns.count.answers	Unsigned Integer	2 bytes
Authority RRs	dns.count.auth_rr	Unsigned Integer	2 bytes
Name length	dns.qry.name.len	Unsigned Integer	2 bytes
Name	dns.qry.name	Character string	-
Label Count	dns.count.labels	Unsigned Integer	2 bytes
Type	dns.qry.type	Unsigned Integer	2 bytes
Class	dns.qry.class	Unsigned Integer	2 bytes
Time	dns.time	Time Offset	-
Answer authenticated	dns.flags.authenticated	Boolean	-
Type	dns.resp.type	Unsigned Integer	2 bytes
Class	dns.resp.class	Unsigned Integer	2 bytes
Time to live	dns.resp.ttl	Signed Integer	4 bytes
Data length	dns.resp.len	Unsigned Integer	4 bytes
Primary name server	dns.soa.mname	Character string	-
Refresh Interval	dns.soa.refresh_interval	Unsigned Integer	4 bytes
Retry Interval	dns.soa.retry_interval	Unsigned Integer	4 bytes
Expire limit	dns.soa.expire_limit	Unsigned Integer	4 bytes

Minimum TTL	dns.soa.mininum_ttl	Unsigned Integer	4 bytes

Table 3.8 Feature taxonomy of DNS protocol

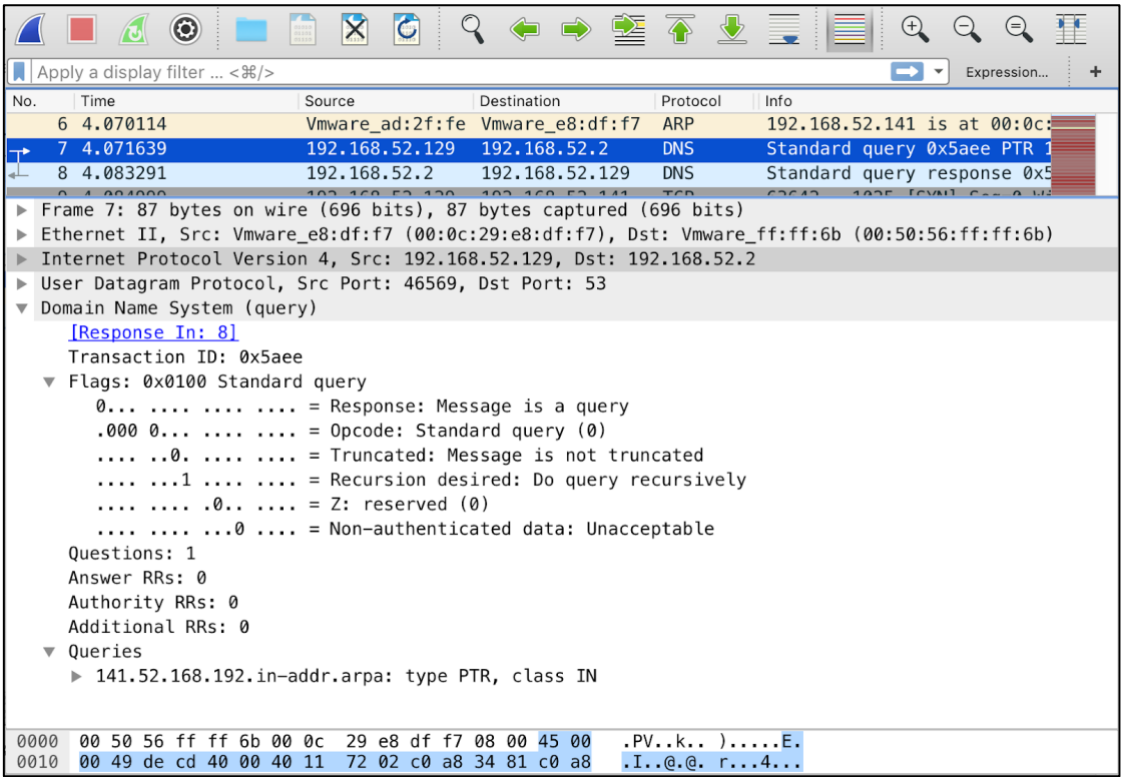

Figure 3.14 DNS request features

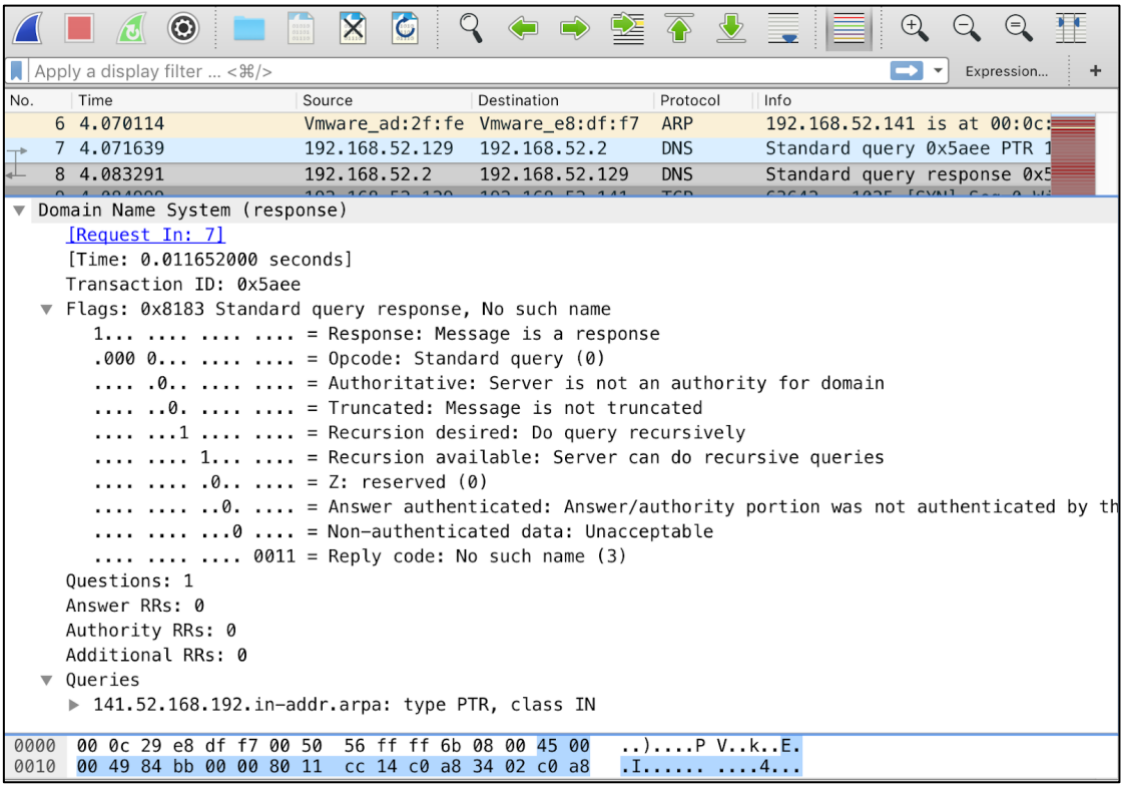

Figure 3.15 DNS response features

3.5.1.3.6 ICMP/ICMPv6

Feature name	Taxonomy	Type	Size
Type	icmp(v6).type	Unsigned Integer	1 byte
Code	icmp(v6).code	Unsigned Integer	1 byte
Checksum	icmp(v6).checksum	Unsigned Integer	2 bytes
Checksum Status	icmp(v6).checksum.status	Unsigned Integer	1 byte
ICMPv6			
Number of Multicast Address Records	icmpv6.mldr.nb_mcast_records	Unsigned Integer	2 bytes

Record Type	icmpv6.mldr.mar.record_type	Unsigned Integer	1 byte
Aux Data Length	icmpv6.mldr.mar.aux_data_len	Unsigned Integer	1 byte
Number of sources	icmpv6.mldr.mar.nb_sources	Unsigned Integer	2 bytes

Table 3.9 Feature taxonomy of ICMP protocol

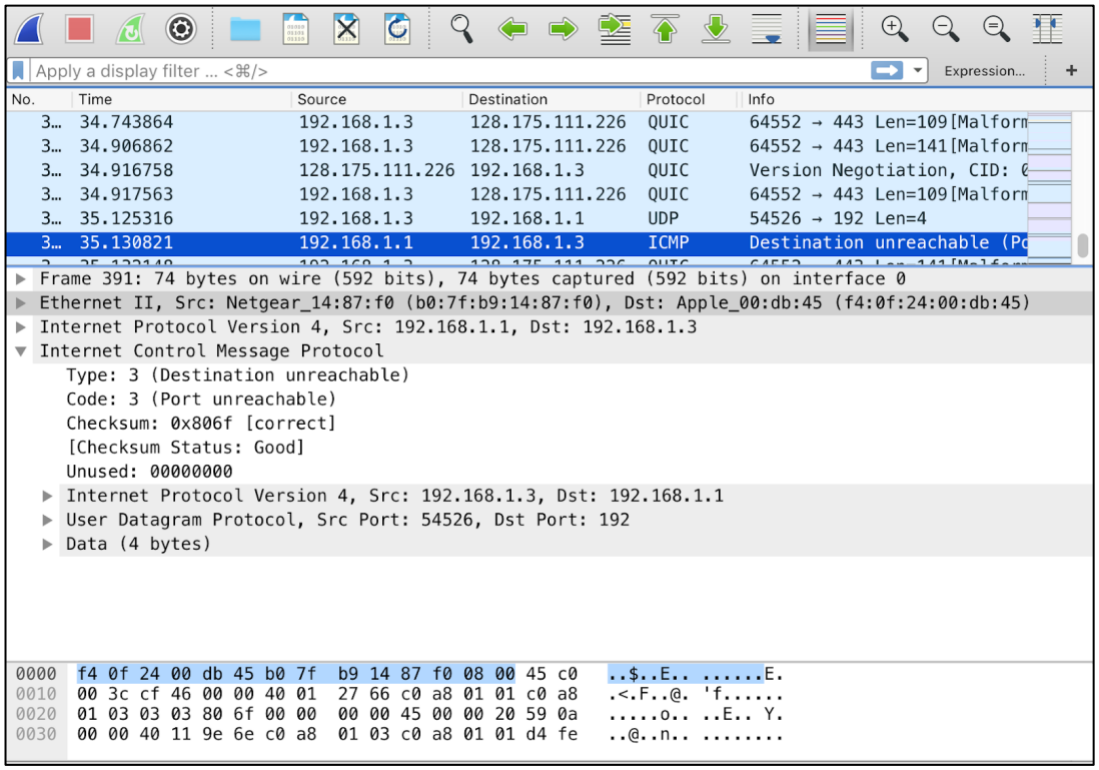

Figure 3.16 ICMP request features

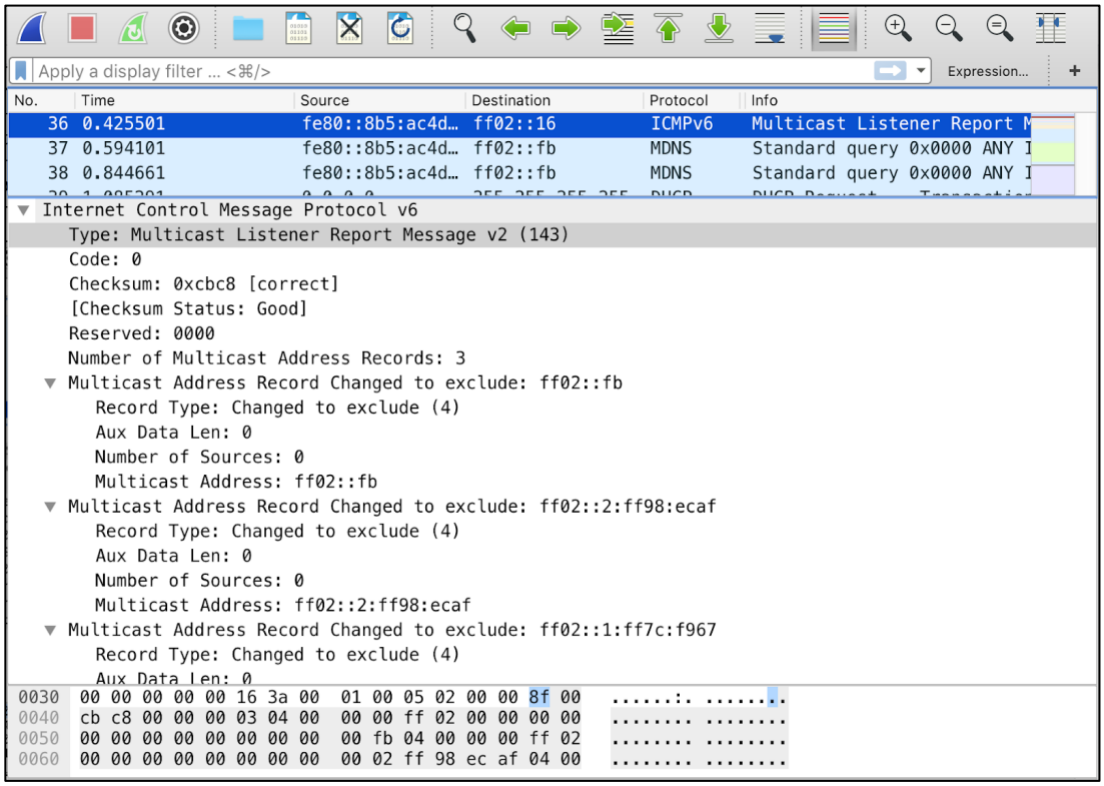

Figure 3.17 ICMP response features

3.5.1.3.7 ARP

Feature name	Taxonomy	Type	Size
Hardware Type	arp.hw.type	Unsigned Integer	2 bytes
Protocol Type	arp.proto.type	Unsigned Integer	2 bytes
Hardware Size	arp.hw.size	Unsigned Integer	1 byte
Opcode	arp.opcode	Unsigned Integer	2 bytes
Sender IP address	arp.src.proto_ipv4	IPv4 address	-
Target IP address	arp.dst.proto_ipv4	IPv4 address	-
Is gratuitous	arp.isgratuitous	Boolean	-

Table 3.10 Feature taxonomy of ARP protocol

Table 3.10 Feature taxonomy of ARP protocol

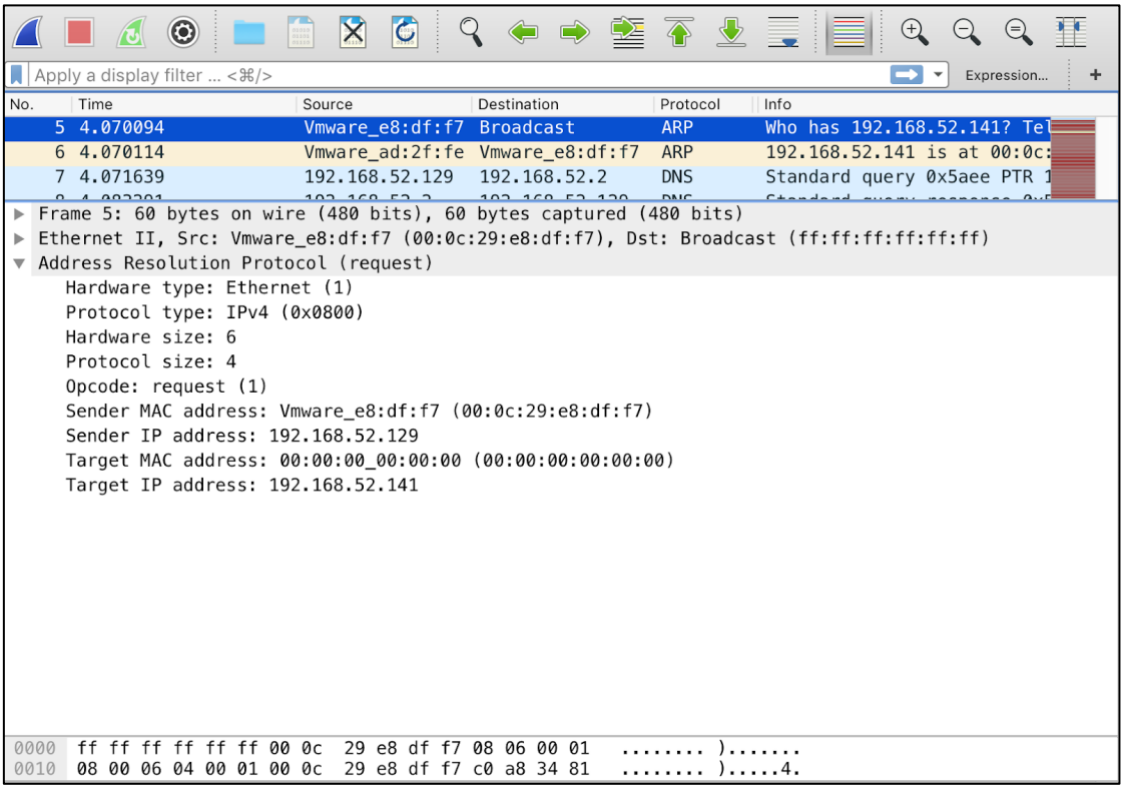

Figure 3.18 ARP request features

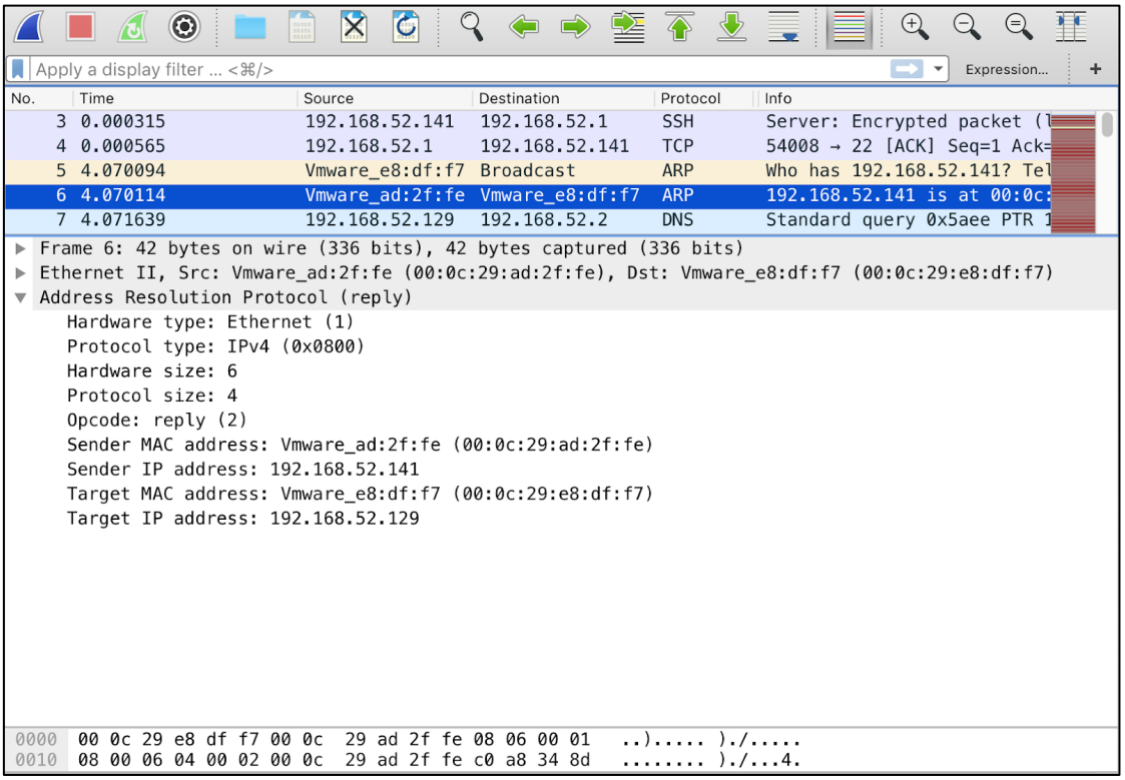

Figure 3.19 ARP reply features

3.5.1.3.8 SSH

Feature name	Taxonomy	Type	Size
Packet Length	ssh.packet_length	Unsigned Integer	4 bytes
Padding Length	ssh.padding_length	Unsigned Integer	1 byte
Message Code	ssh.message_code	Unsigned Integer	1 byte
Kex Algorithm Length	ssh.kex_algorithms_length	Unsigned Integer	4 bytes
Kex_algorithms string	ssh.kex_algorithms	Character String	-

First KEX Packet Follows	ssh.first_kex_packet_follows	Unsigned Integer	1 byte
Server_host_key_algorithms length	ssh.server_host_key_algorithms_length	Unsigned Integer	4 bytes
Server_host_key_algorithms string	ssh.server_host_key_algorithms	Character String	-
Mac_algorithms_client_to_server length	ssh.mac_algorithms_client_to_server_length	Unsigned Integer	4 bytes
Mac_algorithms_client_to_server string	ssh.mac_algorithms_client_to_server	Character String	-
Compression_algorithms_client_to_server length	ssh.compression_algorithms_client_to_server_length	Unsigned Integer	4 bytes
Compression_algorithms_server_to_client string	ssh.compression_algorithms_server_to_client	Character String	-
SSH Protocol	ssh.protocol	Character String	-

Table 3.11 Feature taxonomy of SSH protocol

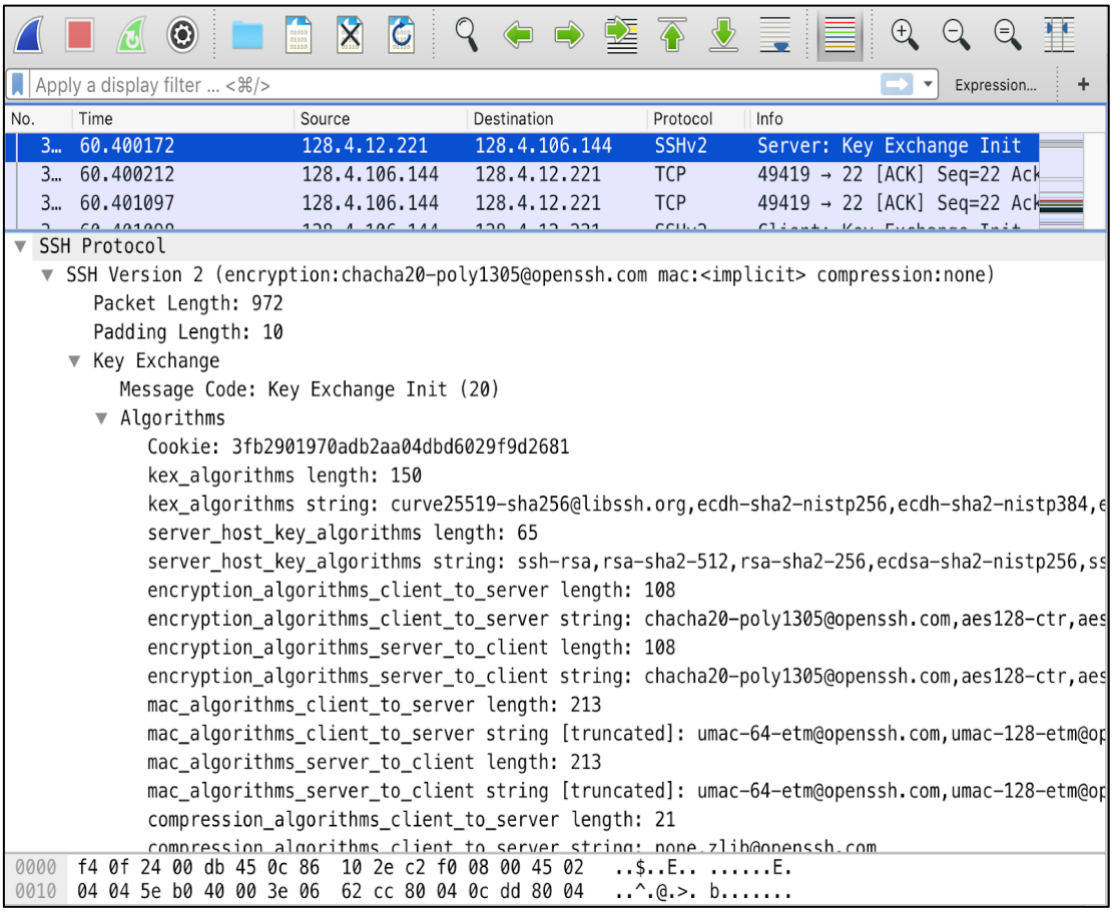

Figure 3.20 SSL features

3.5.1.3.9 HTTP:

Feature name	Taxonomy	Type	Size
Connection	http.connection	Character String	-
Accept Encoding	http.accept_encoding	Character String	-
Request Method	http.request.method	Character String	-
Request URI	http.request.uri	Character String	-
Status Code	http.response.code	Unsigned Integer	2 bytes
Status Code Description	http.response.code.desc	Character String	-
Content-Type	http.content_type	Character String	-

Server	http.server	Character String	-
Time since request	http.time	Time offset	-

Table 3.12 Feature taxonomy of HTTP protocol

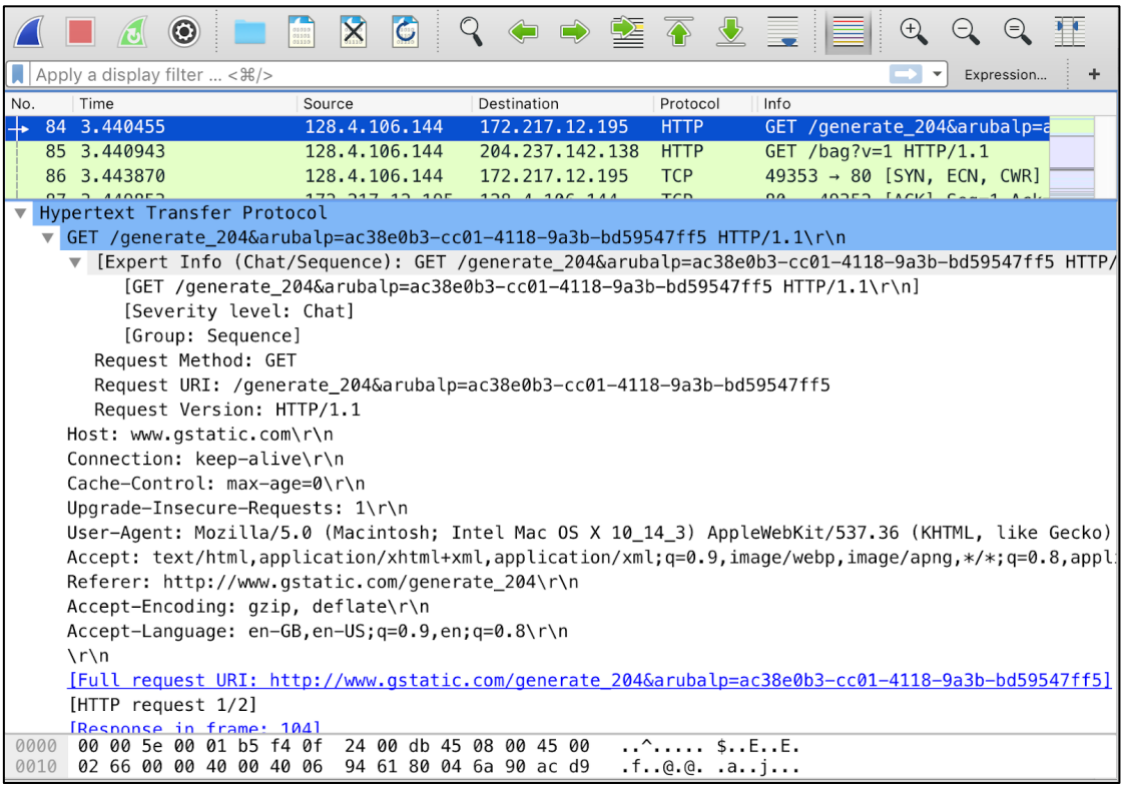

Figure 3.21 HTTP features

3.5.1.3.10 Bootp

Feature name	Taxonomy	Type	Size
Message Type	bootp.type	Unsigned Integer	1 byte
Hardware Type	bootp.hw.type	Unsigned Integer	1 byte
Hardware address length	bootp.hw.len	Unsigned Integer	1 byte
Hops	bootp.hops	Unsigned Integer	1 byte

Transaction ID	bootp.id	Unsigned Integer	1 byte
Seconds Elapsed	bootp.secs	Unsigned Integer	2 bytes
Bootp Flags	bootp.flags	Unsigned Integer	2 bytes
Client IP Address	bootp.ip.client	IPv4 address	-
Your (client) IP address	bootp.ip.your	IPv4 address	-
Private/Proxy autodiscovery	bootp.option.private_proxy_autodiscovery	Character String	-
Relay agent IP address	bootp.ip.relay	IPv4 address	-
DHCP	bootp.option.dhcp	Unsigned Integer	1 byte

Table 3.13 Feature taxonomy of BOOTP protocol

3.5.1.3.11 IGMP

Feature name	Taxonomy	Type	Size
IGMP Version	igmp.version	Unsigned Integer	1 byte
Type	igmp.type	Unsigned Integer	1 byte
Checksum Status	igmp.checksum.status	Unsigned Integer	1 byte
Num Group Records	igmp.num_grp_recs	Unsigned Integer	2 bytes
Record Type	igmp.record_type	Unsigned Integer	1 byte
Aux Data Len	igmp.aux_data_len	Unsigned Integer	1 byte
Num Src	igmp.num_src	Unsigned Integer	2 bytes

Table 3.14 Feature taxonomy of IGMP protocol

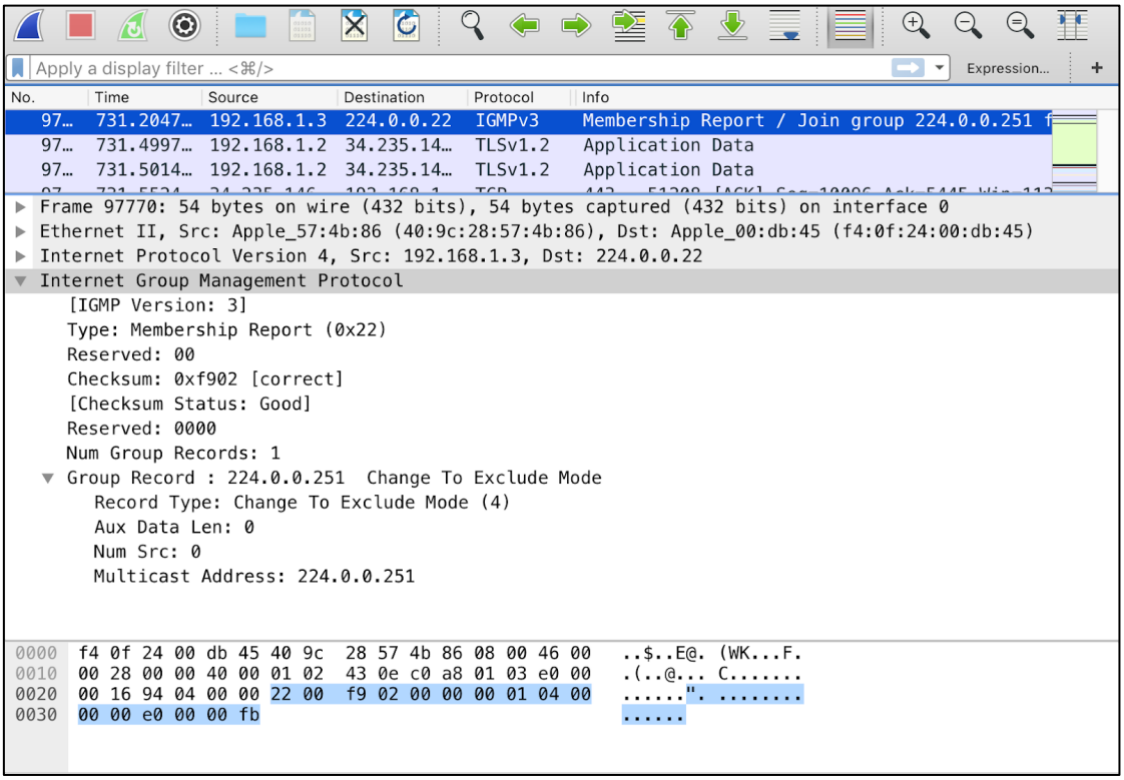

Figure 3.22 IGMP features

3.5.1.3.12 NBNS

Feature name	Taxonomy	Type	Size
Address	nbns.addr	IPv4	-
Transaction ID	nbns.id	Unsigned Integer	2 bytes
Additional RRs	nbns.count.add_rr	Unsigned Integer	2 bytes
Answer RRs	nbns.count.answers	Unsigned Integer	2 bytes
Authority RRs	nbns.count.auth_rr	Unsigned Integer	2 bytes
Questions	nbns.count.queries	Unsigned Integer	2 bytes

Data length	nbns.data_length	Unsigned Integer	2 bytes
Type	nbns.type	Unsigned Integer	2 bytes
Name	nbns.name	Character string	-
Time to live	nbns.ttl	Unsigned Integer	4 bytes

Table 3.15 Feature taxonomy of NBNS protocol

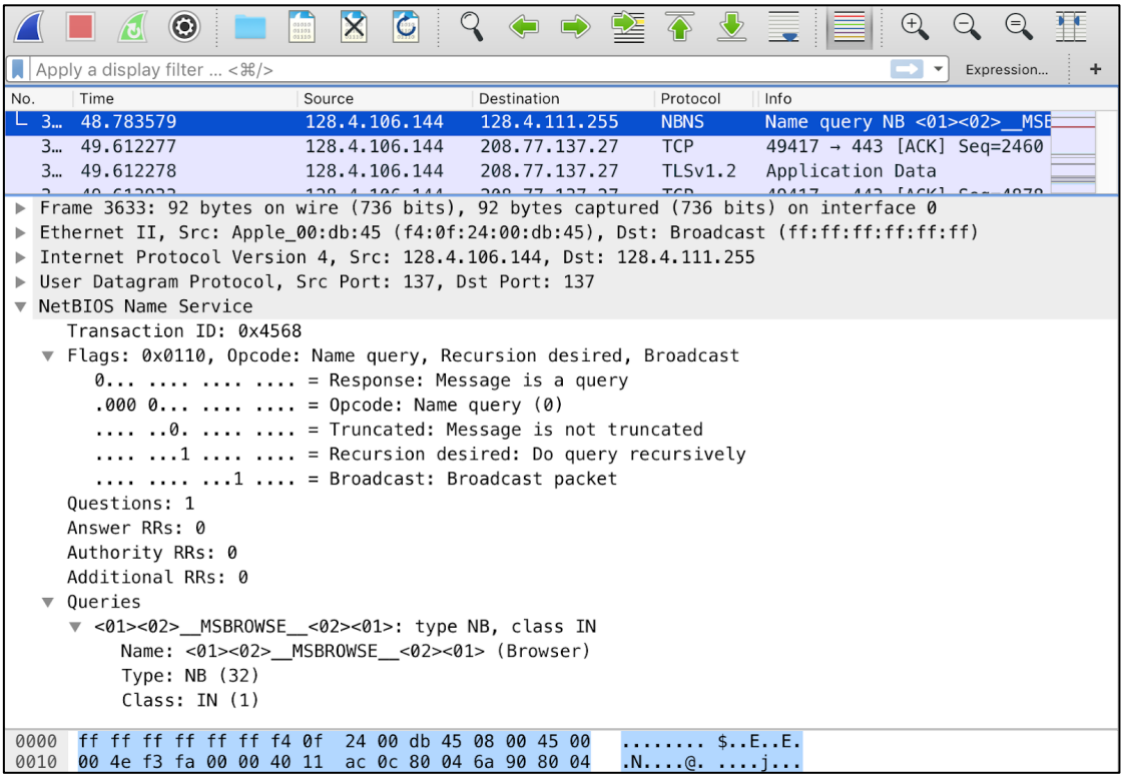

Figure 3.23 NBNS features

3.5.1.3.13 SSDP

Feature name	Taxonomy	Type	Size

Host	http.host	Character string	-
Request line	http.request.line	Character string	-
User-Agent	http.user_agent	Character string	-
Request number	http.request_number	Unsigned Integer	4 bytes
Full request URI	http.request.full_uri	Character string	-

Table 3.16 Feature taxonomy of SSDP protocol

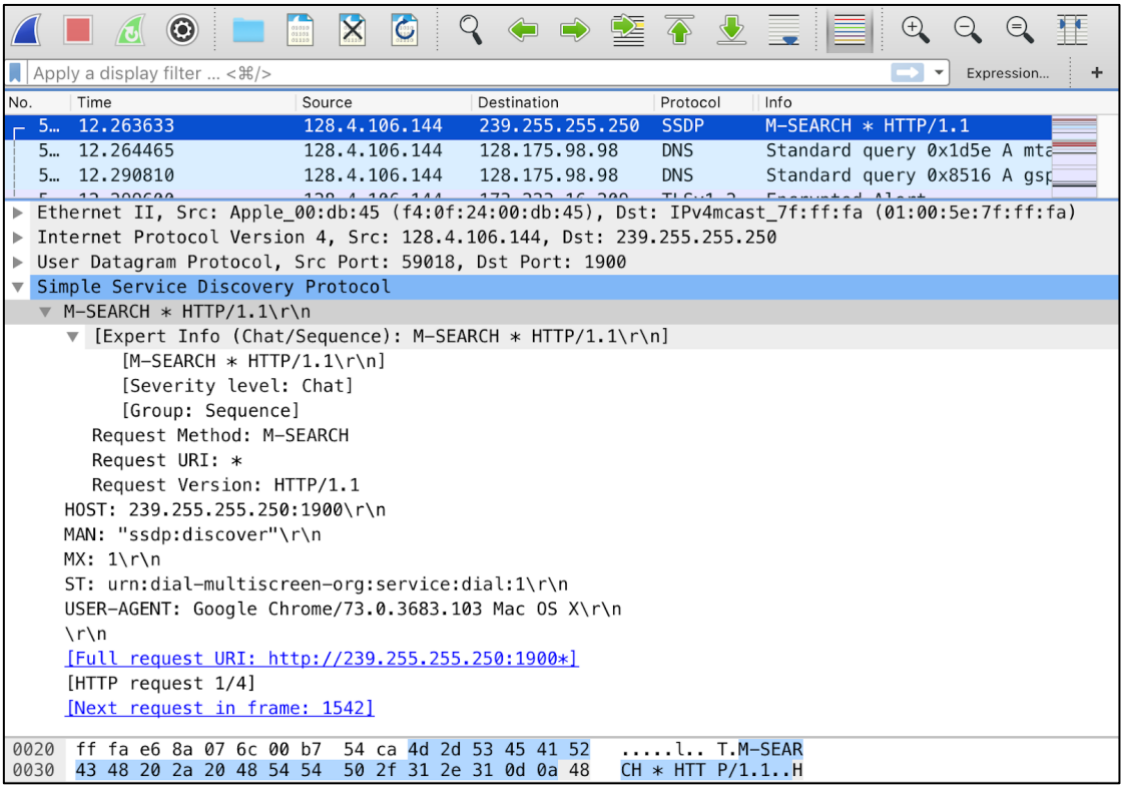

Figure 3.24 SSDP features

3.5.1.3.14 DCE/RPC

Feature name	Taxonomy	Type	Size

Packet type	dcerpc.pkt_type	Unsigned Integer	1 byte
Packet flags	dcerpc.cn_flags	Unsigned Integer	1 byte
Byte order	dcerpc.drep.byteorder	Unsigned Integer	1 byte
Character	dcerpc.drep.character	Unsigned Integer	1 byte
Floating point	dcerpc.drep.fp	Unsigned Integer	1 byte
Call ID	dcerpc.cn_call_id	Unsigned Integer	4 bytes
Alloc hint	dcerpc.cn_alloc_hint	Unsigned Integer	4 bytes
Opnum	dcerpc.opnum	Unsigned Integer	2 bytes

Table 3.17 Feature taxonomy of DCE/RPC protocol

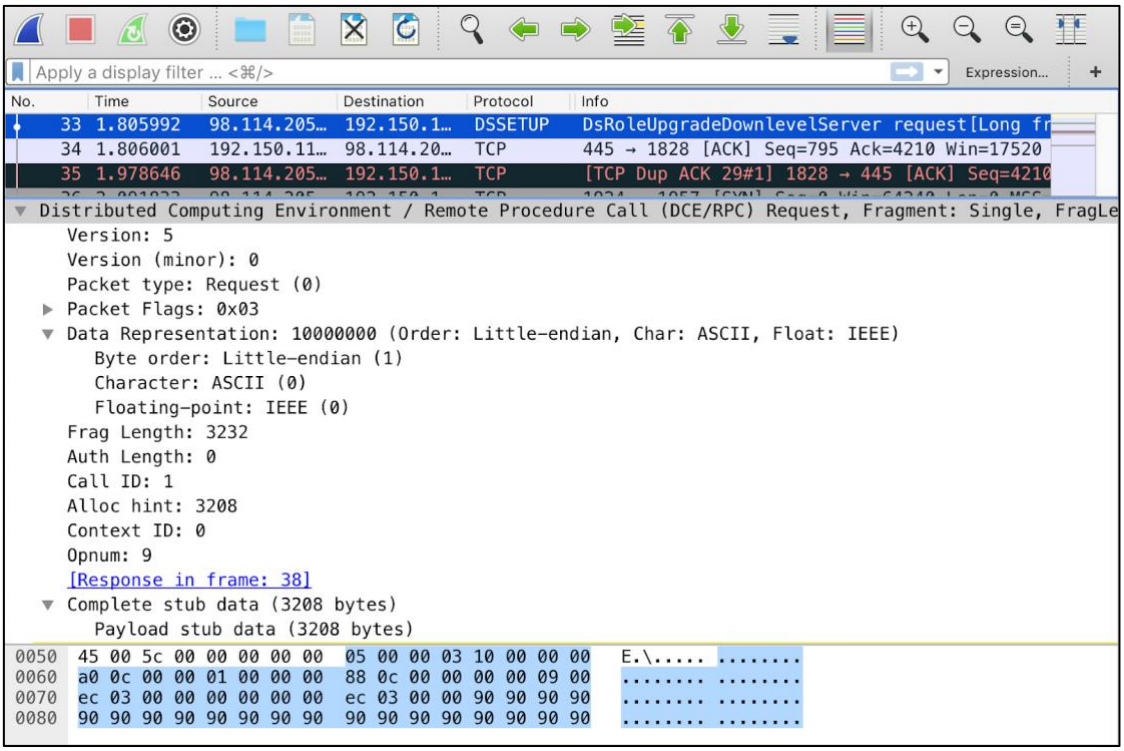

Figure 3.25 DCE/RPC features

3.5.2 Traffic Volume

This branch of the feature taxonomy discusses all the information describing the volume of information seen in the traffic. Volume-based features represent the quantitative measurements of the size or count of entities e.g. the number of packets, packet or field length, average packet length, etc. These measurements are quantified in different groupings or sets of packets, e.g. a flow, in the channel, etc. over a specified period of time e.g. a second, minute, weekly, etc.

Analysis of the volume of traffic is one of the simplest ways to detect anomalous behavior in a network flow. To identify an anomaly, certain metrics of behavior are observed over time. Any disorientation from normal behavior is spotted as an anomaly in the behavior and further investigation is performed on it. An example of the metrics of volume can be average bytes of the data collected on a node. In a normal scenario without an anomaly, this value remains within a specific range for long periods of time. However, in case of an attack, this value may spike to higher values (like DoS) or fall down if the node is brought down. Hence, it is proved that some attack scenarios can be spotted very easily using these simple statistical techniques on volumetric features.

The first step is to identify the entities we want to quantify. When we are talking in terms of network captures, it is obvious to measure the quantities of packets in different sets. In addition to that, we can measure the volume of data that flowed through a node over different periods of time and in different sets. Therefore, this branch is divided into two measurements according to how we are viewing the data flow: packet volume and data volume. There are two ways to view the measurements both as time-series as well as being quantified statistically over different periods of

time (e.g. the packet volume evaluation of “packets per second”, or the data volume evaluation of the “average packet size”).

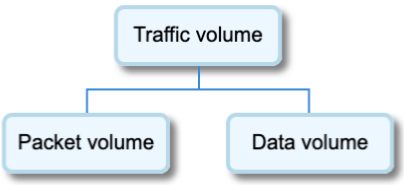

Figure 3.26 Traffic volume

The second step is to identify the metrics we want to use to measure the traffic. This can be done in two ways: we can either measure the count of entities or the size (length) of entities. Both are very important measurements and insightful in anomaly detection. These metrics can be used as follows:

a. Number of packets captured

b. Number of bytes exchanged

c. Size of packets (at different protocol layers, e.g. MAC, IP, etc.)

d. Size of fields

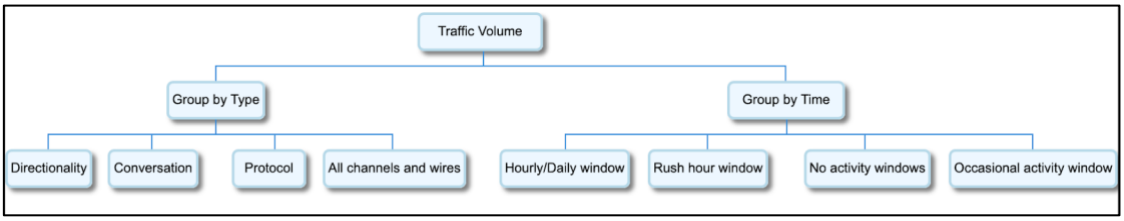

Figure 3.27 Traffic volume grouping

An anomaly can only be identified when we compare the value of any of the above metrics at one instance with other (normal) values of the same metrics at other

instances. So, the third step is to make small groupings or sets and observe the general behavior of these groupings and spot any indifferent behavior.

There are different approaches to group the volumetric features:

3.5.2.1 Group by Type

3.5.2.1.1 Directionality

One of the multiple jobs of a node in a network is to communicate with other nodes in the form of packets. The node first listens to the request from other nodes, and if necessary, send replies. There are several broadcast packets which a node doesn't need to reply. Therefore, this traffic can be either unidirectional or bi-directional.

3.5.2.1.2 Channels

Packets arrive from multiple channels on the wire at the node. The rate of arrival is dependent on external and internal factors. There are cases when the packet loss is occurred due to issues on the wire. Similarly, packets go out to multiple channels on the wire but can be caught in congestion. Monitoring of channels can bring interesting outcomes to the study.

3.5.2.1.3 Conversation or flow

Since a node may converse in a different manner with different nodes, we group the data into its conversation groups. One group contains a conversation from one set of source-destination pair. When we first address an anomaly, we want to find where it is coming from, therefore it is essential to analyze the to-and-fro conversations between each pair of source-destination.

3.5.2.1.4 Protocols

All the packets flowing from one node to another run on multiple protocols at different layers of the OSI model. In order to maintain efficiency, not all the protocols are designed to emphasize on security of information like UDP. Some protocols run the risk of carrying malicious data through them. Therefore, we group the packet captures according to the protocols. To make a comprehensive model, we further divide the outer layer protocols and sub-set them into protocols of the inner layers.

Once we start grouping our captured data into any of the above types, we move on to the analysis phase, but before that, we have to decide how much data we want to analyze in one phase. To find reliable patterns of malicious activity, we analyze the volume of data exchanged during specific periods of time like over a period of a day or week. In our fourth step, we group the volume once again into windows of time and analyze them in groups of the same window.

This process of making groups can be approached in the following ways:

3.5.2.2 Group by Time

3.5.2.2.1 Hourly/Daily window

The straightest way to realize patterns in a group of captured packet streams is to analyze traffic volume on an hourly basis. It is a short span of time which provides a detailed picture of the behavior of the node during the normal course of the day. If we want to get an overall picture of the everyday flow of packets, we analyze packets on a daily basis. For example, during the hourly window, we analyze data from 5 pm to 6 pm every day. On the other hand, if we want to analyze daily, we observe data every 24 hours. We can also analyze packets weekly or monthly, etc.

3.5.2.2.2 Peak hour window

In commercial and financial settings like a bank, thousands of monetary and non-monetary transactions take place every day. Exchanges happen only during fixed hours only from verified sources. In such a case, it is easy to for malicious transactions to sneak their way into the secure traffic. Therefore, we analyze the properties of transactions during rush hours and spot irregularities.

3.5.2.2.3 No activity windows

In industrial organizations, where millions of bytes are exchanged during a fixed period of time every day, traffic at the server during non-rush hours is abnormal and may be a sign of anomalous activity. Therefore, it is important to look for data at windows in time where no activity is expected.

3.5.2.2.4 Occasional activity window

During special occasions like a holiday or a natural phenomenon, there is a spike in the exchange of data in a region. There might be exponential decrease in the online activity of users resulting in an ebb of the packet flow. Such incidents need to be closely monitored and analyzed to prevent unnecessary loss.

3.5.2.3 Statistical Features

The final step is to search for anomalies in the recorded metrics grouped in the above forms. These sets will act as training data for our machine learning model. There are two ways to conclude these anomalies:

3.5.2.3.1 No statistics

In this approach, no statistical operations are computed on the datasets. Raw data collected from the packet captures are used in neural networks (e.g. convolutional

neural networks) to perform supervised learning. Before training the neural networks on these datasets, it is cleaned, transformed and normalized. It is then labeled and trained. When the data stream captured is grouped by time, the dataset is viewed as a time series sequence. We can then perform time-series analysis on these datasets. In this approach, it is the job of neural networks to find intelligence in the datasets to learn the difference between anomalous and non-anomalous traits.

3.5.2.3.2 Traditional statistics

In this approach, traditional statistical operations are derived over the groups created in the previous step. These statistical derivatives are indicators of the behavior possessed by the groups calculated in the previous step. Some of the common statistical operations are:

a. **Max/Min**

Maximum (or minimum) values of column data in all the groups should lie in the same region with a small window of error. If the max and min values of all the columns of the groups are in the same window, they are considered normal. If any max (or min) value do not fall in the same window, it is considered abnormal or anomaly.

b. **Mean**

Mean of the columns in different groups should be similar over time groupings.

c. **Standard deviation**

Standard deviation of the data in a column over a sequence of the groups.

d. **Histogram**

It is a depiction of statistical information over a consecutive interval of time. It is an emphatic way to analyze the sequence of data and detect abnormal patterns.

e. **Null**

It is important to acknowledge the null values in statistical operations any null data captured is abnormal.

A volumetric feature can be generated as follows,

Compute this **statistic** {actual-value, minimum, maximum, average, mean, standard-deviation, histogram, etc.} of the **measurement** {size, count} of this **object** {packet, field} for this **group** {unidirectional, bidirectional, flow, protocol(s)} over this time **period** {all-time, second, hour, day, etc.}.

The result can be a single scalar number, e.g. all-time minimum packet size, the average size of a given field (e.g. TSL segment field length). Or it can be a time series sequence like an average TCP traffic rate in bytes per second over 5-minute measurement windows.

Like the other major categories in the taxonomy, the traffic volume category has an arbitrary set of possible features. In the two subsections that follow, some of the features most likely to be important in the security analysis of packet traffic are defined and discussed.

3.5.2.4 Packet Volume

In this section, we observe the bulk of packets seen at a point in the network (e.g. at a node, or on a wire). Measuring packets is one of the simplest ways to calculate bulk statistics. We analyze this bulk and find voluminous patterns from it. The below figure is a shot of the Wireshark capture. 4104 packets were captured in this Wireshark session. These captures are used to analyze packet bulk statistics.

Figure 3.28 Packet volume examples

We calculate the features using the below formula to calculate

< measure:{size, count}of packets, group:{uni, bi, flow, protocols}, period{second, hour, day,...}, stats{NULL, min, max, avg, mean, std dev, hist} >

Some of these features can be:

3.5.2.4.1 Packets per second

No. of packets (at any layer, but typically MAC) seen at any point over a one second period. The observation point can be a point in a wire (connection) or inbound to a node (ingress) or outbound from a node (egress). It can also be summations of all one or all node interfaces or all measures wires collected in one second. Packets collected in a minute or hour can also be used for the same type of analysis.

3.5.2.4.2 Max no. of packets

Maximum no of packets seen at any point entering or leaving a node over a period of a second or minute. For the purpose of analysis, the minimum no. of packets can also be calculated.

3.5.2.4.3 Average packets per second

Average packet counts captured during a second. Average packets per minute or hour are also calculated for analysis.

3.5.2.4.4 Packet Histogram

A histogram of packet-counts and packet-sizes is a graphic depiction of the patterns found in counts and sizes.

3.5.2.5 Data volume

Similar to the above analysis, this section deals with the analysis of the data in bytes. In this section, we observe the bulk of bytes seen at a point in the network (e.g. at a node, or on a wire). Measuring the volume of data is one of the direct ways to calculate bulk statistics. We analyze this bulk and find voluminous patterns from it. The below figure is a shot of the Wireshark capture. 410 bytes were captured in this Wireshark session between two IP addresses and a total of 812 KB were exchanged in this session. These captures are used to analyze packet bulk statistics.

Address A ▲	Address B	Bytes	Bytes A → B	Bytes B → A
192.168.52.1	192.168.52.141	410	60	350
192.168.52.2	192.168.52.129	174	87	87
192.168.52.129	192.168.52.141	228 k	120 k	108 k

Figure 3.29 Data volume example

We calculate the features using the below formula to calculate

< measure: {size, count}of **bytes**, group:{uni, bi, flow, protocols}, period{second, hour, day,...}, stats{NULL, min, max, avg, mean, std dev, hist} >

Some of the features can be:

3.5.2.5.1 Bytes per second

No. of bytes exchanged during a period of time like seconds or hours.

3.5.2.5.2 Average packet size

Average size of the packets seen during any periodic instance of time.

3.5.2.5.3 Bytes histogram

Statistical representation of byte distribution across dataset.

3.5.3 Protocol state

There are several ways in which a packet reacts to certain conditions. Protocols have a long list of flags which need to be green before a protocol is sent out on the Internet to reach a destination. Some of them are security flags, ACK flags, error flags, etc. Other important information that packets carry are sequential numbers, error bytes. A very important way to detect malicious activity in a system is to observe unusual behavior in before/after/during its interaction with outside systems. In this case, we observe the two-party state behavior. The state of our node, the other nodes it is interacting it with.

The protocol state features deal with closely watching the state of the different component in the packet. By definition, the states are ever-changing and should be address upon the fulfilment of certain conditions. The features defined in this branch of the taxonomy tree have the following functions:

a. Make sure that state of the components/resources are valid.
b. The changes in the state are a result of valid operations.

As much as we can detect invalid activities by watching the sniffed traffic, it can be complicated and time taking. Zeek does a fair job in identifying such unusual operations performed by a node. It logs the unparsed operations in a logger known as weird.log. It provides a detailed statistic about unexplained activities by a node. However, not everything is logged or understood by Zeek. We need to identify those

unusual activities and mark the states which look suspicious. Some of the important checkpoints can be:

3.5.3.1 TCP graceful close

This is the normal way to close a TCP connection. The client sends a FIN packet to the server. The application is set to FIN_WAIT state. The server sends back an acknowledgement(ACK) packet back to the client. Then, it sends all the remaining packets to the client which are in the queue. It also sends a FIN packet after all the packets are sent. The client then sends an ACK to the server. The connection is closed now.

3.5.3.2 Delayed DNS

When the web hosts are changed, the DNS entry in the local cache is rendered obsolete. When a client machine tries to reach a website, it needs to perform a DNS query. If the cache is cleared it goes to the upper level DNS servers (root, Top-level domain, etc.) to find the IP address of the website. This query is a UDP packet, so it travels through a series of hops to find the correct IP address. Once, a response is received, DNS is filled again. The TTL decreases every step. To avoid all this delay, the DNS servers need to be backed up so they can be used in failover

3.5.3.3 ARP request: lots of ARP request and few replies

An example of an attack on the protocol state is ARP request and response. Suppose, if there are 5 hosts in a network and a host receives 200 ARP requests. It means that 195 are anomalous. This type of behavior is a sign of malicious activity. It means that the network was being scanned.

3.5.3.4 Failure nodes in Zeek.

There may be some instances where Zeek nodes fail and are not able to detect attacks. State of the data streams is crucial to observe in this case.

3.5.4 Temporal

A lot of things happen in a very short span of time at a node. In this fast forward exchange of information, it is acceptable to send packets quickly, even if you lose some in the middle. There can be a lot of reasons for a data stream to mess up and cause a disturbance. These disturbances can range from an actual mishap with the node to external interference from malicious sources. We capture such disturbances so that we can record abnormal timings seen in the flow of packets and use them for analysis to detect anomalies. Therefore, we add temporal features as taxonomy branch for anomaly detection.

This branch of the feature taxonomy discusses the instances of time-related information which might be helpful to spot an illegal activity in a packet stream. The analysis of temporal features of a data stream is very crucial because all data streams flowing through a node are time critical operations. An example of such operation is the time of the arrival or delivery of packet sequences. If any discrepancy is found in their timings, it can be assumed to be the result of malicious activity. This identification of the features works in similar ways as the traffic volume branch of the taxonomy except this represents time-based features.

The temporal features are further divided into two branches depending upon how we are quantifying time.

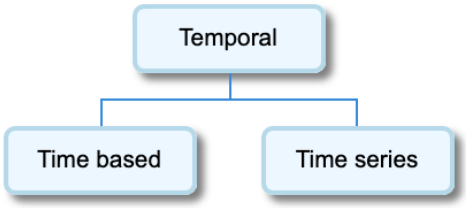

Figure 3.30 Temporal features

3.5.4.1 Time-based features

Network packets are exchanged at a very high rate between nodes and thousands of packets pass through each node very quickly. At this high exchange, it possible that some packets are misplaced or lost in the mix. In some cases, this loss of packets is acceptable like pixels lost causing blurry pictures during video calls, however, in some cases, this loss can be harmful like breaking a connection over a congested channel. Therefore, we need to make sure maximum packets arrive on time and in sequence at a destination. If we find any discrepancy, we need to take a look at it to find the root causes of delay or total loss. Time-based features represent the quantitative measurements of time (e.g. TTL, the lag between packet arrivals, etc.) in different settings of packet-captures (e.g. a flow, in the channel, etc.) represented in any unit of time (e.g. seconds, minutes, etc.). Just as we studied volumetric features in the above section where the metrics were size and counts of captures, in this analysis of time series features, the metrics are time-space and exact instances of time units like seconds, milliseconds, etc.

The first step is to identify the metrics we want to measure. When we are timing the network captures, it is obvious to measure the duration of flows of packets in all possible conditions. In addition to that, we can measure the time-space of a capture. Time-space of a packet can be defined as the time difference between two

consecutive captures that flowed through a node over different periods of time and in different sets. It can be represented as **Δ**t (delta_t). Both of these metrics are calculated over periods of time so that they can be used for statistical evaluations and find behaviors in different scenarios. Some of the examples of these metrics are average time-space between arrival time two consecutive packets from the same session, the average value of TTLs during different instances in time, etc.

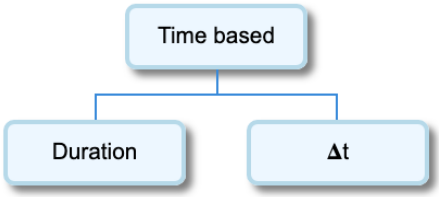

Figure 3.31 Time based grouping

The second step is to group these metrics by type of packet/flow. An anomaly can only be identified when we compare the value of any of the above metrics at one instance with other (normal) values of the same metrics at other instances. So, the third step is to make small groupings or sets and observe the general behavior of these groupings and spot any indifferent behavior. There are different approaches to group the volumetric features:

3.5.4.1.1 Group by Type

a. **Directionality:** One of the multiple jobs of a node in a network is to communicate with other nodes in the form of packets. The node first listens to the request from other nodes, and if necessary, send replies. There are several broadcast packets which a node doesn't need to reply. Therefore, this traffic can be either unidirectional or bi-directional. When a node sends a request, it

expects a response. If the response is not sent within a small time, the conversation becomes obsolete. Therefore, we group the response times by directionality.

b. **Response Type:** Conversation between two nodes is initiated by a sender node. If the destination node replies by a response, the conversation is sustained otherwise, the connection is suspended. Responses vary with the request. For example, the response to a DNS query is an IP address, however the response to a SYN packet is an ACK packet. These responses arrive at different timings. Therefore, it is essential to group the packets by response type.

c. **Conversation or flow:** Since a node may converse in a different manner with different nodes, we group the data into its conversation groups. One group contains a conversation from one set of source-destination pair. When we first address an anomaly, we want to find where it is coming from, therefore it is essential to analyze the to-and-fro conversations between each pair of source-destination.

d. **Protocols:** All the packets flowing from one node to another run on multiple protocols at different layers of the OSI model. In order to maintain efficiency, not all the protocols are designed to emphasize on security of information like UDP. Some protocols run the risk of carrying malicious data through them which can lead to loss of packets. Therefore, we group the packet captures according to the protocols. To make a comprehensive model, we further divide the outer layer protocols and sub-set them into protocols of the inner layers.

Once we start grouping our captured data into any of the above types, we move on to the analysis phase, but before that, we have to decide how much data we want to

analyze in one phase. To find reliable patterns of malicious activity, we analyze the lapse in the of packet exchanged during specific periods of time like over a period of a day or week. In our third step, we group the metrics once again into windows of time and analyze them in groups of the same window. This process of making groups can be approached in the following ways:

3.5.4.1.2 Group by Time

a. **Hourly/Daily window:** The straightest way to realize patterns in a group of captured packet streams is to analyze them on an hourly basis. It is a short span of time which provides a detailed picture of the behavior of the node during the normal course of the day. If we want to get an overall picture of the everyday flow of packets, we analyze packets on a daily basis. For example, during the hourly window, we analyze data from 5 pm to 6 pm every day. On the other hand, if we want to analyze daily, we observe data every 24 hours. We can also analyze packets weekly or monthly, etc.

b. **Peak hour window**: In commercial and financial settings like a bank, security-efficient packets are exchanged at a very high rate every day. Exchanges happen only during fixed hours only from verified sources. At such a high rate, even if every packet is examined, it is easy to for malicious transactions to sneak their way into the secure traffic. Therefore, we analyze the properties of transactions during rush hours and spot irregularities.

c. **No activity windows:** In industrial organizations, where millions of bytes are exchanged during a fixed period of time every day, traffic at the server during non-rush hours is abnormal and may be a sign of anomalous activity. It can be

part of the routine activity or malicious activity. Therefore, it is important to look for data at windows in time where no activity is expected.

d. **Occasional activity window:** During special occasions like a holiday or a natural phenomenon, there is a spike in the exchange of data in a region. This can lead to congestion and highly volatile connections. Such incidents need to be closely monitored and analyzed to prevent unnecessary loss.

The final step is to search for anomalies in the recorded metrics grouped in the above forms. These sets will act as training data for our machine learning model. To conclude these anomalies, we use traditional statistics. In this approach, traditional statistical operations are derived over the groups created in the previous step. These statistical derivatives are indicators of the behavior possessed by the groups calculated in the previous step. Some of the common statistical operations are:

3.5.4.1.3 Statistical Operations

a. **Max/Min:** Maximum (or minimum) values of column data in all the groups should lie in the same region with a small window of error. If the max and min values of all the columns of the groups are in the same window, they are considered normal. If any max (or min) value do not fall in the same window, it is considered abnormal or anomaly.

b. **Mean:** Mean of the columns in different groups should be similar over time groupings.

c. **Standard deviation:** Standard deviation of the data in a column over a sequence of the groups.

d. **Histogram:** It is a depiction of statistical information over a consecutive interval of time. It is an emphatic way to analyze the sequence of data and detect abnormal patterns.

e. **Null:** It is important to acknowledge the null values in statistical operations any null data captured is abnormal.

A time-based feature can be generated as follows,

Compute this **statistic** {actual-value, minimum, maximum, average, mean, standard-deviation, histogram, etc.} of the **measurement** {duration, **Δ**t} of this **object** {packet, field} for this **group** {unidirectional, bidirectional, flow, protocol(s)} over this time **period** {all-time, second, hour, day, etc.}.

3.5.4.2 Time-series features

The process of input or output of packets at a node is done continuously over time. Each packet sequentially captured in a data stream is an entity unique in time. Each of these entities can be represented as a combination of a set of properties. This bag of entities, when collected equidistant in time or when they have the same lag, is called time-series. A dataset is called time series data as it represents a set of properties seen at instances in time which are equally spaced. The time series data is a matrix of a set of characteristics where each row is a unit of time when those characteristics is recorded.

Time series analysis is a behavioral analysis technique of entities distributed equally in time. The behavior of the features is analyzed over all the entities present in the dataset. This analysis helps us to understand the difference between the values of the features in normal conditions and anomalous conditions. Analysis of time-series data is imperative to feature taxonomy of a network as the packets exchanged between

nodes are a function of time. Delay in arrival or departure of packets can lead to issues like loss of connection and high congestion windows.

In this approach, no statistical operations are computed on the datasets. Raw data collected from the packet captures are used in neural networks (e.g. convolutional neural networks) to perform supervised learning. Before training the neural networks on these datasets, it is cleaned, transformed and normalized. It is then labeled and trained. When the data stream captured is grouped by time, the dataset is viewed as a time series sequence. We can then perform time-series analysis on these datasets. In this approach, it is the job of neural networks to find intelligence in the datasets to learn the difference between anomalous and non-anomalous traits.

3.5.5 Traffic flow

Observations in this branch of feature taxonomy are the information captured from the network traffic. Data in the form of packets travel across the Internet all over the earth and space via satellites. During this time, it can get infected with malicious information. Therefore, it is essential to go through the whole traffic and find infectious data. We have discussed identifying infections in packets in the protocol state branch of the taxonomy. In this branch, we will discuss the series of packets as a set and perform analysis on the flow.

Throughout the history, there have been a lot of famous cyber-attacks where the organization which was attacked has no information that they were being attacked. This happened because of the fact that the attack was performed so meticulously that no alarms were triggered. In such type of cases, even though no packets were found individually malicious, the attack was carried out over a long period of time and successfully executed. To overcome this situation, behavioral analysis of network

traffic needs to be taken seriously. The flow of the traffic as a whole is observed to find anomalies. To accomplish this task, a data stream flowing through a node is captured and saved, its properties as a flow are analyzed to find abnormal behavior over time/session.

Flow of traffic at a node is a continuous operation. To keep track of the observations, we group them according to our requirements. Therefore, we look at the flow in two ways: time-based and session-based depending upon the requirement.

3.5.5.1 Group by Time

In time-based approach, the division of traffic is done into fixed intervals of time and its flow characteristics are observed. A network analyzer at an end host continuously screens traffic arriving and exiting the host. In this approach, we want the quantitative analysis of flows. For the purpose of analysis and detection of unusual behavior of flows, we observe them hourly or daily.

Zeek is a very advanced network analyzer which generates hourly reports of the traffic as long it is in operation. These reports contain the connections statistics, signature errors, downloaded files, etc. We can use this data to analyze periodic behavior of the flow at node. We can analyze flow in the following ways:

3.5.5.1.1 Hourly/Daily window

The straightest way to realize patterns in a group of captured packet streams is to analyze them on an hourly basis. It is a short span of time which provides a detailed picture of the behavior of the node during the normal course of the day. If we want to get an overall picture of the everyday flow of packets, we analyze packets on a daily basis. For example, during the hourly window, we analyze data from 5 pm to 6 pm

every day. On the other hand, if we want to analyze daily, we observe data every 24 hours. We can also analyze packets weekly or monthly, etc.

3.5.5.1.2 Peak hour window

In commercial and financial settings like a bank, security-efficient packets are exchanged at a very high rate every day. Exchanges happen only during fixed hours only from verified sources. At such a high rate, even if every packet is examined, it is easy to for malicious transactions to sneak their way into the secure traffic. Therefore, we analyze the properties of transactions during rush hours and spot irregularities.

3.5.5.1.3 No activity windows

In industrial organizations, where millions of bytes are exchanged during a fixed period of time every day, traffic at the server during non-rush hours is abnormal and may be a sign of anomalous activity. It can be part of the routine activity or malicious activity. Therefore, it is important to look for data at windows in time where no activity is expected.

3.5.5.1.4 Occasional activity window

During special occasions like a holiday or a natural phenomenon, there is a spike in the exchange of data in a region. This can lead to congestion and highly volatile connections. Such incidents need to be closely monitored and analyzed to prevent unnecessary loss.

3.5.5.2 Group by Session

In session-based approach, traffic is divided into sessions. Sessions are the interaction between a client and a server for a certain duration of time. If our device is

the client, a session is unique to the server our device is interacting with. A device may interact with multiple servers at the same time. Therefore, while capturing traffic using sniffing devices like Wireshark or Zeek, we may capture many sessions.

Wireshark downloads all the traffic in one file, and we need filters to manually filter out the session by providing specific source and destination IP addresses. Zeek has advanced features where it can group connection between two nodes automatically and shows their interaction like the bytes exchanged and packets flowed. The features that can be found in periodic evaluations of a flow of network traffic can be:

3.5.5.2.1 Frequency of protocols used

Number of different types of protocols used during the duration of time or the session we are taking into observation.

3.5.5.2.2 Average number of bytes exchanged

Average number of bytes exchanged during the duration of time or the session we are taking into observation.

3.5.5.2.3 Average lengths of exchanges

Average number of packets exchanged during the duration of time or the session we are taking into observation.

3.5.5.2.4 Average duration of a connection

Average duration of the connection taken into observation during the duration of time or the session.

3.5.5.2.5 Average number of ports used

Average number of ports used during the duration of time or the session we are taking into observation.

A traffic flow feature can be generated as follows,

Compute this **statistic** {frequency, average, quantitative, etc.} of the **measurement** {period, session} of this **object** {flow}.

3.5.6 Computed Features

Computed features are not directly present in the dataset that we capture at the IDS, neither during the exploratory analysis phase. They are a set of features which are created explicitly to implement the domain knowledge of network captures and make the taxonomy more robust to unexplored scenarios. Some of the notable computed features are:

3.5.6.1 n-grams

Any sequences containing a group of "n" adjoining elements. An n-gram model is a statistical technique to identify patterns in a large set of data. In a n-gram model, the dataset is run over with a sliding window of length n. All entities which have the same n-items in a window are statistically related. n-grams models are generally used for prediction in language processing entities. A very clear example of n-gram is our DNA. DNA sequences are made of nucleotides and the sequence of nucleotides in a DNA sets them apart from others. The nitrogenous bases in the nucleotides are adenine, guanine, cytosine and thymine (A, G, C, T). It makes DNA a 4-gram.

In a data stream, with a fixed number of elements contributing to the dataset, we can look for n-grams present in normal packets. There are three ways to use a n-gram model:

a. Identify n-grams which should be present in the traffic for a traffic to be identified as normal

b. Identify n-grams which should not be present in a normal traffic. If these n-grams are found, traffic might be malicious.

Although, presence or absence of n-grams in a packet is not necessarily an absolute indicator of anomaly, long-term behavior of these entities can be an indicator of their nature.

3.5.6.2 Entropy of compressed payload

Entropy of a system is an indicator of its diversity or disorder. It is directly proportional to randomness. Higher the entropy[31] of a payload is, the less redundant it is. Less the redundancy of the payload, less compressible it is. Compression techniques can be either lossy or lossless where bits of the payload are statistically reduced after compression than the original bits. Lossy compression can't be fully decompressed as while lossless can be. Entropy is a measure of limit of lossless compression.

When a packet is sent from one node to another, it is compressed at the browser to increase the efficiency. In Google chrome, Shared Dictionary Compression for HTTP (SDCH) scheme is used and the most common HTTP compression technique - gzip is used where the payload is encoded. The below figure depicts that gzip was used to encode the packet and compress it.

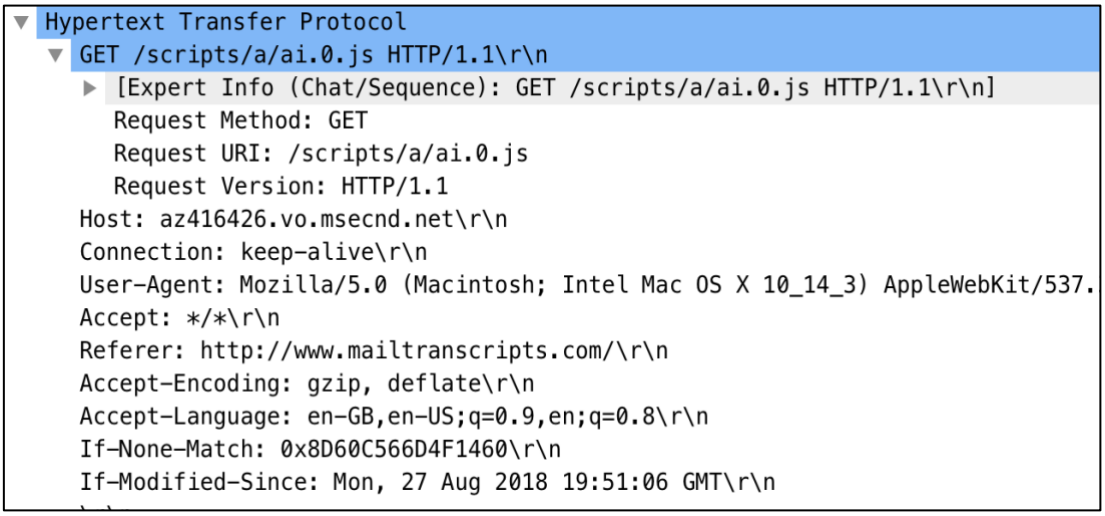

Figure 3.32 HTTP compression

Although compression does not help in securing data, it is used by all web servers to increase the speed of transfer. Direct observation of compressed bits cannot be performed on the packets. Therefore, we record the before and after data of the decompression scenario and calculate the usage statistics of the compression techniques.

3.5.7 Other

All the other features which do not fall under the above categories go into this category. Properties displayed due to human error can be a part of this classification. Attacks like SQL injections[36] and cross-site scripting show some unique properties which can be captured in this category.

Chapter 4

MACHINE LEARING USING TRAFFIC DATA FEATURES

A lot of people confuse machine learning with artificial intelligence. AI is a technology and machine learning is a methodology for implementation of AI. Machine learning, as the name suggests, is a way of making a computer knowledgeable. The computer learns from the algorithm it runs with the help of data that we train it on. We don't need to teach it a lot of algorithms. They are small set of algorithms which can be used differently in different situations which means we don't need to explicitly write algorithms for specific problems. For example, a classification algorithm which can classify spam and non-spam emails, can also classify one image from the other. The only difference here is that we train a general machine learning algorithm on different training datasets. Once, the machine is trained, it has learned and it ready to make decisions for example, a classifier can classify any unknown data that we give it. This process is called modeling and the learned system is called a model. But how do we know that model's decisions are right. Well, the metric which validates a model is its accuracy which is tested on unseen data. Let's dig deep into machine learning.

4.1 Learning

Learning for machines is divided into four general categories, supervised, unsupervised, semi-supervised and reinforcement.

4.1.1 Supervised Learning

As the name suggests, in supervised learning an algorithm knows what it is looking for. The training data provided to it is labeled with output class. Its ultimate goal is to find a correlation between the input and output. The algorithm will need to fit itself with the training data to reach a reasonable parametric equation which will try to fit with all the samples in the training data. This process is cyclic to reach an equation by minimizing loss.

Examples of supervised learning models are classification, decision trees, neural networks, support vector machines and regression.

4.1.2 Unsupervised Learning

Unsupervised learning is used when there is no output class defined for the system. It does not know what it's looking for. It is the task of a non-custom algorithm to find patterns, develop an equation which calculates input values from the sample data and form groups of alike samples. It is the most common form of machine learning. There is no need to train the samples. Examples of unsupervised learning models are cluster analysis, pattern recognition, genetic programming and association rules.

4.1.3 Reinforcement Learning

It is dependent on a feedback system, where the model learns from its previous steps. It doesn’t know the output, nor does it knows how to reach the output. If it learns the wrong thing, the user can penalize it, if it learns the right thing the user can reward it, and so it learns. Examples are driving a vehicle or playing a game.

4.1.4 Semi-supervised Learning

It is combination of unsupervised and supervised learning. It is done when we have some missing information from the labeled dataset. First, we perform unsupervised learning on the dataset to identify clusters. Then, cluster wise we fill the missing values with the mode or mean of that cluster. Once all the samples are completely filled, we perform supervised learning.

4.2 Supervised Learning Models

Since, we want to detect an anomaly based on numerous attributes, we are interested in working with supervised learning in this study, so we will dig deeper into some of the famous supervised machine learning models.

4.2.1 Regression models

Regression models are considered the simplest. These models work for a continuous data, for example age of person, sales of a company, salary of an employee etc. The training data for regression models should be real numerical values. They are used to predict output values of continuous data. Regression are a statistical way of finding correlation between the predictor (X) which is independent variable(s) and the target/output label (y) or dependent variable of a sample. This correlation can be depicted in the form of a parametric equation which tries to fit with all the values in a sample with minimum root mean square errors (RMSE) and predictions for unseen cases are mapped based on that equation. The figure below[34] shows the distribution of X and y. In an ideal world, a regression model will try to fit a curve on this distribution which should pass through all these points. Unfortunately, that is very difficult to achieve, so a model is computed where the curve passes between these points in such a way that distance between them is minimum.

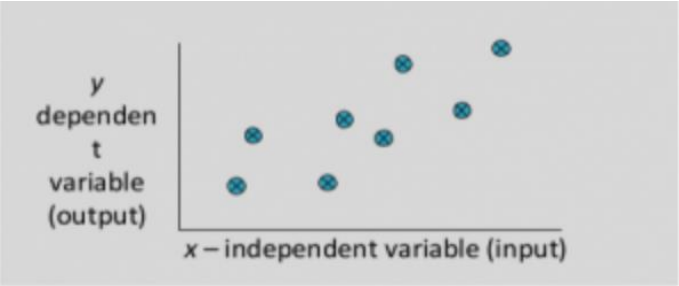

Figure 4.1 Distribution of points in X-Y plane

If the predictor X contains a single feature, it is called a simple model, else if X is conjunction of more than one features, it is called multiple model. The common types of regression techniques are linear regression, logistic regression, polynomial regression, ridge regression, stepwise regression, lasso regression and Elasticnet regression.

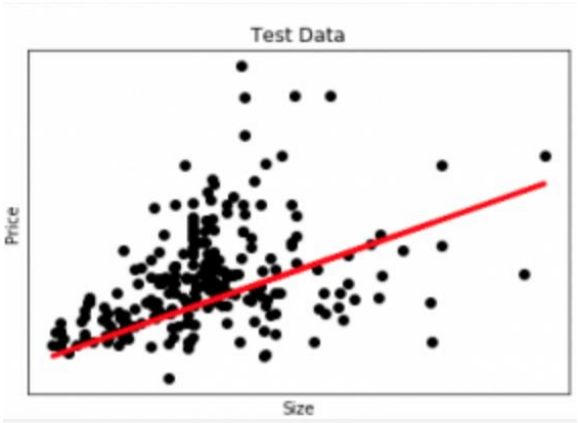

Figure 4.2 Linear regression

Above is figure[33] depicting linear regression for a housing price prediction project. It shows the relationship between the price of the house given the size. As we can see, the red line is the fitting equation that defines the linear model but not all the test samples fall on the line. Some samples are very far away from the line which

means the RMSE values would be very high for them. Thus, the linear regression is not accurate and therefore cannot be used in all situations.

4.2.2 Classification Models

Classification models are the most common application of supervised machine learning. Classification models are tasked to make observations from non-continuous categorical data. The output labels of a classification model are the categories. A very common example of classification model is categorizing an email as ‘spam’ or ‘not spam’. Here, the features can be email address of the sender, number of emails per day, number of links in an email, etc. and the ‘spam’ and ‘not spam’ are the classes or output label.

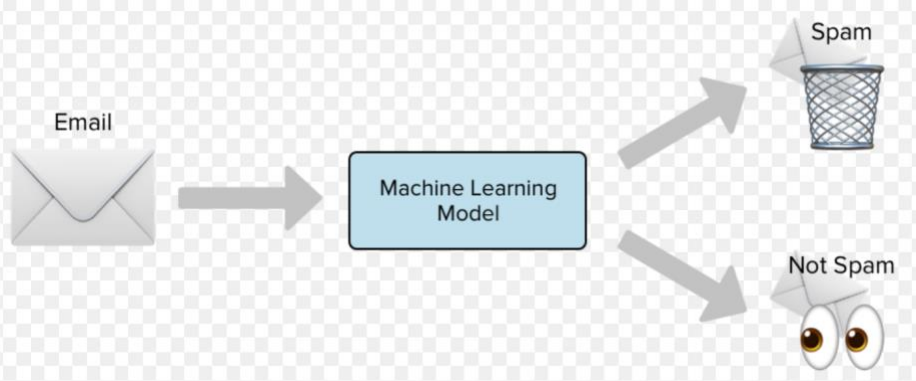

Figure 4.3 Machine learning based classification

The classification model can be either binary as depicted in above figure or multi class when the number of categories are more than two. The classification model is heavily dependent on the quality of training data containing discrete sequence of the features (X) and the category (y). To test whether the model is working or not, we test

it on new samples and find the accuracy by calculating the number percentage of correct hits over all hits.

4.2.3 Support Vector Machines

Support vector machines are highly efficient in solving non-linear supervised learning problems. Support vector machines are an equation which represent some hyperplanes A hyperplane is like a decision boundary between two or more classes and support vectors are data points which validate the hyperplane.

4.2.4 Artificial Neural networks

Artificial Neural Networks are one of the simplest models closer to artificial intelligence. One important specialty of these artificial neural networks that make them similar to biological ones is the ability to make decisions just based on the dataset they know i.e. training data without need a complicated algorithm to perform computations.

The main components of a neural network are neurons, weighted connections, layers, propagation function, bias and a learning rule. A layer is a vertical array of neurons. Value of a neuron from the previous layer is propagated to all the neurons of the next layer as output after multiplying it with some weight. This output goes to the next layer as input. This The more layers we create, the more fitting functions we can create but at the same time we can increase bias and do not react well to new data. In addition to these computations at the layers, we need some more functions like activation functions and feature scaling. The below[32] figure is plain representation of Artificial Neural Networks.

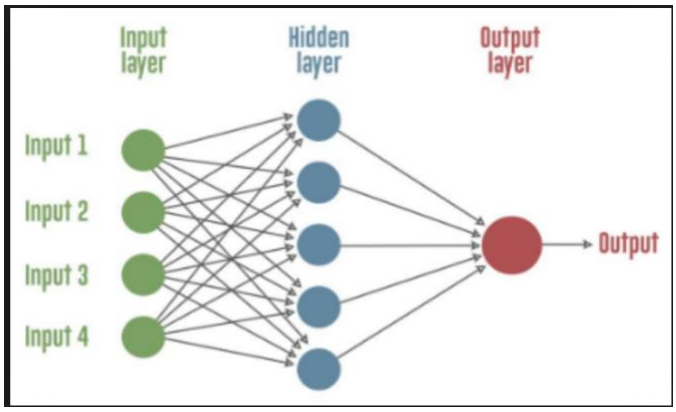

Figure 4.4 Artificial neural network

Neural networks can be categorized into stateless and stateful. A stateless neural network does not remember its previous states which means that its response will remain same with respect to a certain input. However, in a stateful neural network the neural net remembers its previous state and it affects its decision in the current state. In this way, it is continuously learning. Convolutional neural networks (CNN) and multi-layer perceptron models are stateless neural nets, whereas recurrent neural networks (RNN) and Long Short-Term Memory (LSTM) are stateful.

It has been widely assumed that any supervised learning problems can be solved by either using classification or regression if it gives more than 90% accuracy. Even though both the categories have the same goal to predict an output class of a sample, they approach the problem in a different way. They do not always give the highest accuracy.

4.3 Stages of Machine Learning Problem Solving Process

4.3.1 Identify the problem

The first stage of solving a machine learning problem is to actually identify the problem. This step will help us in realizing if the problem is actually a machine learning problem or not. If it is a machine learning problem, then further categorization into supervised or unsupervised can be made. If a problem is a prediction problem most common in economic field, it can be solved using supervised learning, or else if it is more of an exploration, we can it by using unsupervised learning. To successfully identify a problem, we should explore the following sub-tasks:

a. Define the objective of the problem

b. Define the metric which decides the success criteria

c. Discover constraint over the solution of the problem, if any.

4.3.2 Data Gathering

During this stage, dataset which will help us formulate our model is acquired. In our case the data has come from IoT devices. In case of supervised learning, the dataset should represent all the output classes justifiably, whereas in unsupervised

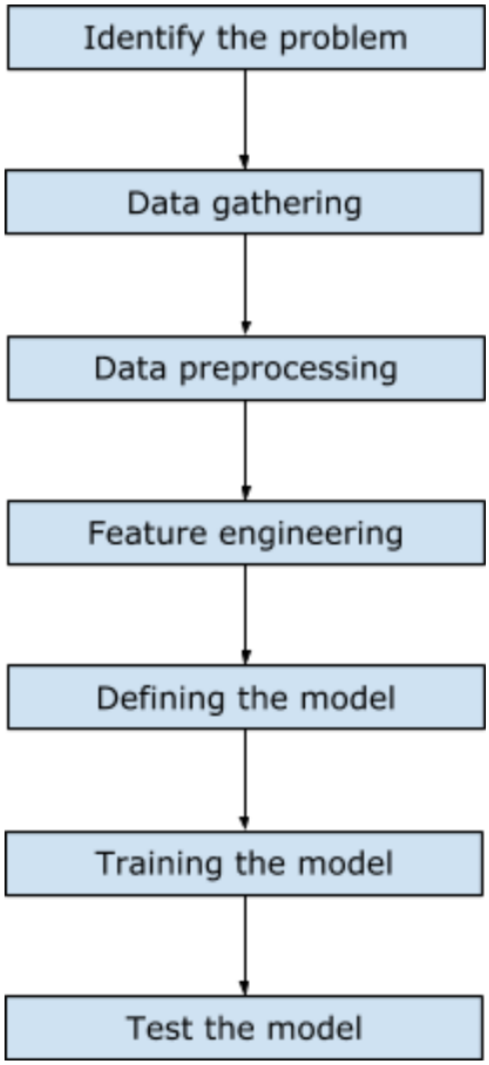

Figure 4.5 Stages of ML process

learning, the dataset should represent true characteristics. We perform the following tasks to ensure successful completion of this stage:

a. We assess the type of dataset we need

b. We access the data from proper means

c. Data exploration to find interesting facts about the data and clear understanding.

4.3.3 Data Preprocessing

After the dataset has been identified, in this stage, the dataset is cleaned to remove any noise, repetition, and unrelated information which does not contribute to the problem to be solved. We perform transformation of values in the dataset if required. We also perform normalization on dataset so that all the numeric columns are follow the same distribution which is normal distribution. The following tasks are performed during phase:

a. Data cleaning from noise and repetitive values

b. Data normalization to bring all the columns on the same scale

4.3.4 Feature Engineering

The dataset collected directly from the IoT devices during the data gathering phase may have several impurities and noise in it. We cannot use these raw features directly in our model. Therefore, we need to clean the data first and then perform feature engineering on the dataset so that we can convert the features into estimators of the model. There are two reasons to perform feature engineering:

a. To discover new feature to make the model more effective

b. To implement domain knowledge of the data and make the training data more compliant to the algorithm so it can understand the input.

This stage of problem solving is the most important and maximum significant proportion of our effort is required in this stage. Other than directly using the crude fields in the dataset, we perform feature engineering to deduce new information and develop a more sensible model which is capable of understand the underlying reasoning, rather than just a probabilistic decision system. For example, if we are performing stock prediction modeling, we want our machine learning model to

understand how the stock markets work. In order to make our model excel and understand these concepts, we develop equations to make statistical computations to our dataset, which in turn discover new features to train our model on. This will make our model more effective for new unseen data.

Feature engineering is performed in the following steps:

4.3.4.1 Feature selection

During this step, irrelevant features or partially relevant features are dropped from the training set. These features might negatively influence the decision-making capability of the model. These can also result in long training periods, so we avoid such features. To find the relevant features, we find the relationship of output variable with other features. Following are the common practices used to achieve this relationship:

a. We discover the relationship of each feature with the output variable using statistical tests like chi-square tests. This process is called **univariate** selection. These tests show the features which have the strongest relationship with the output variable.

b. Several tree-based classifiers are helpful in calculating the important features like Extra Tree classifier.

c. Correlation matrices and heatmaps are very helpful in visualizing the features which are closely related to the output variable.

4.3.4.2 Feature Creation

Valid datasets to solve certain machine learning problems are not always easy to find. Due to the issue of information sensitivity, especially in the field of

cybersecurity, most of the datasets are either truncated or too corrupt to even be downloaded on a machine. In some cases, we have to adjust with the dataset that is made available to us. In this scenario, feature creation is very important. A set of newly created features are derived from the information that we already have in the training set. We create new features for the following reasons:

a. Due to unavailability of rich datasets, we tend to use whatever data we have and enrich it by performing some statistical operations. These operations are the art of adjusting the model to improve its domain knowledge. Some datasets are plain numbers and do not possess any intelligence. We modify such datasets to provide insights to the behavior of the plain numbers. One of the simplest statistical operations include mean, mode, median, standard deviation etc.

b. Not all the features in the training set can be directly used as their formats might mismatch. For. example, some values may be numerical, and some may be Boolean. Since the neural network only works with numbers, we need to deal with such mismatch we introduce solutions like hot encoding and dummy variables. These create encoded columns which represent categorical data which can be directly used to train the model.

4.3.4.3 Feature Compliance

The newly created features need to be tested if they can be implemented into the model. As we know that the neural network will work only with numbers, we need to make the features comply. If it does not fit the model, we need to devise the features again.

4.3.5 Defining the model

In this stage, we model the objective to perform either clustering or prediction. For example, for clustering, we can use k-means algorithms, and for classification we can use SVM or neural networks. There are a lot of framework available to develop a neural network like TensorFlow, Keras etc. which help us make neural networks and libraries like scikit-learn which help make clustering or classification models. To make a model, we perform the below tasks:

a. Decide which approach will help us develop the model

b. Select the modeling algorithm

c. Use libraries and framework to build it.

4.3.6 Training the model

This is the stage where the model is constructed into an intelligent machine. It takes each sample of training data one-by-one, learns its values, moves on to the next sample until the end of the dataset. Once it is trained, it is deemed to be learned and ready to use. It is not always necessary that model learns from one cycle of training also known as one epoch, especially in case of a neural network. Therefore, we run it through several epochs for better accuracy. Also, the layers of the model are not so long that it can accommodate all the training data in one go, therefore we send the samples in small batches. The number of batches and the number of epochs are called the "hyperparameters". In case of a clustering model, the trained model is set of clusters of samples in a training set.

4.3.7 Testing the model

As the model is trained now, we still don't know if it is the best version of the model. We have to verify if the trained model has learned what it is supposed to learn.

Some models perform better than others. To test a supervised learning model, we give it unseen test data and the models give a prediction of the test data in a percentage form. It gives the percentage of the output being one class or the other. The output class which has the highest percentage is the prediction of the model. If it is an unsupervised model, the output of the evaluation will be the assignment of the test data into a cluster.

4.3.8 Tuning the parameters

Sometimes, we are not satisfied with the performance of our model and sometimes are curious to know if it can perform better. In this case, we can change the values of the hyperparameters to find out if we can get better results.

4.3.9 Prediction

Once we are satisfied with the result, our model is ready for operational use. We can deploy it in our business and make predictions. In our case, it will be ready for anomaly detection.

4.4 Time series data

Any data is a measurement of a set of entities at different instances of time. In order to perform data analysis, there are three different types of data that spike interest:

a. **Time series data:** A series of recorded data points where any two consecutive points are recorded equidistant in time. For example, stock prices of a company collected over a continuous period of time, customer churn data, etc.

b. **Cross sectional data:** A series of recorded data of different entities at same instance of time. For example, stock prices of different companies at the same instance of time.

c. **Panel data:** Panel data is a conjunction of time-series and cross-sectional data. It is a matrix of records of different entities at different instances of time where each row is a record of multiple entities at one time. Therefore, the rows are equidistant in time. In other words, it is a cross-sectional time series data. For example, stock prices of different companies at collected over a continuous period of time.

We will deal with cross-sectional time series data which essentially means for different units of time in data, we will observe multiple qualities of data that define the data. These features will later be used as estimators in our machine learning model.

4.5 Why perform time series analysis

We analyze time series data assuming that there might be a repetitive pattern between the successive values in the data which were measured at equally spaced time intervals. It helps us differentiate between patterns and noise. Major two ways to explore time series data which coexist in the same training data:

a. **Analysis of Trend:** As the name suggests, find components which change over time (either increase or decrease) and non-repetitive over the sample range.

b. **Seasonality Analysis:** Seasonality repeats after some intervals of time.

4.6 Time series modeling

In the past, statistical techniques were used to analyze time series data. We use machine learning to analyze time series now because it ensures high interpretability,

realistic modeling in some cases, and less biased prediction. Machine learning is able to look past trend and seasonality issues in the data and achieve true AI. This is achieved with the help of feature engineering. Time series modeling, unlike general modeling, is independent of the length of training data. If we train on more data, the model might show overfitting. We use time series modeling in anomaly detection. The most effective ways to perform time series modeling are:

a. **ARIMA**: As the name suggests, an Autoregressive Moving Average Model is an old statistical technique which is made up of two parameters: autoregressive and moving average. Along with stationary mean and variance, ARIMA is identified by stationary autocorrelation over time. This model is widely used in financial forecasting.

b. **LSTM:** Long Short-Term Memory models are a special form of recurrent neural networks. Most neural networks including RNNs conventionally work using a feed-forward network, which is why they are good in making decisions, but they are not as good in understanding the underlying context and make aware AI. LSTMs are called so, because of their ability to hold previous information to make aware decisions. The below figure is a simplified version of an LSTM.

Chapter 5

CONCLUSION

In this study, we have performed a comprehensive study of the network data stream. The packet stream has been observed in two ways, as set of fields as well as a pattern. As a result of this study, we have generated a detailed tree-like structure of features which represent a packet stream. Some of these features are directly observed in the stream, some are observed statistically from the stream and some are computed using detailed knowledge of the factors that may affect packets like entropy and compression.

With the help of this study, we will address the issue of data security which is the universal problem of understanding all kinds of data flows. The future work after this study is to use the features calculated in the above tree as guidance and obtain knowledge about good and bad packets, and normal and abnormal flow of data. The streams which present as abnormal from normal flows will be labeled as anomalous and their behavior will be recorded for future use to detect more anomalies.

To detect anomalous streams from the normal ones, we will approach these streams as time series datasets as packets arrive at a node sequentially in time. Regression analysis is one of the widely used ways to analyze such data. Another very effective way to analyze time series data is by using 1-Dimensional Convolutional Neural Networks. We use the power of neural network's perceptibility and its generality of algorithms to make intelligent systems which will be able to detect abnormal patterns from the normal ones.

REFERENCES

[1] Shahid Anwar, Jasni Mohamad Zain, Mohamad Fadli Zolkipli, Zakira Inayat, Suleman Khan, Bokolo Anthony and Victor Chang, From Intrusion Detection to an Intrusion Response System: Fundamentals, Requirements, and Future Directions, Algorithms, 2017

[2] Asmaa Shaker Ashoor and Sharad Gore. Intrusion Detection System (IDS): Case Study, International Conference on Advanced Materials Engineering, 2011

[3] Zhang, Y.; Lee, W. Intrusion detection in wireless ad-hoc networks. In Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston, MA, USA, 6–11 August 2000

[4] IPS vs. IDS, Similar on the Surface Polar Opposites Underneath. 503190-003 10/09 www.getadvanced.net

[5] Scarfone, K.; Mell, P. Guide to Intrusion Detection and Prevention Systems (IDPS); Report Number: 800-94; NIST Special Publication: Gaithersburg, MD, USA, 2007

[6] Avrim L. Bluma, Pat Langley. Selection of relevant features and examples in machine learning. Artificial Intelligence December 1997

[7] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B. Khalil, Deepak Turaga. Learning Feature Engineering for Classification, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)

[8] Hofstede, Rick; Čeleda, Pavel; Trammell, Brian; Drago, Idilio; Sadre, Ramin; Sperotto, Anna; Pras, Aiko (2014). "Flow Monitoring Explained: From Packet

Capture to Data Analysis with NetFlow and IPFIX". IEEE Communications Surveys & Tutorials

[9] Sambuddho Chakravarty, Marco V. Barbera, Georgios Portokalidis, Michalis Polychronakis, Angelos D. Keromytis. On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records, 2013

[10] Chandola V.; Banerjee A. und Kumar V. Anomaly Detection: A Survey. ACM computing survey (CSUR), 2009

[11] Julian Keppel and Sascha Schmalz. Real-time detection of anomalies in computer networks with methods of machine learning, 2017. https://www.inovex.de/blog/real-time-detection-of-anomalies-in-computer-networks-with-methods-of-machine-learning/

[12] H. Zenati, M. Romain, C. Foo, B. Lecouat and V. Chandrasekhar. "Adversarially Learned Anomaly Detection," 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 2018

[13] A. Radford, L. Metz, S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", International Conference on Learning Representations Workshop Track, 2016

[14] Mitchell, Tom. M. 1997. Machine Learning. New York: McGraw-Hill

[15] Witten, Ian H., and Eibe Frank. 2000. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. San Diego, CA: Morgan Kaufmann

[16] Cunningham, Padraig & Delany, Sarah. k-nearest neighbor classifiers. Multi Classification System, 2007

[17] Cover TM, Hart PE. “Nearest neighbor pattern classification”. IEEE Transaction on Information Theory 1967.

[18] Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries, 2003

[19] Berger, A & Lafferty, J. (1999). Information Retrieval as Statistical Translation. In Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIRí99)

[20] Salton, G. & Buckley, C. Term-weighing approaches in automatic text retrieval. In Information Processing & Management, 1988

[21] Stephen Robertson. Understanding Inverse Document Frequency: On theoretical arguments for IDF. Journal of Documentation, 2014

[22] Lukas Havrlanta and Vladik Kreinovich. A Simple Probabilistic Explanation of Term Frequency-Inverse Document Frequency Heuristic. International Journal of General Systems, 2014

[23] Manning, C.D.; Raghavan, P.; Schutze, H. "Scoring, term weighting, and the vector space model". Introduction to Information Retrieval, 2008

[24] Kenneth G. Paterson , Thomas Ristenpart , and Thomas Shrimpton. Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol, 2011

[25] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999. http://www.ietf.org/rfc/rfc2246.txt

[26] Artūrs Lavrenovs, HTTP Security Headers Analysis of Top One Million Websites. 10th International Conference on Cyber Conflict, 2018

[27] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J. Alex Halderman, Vern Paxson. The Security Impact of HTTPS Interception. NDSS Symposium, 2017

[28] Yichuan Tang. Deep Learning using Linear Support Vector Machines. International Conference on Machine Learning 2013: Challenges in Representation Learning Workshop. Atlanta, Georgia, USA

[29] Elnaggar, Ahmed. (2015). Secure Socket Layer. 10.13140/RG.2.1.2671.3044.

[30] K. Limthong and T. Tawsook, "Network traffic anomaly detection using machine learning approaches," *2012 IEEE Network Operations and Management Symposium*, Maui, HI, 2012

[31] Data streaming algorithms for estimating entropy of network traffic. .Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS/Performance 2006, Saint Malo, France, June 26-30, 2006

[32] https://quantra.quantinsti.com/glossary/Artificial-Neural-Network

[33] https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/

[34] https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/

[35] Source: https://www.cisco.com

[36] Halfond, W.G.; Viegas, J.; Orso, A. A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE International Symposium on Secure Software Engineering, Washington, DC, USA, 13–15 March 2006

[37] Michael J. De Lucia, Chase Cotton. Importance of Features in Adversarial Machine Learning for Cyber Security. 2018 Proceedings of the Conference on Information Systems Applied Research, Norfolk, Virginia

[38] J. Muehlstein et al., "Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application," 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2017, pp. 1-6