TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

REAL TIME NETWORK INTRUSION AND MALICIOUS URL DETECTION WITH NETWORK TRAFFIC VISUALIZATION

By:

Anuska Pant 072/BEX/405

Bibek Thapa Magar 072/BEX/409

Keshav Chaurasia 072/BEX/423

Anil Dhakal 072/BEX/451

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN ELECTRONICS & COMMUNICATION / COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

AUGUST, 2019

COPYRIGHT

ACKNOWLEDGEMENT

We have done this project as our major project of Electronics and Communication engineering. We express our deepest gratitude to the Department of Electronics and Computer Engineering, IOE, Pulchowk Campus for providing us with this opportunity to work on Major Project as part of our syllabus.

We are very grateful and honored to work under the supervision of Dr. Diakar Raj Pant who has been helpful and supportive in our project. We are thankful to our supervisor for guiding us the right path and for his encouragement and mentorship. We thank all our teachers for their invaluable suggestions which paved the way towards completion of the project.
We would also like to thank all the respective persons whose articles and videos have been so much helpful for us to catch on the path while we were stuck looking for the solution of different types of problems.

Last but not the least, we would like to thank our seniors and colleagues for helping us by providing us with necessary feedback and suggestions during the project.

Anuska Pant (072BEX405)
Keshav Chaurasia (072BEX423)
Anil Dhakal (072BEX451)
Bibek Thapa Magar (072BEX409)

ABSTRACT

The internet has become an important part of everyday life for information, communication and knowledge dissemination. But the internet is full of criminal activities and host a variety of unwanted contents ranging from spam, malicious scripts, phishing sites, ransomware, to dangerous attacks and exploits that infect and sensitive information. Therefore intrusion detection and safe internet browsing is one of the major concern in the task of network administration and security. Due to the lack of reliable test and validation data set, intrusion detection approaches are suffering from consistent and accurate performance evaluations. Also the system needs to be real time and accurate in detecting network intrusions and block potential malicious websites with very low false alarms. Thus neural network based real time network intrusion detection and malicious URL detection is been developed so that the attacks can be detected with greater accuracy and low false alarm rate. The system is trained in an simulated test bed environment by generating network attacks and training the system with the traffic data collected. The system detected malicious URL links with an accuracy of 96.44% and was able to classify attacks like Port Scan, Dos, Brute force and malware infiltration with accuracy of 93.38%. An accuracy of 79.82% was obtained upon evaluating the system with the data set from CICIDS2017. With the use of real time network intrusions detection and malicious URL detection, the user is protected from accidental malicious web browsing and attacks in the network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

METIS	Mobile and Wireless Communications Enablers for the Twenty-Twenty Information Society
DdoS	Distributed Denial of Service
IoT	Internet of things
5G	5th generation
IDS	Intrusion Detection System
DBN	Deep Belief Network
BM	Boltzmann Machine
RBM	Restricted Boltzmann Machine
DBM	Deep Boltzmann Machine
DNN	Deep Neural Network
DAE	Deep Auto-Encoder
SAE	Stacked Auto-Encoder
SDAE	Stacked De-noising Auto-Encoder
CNN	Convolutional Neural Network
URL	Uniform Resource Locater
TCP	Transfer Control Protocol
LAN	Local Area Network
WAN	Wide Area Network
RIP(protocol)	Routing Information Protocol
OSPF	Open Shortest Path First
BGP	Border Gateway Protocol
MAC	Media Access Control
OSI	Open Systems Interconnection
VPN	Virtual Private Network
SNMP	Simple Network Management Protocol
CLI	Command line interface
IGMP	Internet Group Management Protocol
RSTP	Rapid Spanning Tree Protocol
SPAN	Switched Port Analyzer
SFP	Sender Policy Framework

CGMP	Cisco group management protocol
IGMP	Internet Group Management Protocol
CAM	Content-addressable memory
VLAN	Virtual Local Area Network
TF-IDF	Term Frequency Inverse Document Frequency
CSV	Comma Separated Value
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
PHP	Predefined Hypertext Processor
ROC	Receiver Operating Characteristics
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface

1. INTRODUCTION

Computer networks and Internet are inseparable from human life today. Abundant applications rely on Internet, including life-critical applications in healthcare and military. Moreover, extravagant financial transactions exist over the Internet every day. This rapid growth of the Internet has led to a significant increase in wireless network traffic in recent years. According to a worldwide telecommunication consortium, Mobile and Wireless Communications Enablers for the Twenty-Twenty Information Society (METIS) [1], a proliferation of 5G and Wi-Fi networks is expected to occur in the next decades. By 2020[2] wireless network traffic is anticipated to account for two-thirds of total Internet traffic—with 66% of IP traffic expected to be generated by Wi-Fi and cellular devices only.

In a report published by IBM, common attacks noticed in report include brute-force, malvertising, phishing, SQL Injection, DDoS, malware, etc. Majority of malwares are accounted as ransomware (85% of malwares existed in a year are ransomware). These attacks possess high risk to sensitive data which could affect daily operation and can cause huge financial loss. The most popular companies impacted by security incidents are financial services-related companies, followed by information and communications, manufacture, retail, and healthcare [3].

According to latest research, the number of IoT devices globally is believed to have reached 10 billion in 2018. Note that this is higher than the number of mobile devices in use in the world. With 5G on the horizon and an uptick in IoT adoption, companies' plans to invest in IoT solutions seem to be accelerating. Estimates vary, of course, with another study pegging the 2018 figure at 7 billion devices and the 2025 estimate at 22 billion. Another set of internet of things statistics by Gartner predicts 25 billion IoT devices by 2021. We have also seen very high numbers being cited in some other studies (e.g., 200 billion according to Intel)[4]. The rapid spread of IoT devices has resulted in massive internet growth and billions of devices connected and online, which made the network prone to both passive and active attacks. Example of these attacks are impersonation, flooding and injection attacks. The wide and rapid spread of computing devices using Wi-Fi networks creates complex, large, and high-dimensional data, which cause confusion when capturing attack properties and force us to strengthen our security measures in our system.

Phishing and Social Engineering and other fraudulent attempts to obtain sensitive information such as username, password, and credit card details are increasing exponentially. Typically they are carried out by email spoofing and instant messaging. It often directs users to enter personal information at a fake website which matches the look and feel of the legitimate site. Similarly Malicious Web sites are a cornerstone of Internet criminal activities. They host a variety of unwanted content ranging from spam-advertised products, to phishing sites, to dangerous "drive-by" exploits that infect a visitor's machine with malware. As a result, there has been broad interest in developing systems to prevent the end user from visiting such sites. The most prominent existing approaches to the malicious URL problem are manually-constructed blacklists, as well as client-side systems that analyze the content or behavior of a Web site as it is visited.

Comprehensive researches have been executed to avoid attacks as mentioned earlier. An Intrusion Detection System (IDS) is one of the most common components of every network security infrastructure [5] including wireless networks [6]. In an IDS, many machine learning techniques has been adopted as primary detection algorithm. IDS can be classified into three types on the basis of detection method: misuse-, anomaly-, and specification-based IDS. A misuse-based IDS detects attack by checking the signature or pattern of attack in history.

IDS shows very promising results in detecting attacks however the security experts are still pursuing better performance with very high detection rate and lowest false alarm rate. In the recent years, improvement in IDS is achieved by embracing deep learning. Deep learning employs consecutive layer of information-processing stage in hierarchical manners for pattern classification and feature or representation learning.[7]. There are many deep learning methods such as Deep Belief Network (DBN), Boltzmann Machine (BM), Restricted Boltzmann Machine (RBM), Deep Boltzmann Machine (DBM), Deep Neural Network (DNN), auto-encoder, Deep Auto-Encoder (DAE), Stacked Auto-Encoder (SAE), Stacked De-noising Auto-Encoder (SDAE), distributed representation, and

Convolutional Neural Network (CNN). The advancements in learning algorithms might improve the performance of IDS to reach higher DR and lower false alarm rate.

There is a confusion of how to adopt deep learning in IDS applications properly since the different approaches have adopted by each previous one. Several researches use deep learning methods in a partial sense only, while the rest still uses conventional neural networks. The complexity of deep learning method may be one of the reasons. Also, deep learning method requires a lot of time to train correctly. However, we found that several researchers adopt deep learning method for feature learning and classification for an intelligent IDS in their network. We compare the IDS performance among them.

Deep learning is beneficial in IDS. This project examines such deep learning methods and apply deep learning to build real time intrusions detection system. At the end we, provide future challenges and directions to deploy real time IDS accordingly.

1.1. Intrusion Detection System

1.1.1. Definition

An IDS becomes a standard security measure in computer networks. Unlike Firewall (FW) in Figure 1.1a, IDS is usually located inside the network to monitor all internal traffics as shown in Figure 1.1b. One may consider using both FW and IDS to protect the network efficiently. IDS is defined autonomous process of intrusion detection which is to find events of violation of security policies or standard security practices in computer networks [8]. Besides identifying the security incidents, IDS also has other functions: documenting existing threats and deterring adversaries [9]. IDS requires particular properties which acts as a passive countermeasure, monitors whole or part of networks only, and aims high attack detection rate and low false alarm rate.

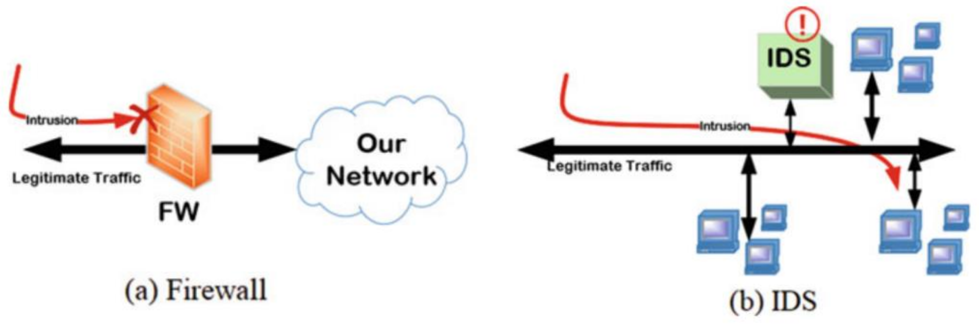


Figure 1.1: Typical network using (a) Firewall and (b) IDS

1.1.2 Classification of IDS

IDSs can be divided depending on the placement and the methodology deployed in a network. By the positioning of the IDS module in the network, we might distinguish IDSs into three classes: network-based, host-based, and hybrid IDSs. The first IDS, network-based IDS as shown in Figure 1.2a, puts the IDS module inside the network where whole can be monitored. This IDS checks for malicious activities by inspecting all packets moving across the network. On the other hand, Figure 1.2b shows the host-based IDS which places the IDS module on each client of the network. The module examining all inbound and outbound traffics of the corresponding client leads to detailed monitoring of the particular client. Two types of IDSs have specific drawbacks—the network-based IDS might burden the workload and then miss some malicious activities, while the host-based IDS does not monitor all the network traffics, having less workload than the network-based IDS. Therefore, the hybrid IDS as shown in Figure 1.2c places IDS modules in the network as well as clients to monitor both specific clients and network activities at the same time.

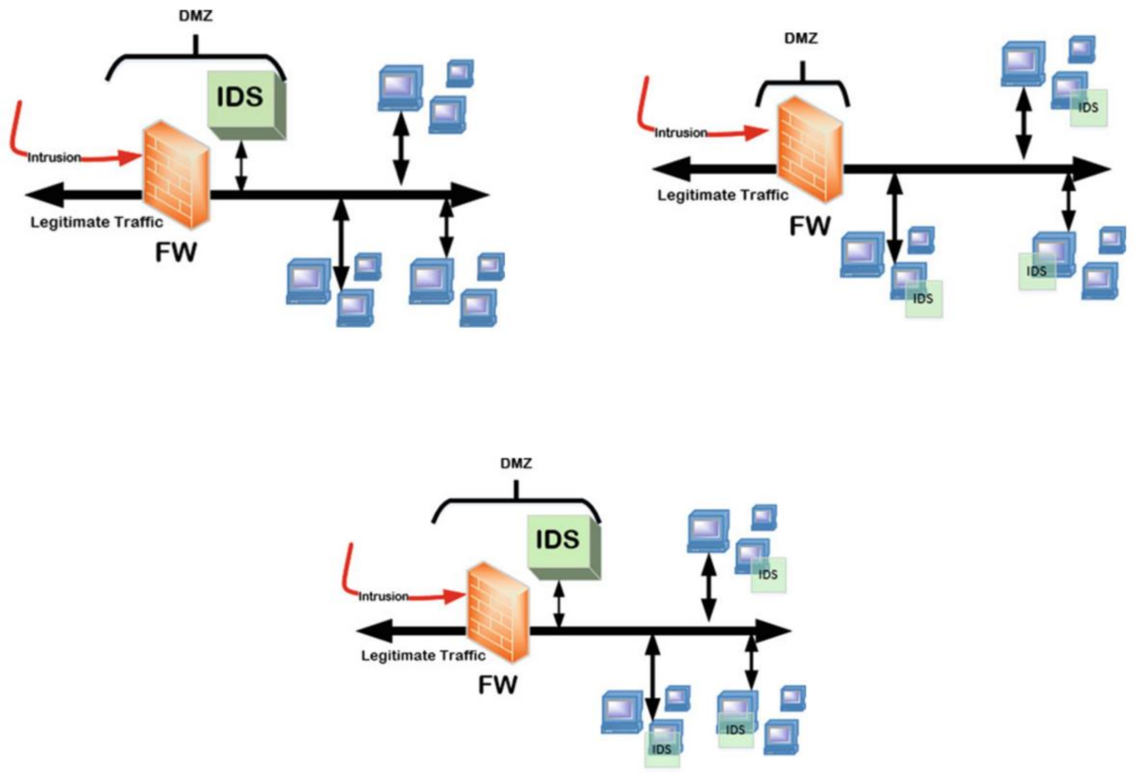


Figure 1.2: (a) on top left Network-based IDS (b) on top right Host-based IDS (c) on bottom center Hybrid-based IDS

Different types of Intrusion Detection has different advantages and disadvantages. They need to be used accordingly upon the requirements. Table 1.1 shows the categorization of IDS and their specifications.

	Misuse-based	Anomaly-based	Specification-based
Method	Identify known attack patterns	Identify unusual activity patterns	Identify violation of predefined rules
DR	High	Low	High
False alarm rate	Low	High	Low
Unknown attack detection	Incapable	Capable	Incapable
Drawback	Updating signatures is burdensome	Computing any machine learning is heavy	Relying on expert knowledge during defining rules is undesirable

Table 1.1: Classification of IDS and their specification

Machine Learning in IDS can be either supervised or unsupervised. In case of supervised machine learning detection rate of known attacks are much higher than the detection of unknown attack because the system is trained with the known attacks. However this is not the case in unsupervised learning. In unsupervised, since the model is trained to look for the unknown attacks and anomalies, it has much higher detection rate of unknown attacks.

	Supervised	Unsupervised
Definition	The dataset are labeled with predefined classes	The dataset are labeled **without** predefined classes
Approach	Classification	Clustering
Method	Support Vector Machine, Decision Tree, etc.	K-means clustering, Ant Clustering Algorithm etc.
Known attack detection	High	Low
Unknown attack detection	Low	High

Table 1.2: Classification of IDS in terms of machine learning

1.2. Malicious URL Detection System

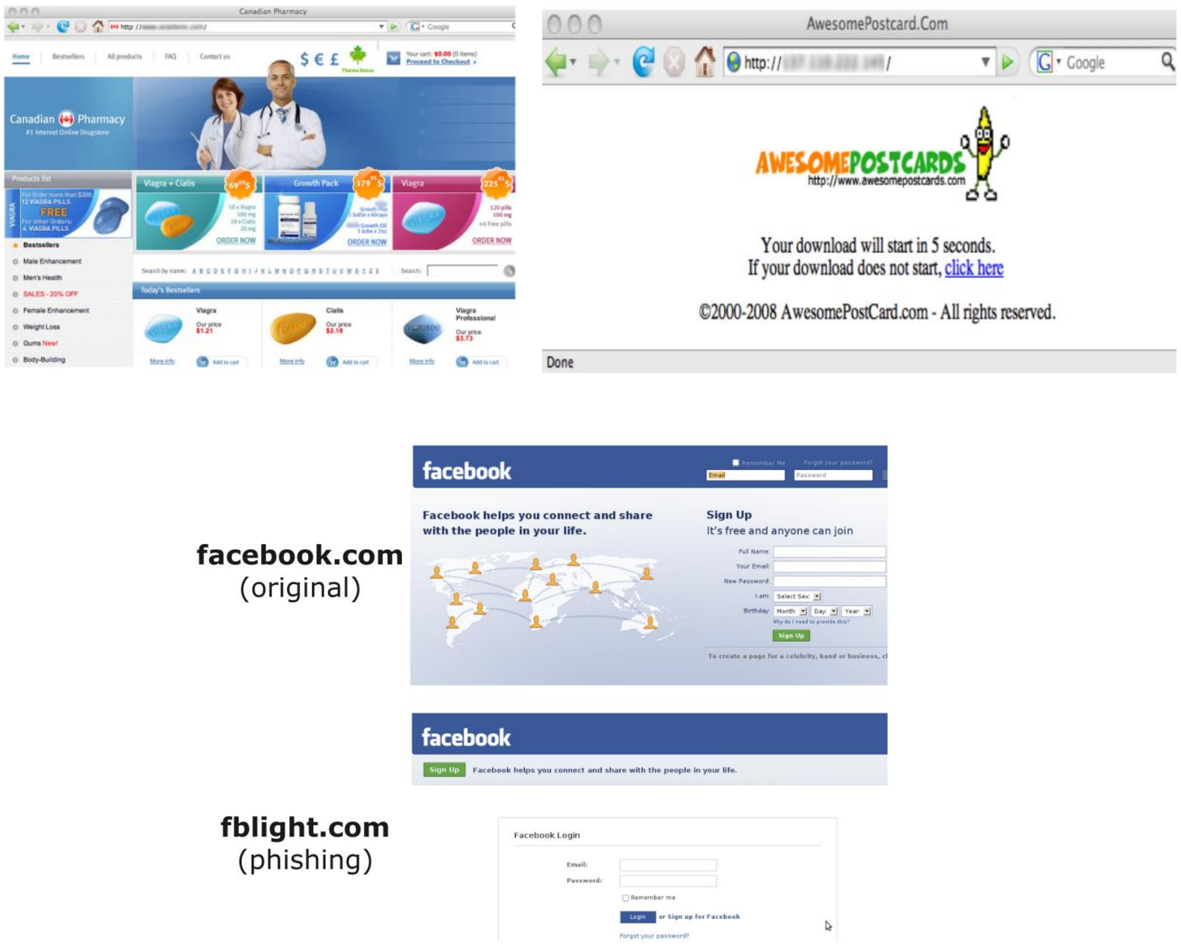
1.2.1. Definition

The human understandable URLs are used to identify billions of websites hosted over the present day internet. Adversaries who try to get unauthorized access to the confidential data may use malicious URLs and present it as a legitimate URL to naive user. Such URLs that act as a gateway for the unsolicited activities are called as malicious URLs. These malicious URLs can cause unethical activities such as theft of private and confidential data, ransomware installation on the user devices that result in huge loss every year globally. Even security agencies are cautious about the malicious URLs as they have the potential to compromise sensitive and confidential data of government and private organizations. With the advancement of social networking platforms, many allow its users to publish the unauthorized URLs. Many of these URLs are related to the promotion of business and self-advertisement, but some of these unprecedented resource locators can pose a vulnerable threat to the naive users.

The naive users who use the malicious URLs, are going to face serious security threats initiated by the adversary. The verification of URLs is very essential in order to ensure that user should be prevented from visiting malicious websites. Many mechanisms have been proposed to detect the malicious URLs. One of the basic feature that a mechanism should possess is to allow the benign URLs that are requested by the client and prevent the malicious URLs before reaching the user. [10]

There are four major categories of malicious sites, and each category is distinguished by the level of interaction required by the user, as well as the motivation of the site owners.

Illicit Products: These sites typically sell illicit goods such as pharmaceuticals (often counterfeit), counterfeit watches and pirated software. Because of the criminal nature of these enterprises, these sites typically rely on URLs in spam advertising or artificially-optimized Web search results to attract traffic. Figure 1.3(a) shows an example of a pharmacy campaign selling male enhancement drugs.



Your download will start in 5 seconds.
If your download does not start, click here

©2000-2008 AwesomePostCard.com - All rights reserved.

Facebook helps you connect and share with the people in your life.

Figure 1.3: Examples of three categories of malicious Web sites: (a) illicit products, (b) phishing, and (c) trojan downloads.

Phishing: Phishing sites use mimicry to trick users into divulging private information such as passwords and account numbers. First, phishers send email messages to unsuspecting users that prompt the users to visit the phishing site's URL. Users are then asked to enter their sensitive information on the site, which is intentionally crafted to resemble a legitimate site (e.g., their bank, email or social networking account page). A common deception phishers use is to embed tokens from the legitimate target site into their own phishing URL. After the criminals acquire the user's credentials, they may use them to withdraw money, launder money, send spam under a new guise, as well as commit other nefarious acts. We present a phishing example in Figure 1.3(b), which compares the legitimate login page for Facebook with the illegitimate login page for the phishing site fblight.com.

Trojan Downloads: These sites encourage users to download and execute a malware binary that purports to be benign. Figure 1.3(c) shows an example of a site that encourages users to download an electronic postcard. Other examples of trojan downloads can include sites hosting screensavers, codec plugins for video players, and other seemingly mundane programs. Criminal organizations construct these kinds of sites to infect new hosts and expand their networks of compromised hosts (known as botnets).

Drive-By Exploits: Drive-by exploits are scripts or objects embedded in a Web page that exploit a vulnerability in the user's browser. Once the browser is compromised, the exploit code causes the user to involuntarily download malicious code. Examples of drive-by exploits include maliciously-crafted JavaScript, Flash objects and images. Criminals may embed the exploits into stand-alone pages, but often the exploits are also loaded into legitimate sites via third-party advertising. Although we do not directly address the threat of drive-by exploits loaded as advertisements (this would require analyzing content), our approach could still address this problem by letting legitimate servers determine the safety of the advertisement URLs before including such third-party content.

2. PROBLEM STATEMENT

Malicious activities occurring in the network can cause severe damage to the underlying network infrastructure. Measurement of the traffic from and to the base is required to understand the network behavior. Monitoring the traffic is essential to save the users from unwanted problems and save the resources from getting saturated unnecessarily. Moreover, data needs to be kept confidential and integrity of data also needs to be preserved. Otherwise, anybody could enter into the system and can exploit the resources. It is well known that NIDS suffers from the high rate of false alarms. Continuous efforts are being made to reduce the high false positive rate. We believe that intrusion detection is a data analysis process and can be studied as a problem of classifying data correctly. From this standpoint, it can also be observed that any classification scheme is as good as the data presented to it as input. More clean the data, higher accurate results are likely to be obtained. From NIDS point of view, it implies that if we can extract features that demarcate normal data from abnormal one properly, false positive rate can be reduced to a great extent. Therefore, in this work, we investigate the techniques which facilitate in the process of demarcating normal data from abnormal ones.

On the similar lines, we observe that most of the data mining and machine learning based methods in intrusion detection make use of well-known tools and techniques. It may turn out that these general techniques are not very effective in classifying data as normal or abnormal with very high accuracy. There is a need to customize those techniques according to the requirement of intrusion detection. We also focus on this aspect in our work. Apart from the problems mentioned above, the fast detection of attacks remains one of the focal points to be worried about. With the present complexity and variety of attacks, we need a huge amount of data to analyses and produce results. But larger the amount of data, longer the time to analyses it, which delays the detection of attacks. An IDS will be of more use if it can trigger an alarm early enough to reduce the damage that an ongoing attack can do. Thus, there is a need to make IDS as fast as to operate on-line. We believe that this can be achieved if we can reduce the data, to be analyzed, without degrading its quality.

3. MOTIVATION

Computer networking is a developing technology that is gaining widespread acceptance and popularity in the commercial sector as a result of standardized protocols and specifications. To truly secure a network a second line of defense is also need: NIDS that can detect a third party that tries to exploit the security of the network, even if this attack has not been experienced before. If the intruder is detected soon enough, it can take any appropriate measures before any damage is done or any data is compromised. Thus intrusion detection presents a second wall of defense and it is a necessity in a high survivability network. Also, gaining hands on experience while working on developing a NIDS will broaden the horizon of our knowledge in the field of networking and machine learning.

4. OBJECTIVE

1. Generating our own data set of network traffic and malicious attacks in a test bed environment
2. Building a real time network intrusion detection system and malicious URL detection
3. Building a Network Reporting and Visualization Dashboard

5. LITERATURE REVIEW

ANNs have been used in a wide variety of classification tasks across many application domains. In contrast to traditional classification methods, such as logistic regression and discriminant analysis, which require a good understanding of the underlying assumptions of the probability model of the system that produced the data, ANNs are a “black-box” technique capable of adapting to the underlying system model [11]. This makes them particularly useful in fields such as decision support for concealed weapons detection [12], prediction and classification of Internet traffic [13], and signature verification [11] where their ability to adapt to the data, especially in high dimensional datasets, overcomes many of the difficulties in model building associated with conventional classification techniques such as decision trees and k-nearest neighbor algorithms [14].

ANNs have also been used in several computer security domains, including the analysis of software design flaws [15] and computer virus detection [16]. ANN approaches to detection of multiple types of network attacks have also been shown to be effective [17], though their application to the detection of shellcode was not considered.

According to the survey conducted in “Survey on Malicious Web Pages Detection Technique”, the authors mention that the web attacks are on a rise. In the year 2012, each month noted a total of approximately 33,000 phishing attacks, which summed to a loss of $687 million. With such a hike in the number of web attacks, there is a dire need of a system which prevents such types of webpage attacks. The user goes to a website unknowingly which might not be safe for the user and ends up either giving his credentials to other intruders and hackers or downloading malicious data. So to prevent this type of attacks, a tool is needed which examines the URL entered by the user and checks if the website is malicious or not.

In the paper "Identifying Vulnerable Websites by Analysis of Common Strings in Phishing URLs" [18], the authors mention about the rise in the phishing websites and the method they used to detect and prevent it. For the detection of a safe or a phishing webpage, they implemented Largest Common Substring (LCS) method. They prepared a database of phishing websites of their own and tested the LCS on the new website. In the paper "On URL Classification", the authors mention about how URLs can be used to use the victim's computer resources for different attacks like phishing, denial of service. Also, they compared various methodologies including traditional ones and the recent trends in the field of machine learning. The results show that machine learning techniques are better for detection [19]. So instead of using traditional techniques, we plan to integrate the concept of machine learning in our tool.

In the paper "Malicious URL Detection using Convolutional Neural Network" [2], the authors Farhan Douksieh Abdi, and Lian Wenjuan used simple CNN algorithm to detect and predict URLS and compared it with two other algorithms to know (Support Vector Machine and Logistic Regression). In the authors proposed design the URL is the input to the Database. Then, when the URL is input to Blacklist, it has two cases: First, in case where the URL already exists in our blacklist, the URL will be qualified as malicious. Second, the Feature Extraction of the URL is extracted for the analysis. The Feature extraction is done by using Word2Vec vectorizer. The outputs of the classifier is malicious or benign. According to the paper, they obtained the accuracy of 96% in detecting malicious URL.

6. METHODOLOGY

Real Time Network Intrusion Detection System and network traffic visualization comprises of many modules and are implemented independently. These modules work separately parsing data received from other modules and processing it in the required way. The system is developed in 5 modular steps

- Setting up testbed architecture for IDS
- Data Set Generation for Normal Traffic and Attack Traffic
- Real Time Malicious URL Detection
- Real Time Network Intrusion Detection
- Network Traffic Reporting and Visualization

6.1. Setting up testbed architecture for IDS

6.1.1. Network and Networking Fundamentals

Before discussing the actual detection and network intrusion detection, we must provide the fundamental concepts on networks, network traffics, and traffic visualization. A network is an interconnection of a set of devices, hosts, nodes, and end-computing devices that can communicate among themselves. For successful communication, it is necessary to follow certain rules to ensure efficient communication among the interconnected devices. Such a set of rules is known as a protocol. A modern data communication system needs many such protocols to communicate with each other over heterogeneous networks. This interconnection of one device to many other devices is known as networking.

A network architecture is a blueprint of the complete computer communication network, which provides a framework and technology foundation for designing, building and managing a communication network. It typically has a layered structure. Layering is a modern network design principle which divides the communication tasks into a number of smaller parts, each part accomplishing a particular sub-task and interacting with the other parts in a small number of well-defined ways. Layering allows the parts of a communication to be designed and tested without a combinatorial explosion of cases, keeping each design relatively simple.

If a network architecture diagram is open, no single vendor owns the technology and controls its definition and development. Anyone is free to design hardware and software based on the network architecture. The TCP/IP network architecture, which the Internet is based on, is such an open network architecture and it is adopted as a worldwide network standard and widely deployed in local area network (LAN), wide area network (WAN), small and large enterprises, and last but not least, the Internet. A typical network architecture is shown in figure below.

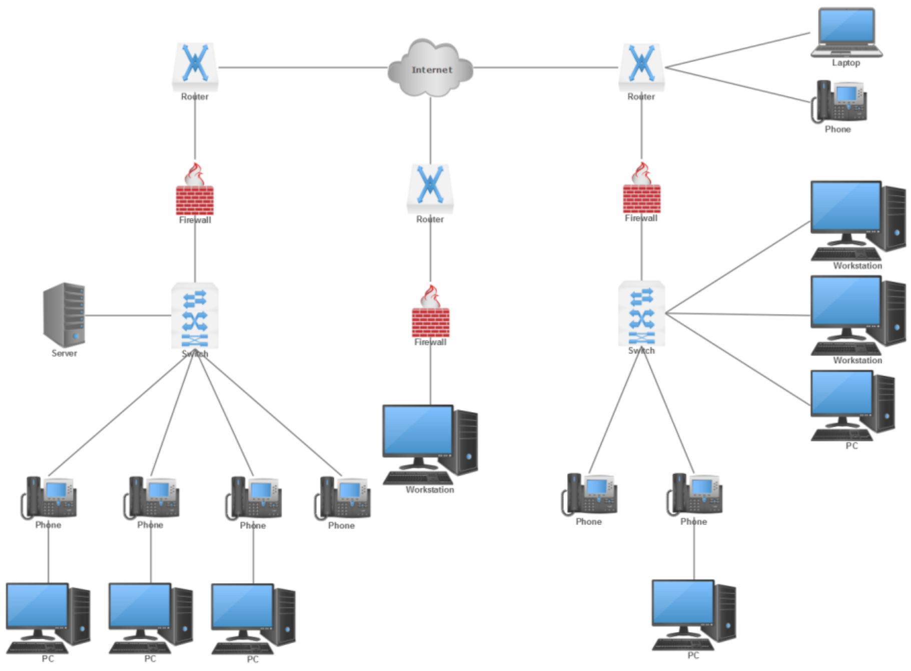


Figure 6.1: an example of a network architecture

A network contains both hardware and software components to communicate between stations. Software components are used to share files, programs, printers, and operating systems. However the primary focus of computer network lies on its hardware components which enables the PC to connect to other devices. Network interface card (NIC), transmission media, client devices, servers, network connecting devices are some of the essential hardware components needed to run a network.

6.1.2. Prototype Testbed Architecture for IDS

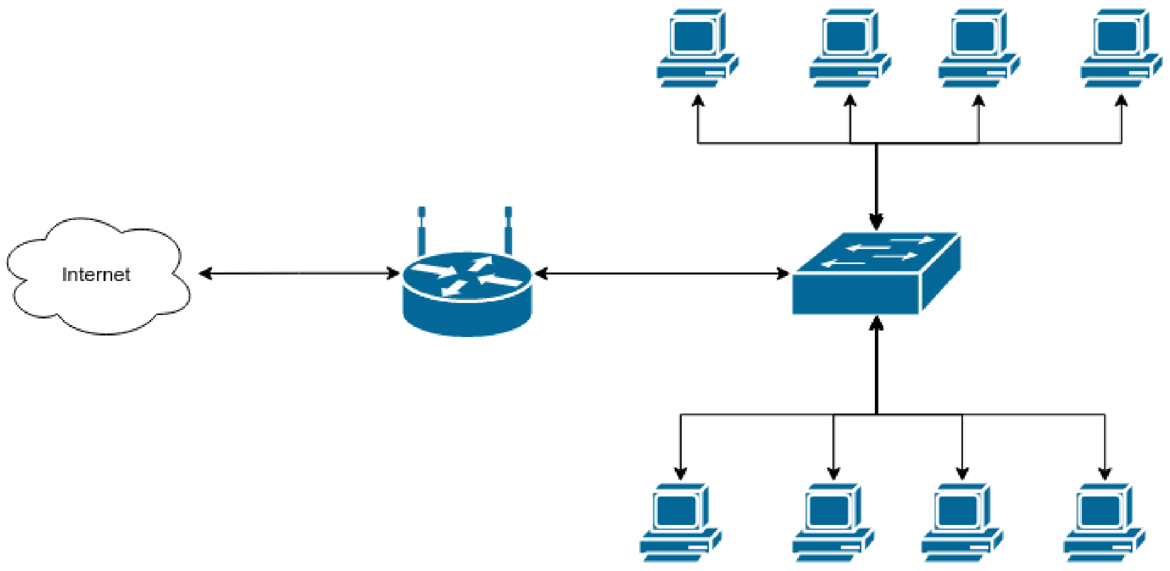


Figure 6.2: Testbed architecture for prototype IDS

Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs) are the most important defense tools against the sophisticated and ever-growing network attacks. It secures the network by forewarning them about malicious behavior such as intrusion, attacks and malware infiltrations. However the cost of setting up real-life IDS and IPS is very high and requires lots of network devices and network assistant. Figure 6.2 shows the prototype testbed architecture of our simulated environment. All the data set generation and real time IDS setup are performed in this prototype. This architecture consists of minimal networking devices that are required to create life-like network environment. Some of the major network hardware devices are explained below.

6.1.2.1. Router

A router is a networking device that forwards data packets between computer networks. Routers perform the traffic directing functions on the Internet. Data sent through the internet, such as a web page or email, is in the form of data packets. A packet is typically forwarded from one router to another router through the networks that constitute an internetwork (e.g. the Internet) until it reaches its destination node [21].

A router is connected to two or more data lines from different networks. When a data packet comes in on one of the lines, the router reads the network address information in the packet to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey.

Routers are at least two kinds: internet routers administered by your internet service provider (ISP) and home router to connect your LAN to the internet and hopefully protect you from most attacks. Home-Router are also often called gateways, because they manage the interaction of a network with another. They receive all packets from internal hosts that should be send to some computer on the internet, write their own public IP address received from the ISP as source address into it and forwards them to the next router of the

ISP. Internet routers also forward packets, but they do so by depending on a more or less huge routing table. They don't have a static routing table but use different protocols like RIP, OSPF and BGP to share routing information between each other and find the shortest or otherwise quickest way to the desired destination.

In the project we used simple DSL router that connected the multiple endpoints with the internet.

6.1.2.2. Switch

A network switch (also called switching hub, bridging hub, officially MAC bridge) is a computer networking device that connects devices on a computer network by using packet switching to receive, process, and forward data to the destination device.

A network switch is a multiport network bridge that uses hardware addresses to process and forward data at the data link layer (layer 2) of the OSI model. Some switches can also process data at the network layer (layer 3) by additionally incorporating routing functionality. Such switches are commonly known as layer-3 switches or multilayer switches [22].

With regard to management options, the three primary classes of switches are Unmanaged Switches, Managed Switches, and Web Smart switches.

Web Smart Switches: Web smart switches—sometimes called smart switches or Web managed switches—have become a popular option for mid-sized networks that require management. They offer access to switch management features such as port monitoring, link aggregation, and VPN through a simple Web interface via an embedded Web browser. What these switches generally do not have is SNMP management capabilities or a CLI. Web-smart switches must usually be managed individually rather than in groups.

Although the management features found in a Web-smart switch are less extensive than those found in a fully managed switch, these switches are becoming smarter, now offering many of the features of a fully managed switch.

Unmanaged Ethernet Switches: Unmanaged switches are basic plug-and-play switches with no remote configuration, management, or monitoring options, although many can be locally monitored and configured via LED indicators and DIP switches. These inexpensive switches are typically used in small networks or to add temporary workgroups to larger networks.

Unmanaged switches are plug and play devices without the need of a complex setup. These switches allow Ethernet devices to communicate with one another (such as a PC or network printer) by providing a connection to the network and passing on information to where it needs to go. They come with a fixed configuration and do not allow any changes to this configuration, therefore, there is no need for prioritizing the packets. They can be desktop or rack mounted and are a less expensive option for connectivity switching needs.

One major setback within an industrial setting is that they don’t support IGMP and treat multicast traffic in the same fashion as broadcast traffic, which can overwhelm a network when uncontrolled and lead to broadcast storms. This feature is particularly important on industrial networks because the protocol Ethernet/IP relies heavily on multicast traffic.

Unmanaged switches are mostly used to connect edge devices on network spurs, or on a small stand-alone network with only a few components. It's suitable for any business network that wants to simplify the installation of wireless access points and IP-based surveillance cameras. They are also applicable for home use, SOHO, small businesses or to add temporary workgroups to larger networks.

Managed Ethernet Switches: Managed Switches are oriented towards redundancy as they provide Port Mirroring or SPAN which allows to monitor ports and capture traffic flows, QoS (Quality of Service) which prioritizes bandwidth for data subsets, allowing more bandwidth to be allocated through the network to ensure IP data comes in smoothly, obtaining the sensor data without an interruption using minimal bandwidth. Managed switches support Simple Network Management Protocol (SNMP) via embedded agents and have a command line interface (CLI) that can be accessed via serial console, Telnet, and Secure Shell. These switches can often be configured and managed as groups. Managed switches also provide additional protocols like RSTP (Rapid Spanning Tree Protocol) allowing alternate cabling paths and preventing loop situations which usually are the reason for network malfunctions. A managed switch offers redundancy capabilities which helps to reduce unplanned downtime. Managed switches also offer powerful features like VLANs, LACP, as well as all advanced filter and multicast algorithms needed today to easily priorities, partition and organize a reliable high-speed network. The conveniences of purchasing a managed switch is that if you later desire to change the configuration you can still do so and start vlanning the section of your network you'd wish for your specific networking needs.

Managed switches are aimed at users that require a response time of milliseconds. They are especially suitable for organizations that need to manage and troubleshoot their network remotely and securely; allowing network managers to reach optimal network performance and reliability.

Managed switches should be used on any network or segment of a network where the traffic has to be monitored and controlled. Managed network switches enable complete control of data, bandwidth and traffic control. Some switches come equipped with SFP slots, which allow the network to be expanded with flexibility. The benefit of having multi-rate SFP slots is to be able to use 100Mbps and 1Gbps SFP Modules for either multi-mode or single-mode in a mix and match as needed. If requirements change, it´s possible to replace the SFP module and protect your switch investment.

In our network we used Cisco Catalyst 1900 series managed switch. The purpose of using this switch was to use the feature SPAN (Switched Port Analyzer). It mirrors the packets incoming and outgoing through different ports to a specific port.

The SPAN feature was introduced on switches because of a fundamental difference that switches have with hubs. When a hub receives a packet on one port, the hub sends out a copy of that packet on all ports except on the one where the hub received the packet. After a switch boots, it starts to build up a Layer 2 forwarding table on the basis of the source MAC address of the different packets that the switch receives. After this forwarding table is built, the switch forwards traffic that is destined for a MAC address directly to the corresponding port.

For example, if we want to capture Ethernet traffic that is sent by host A to host B, and both are connected to a hub, just attach a sniffer to this hub. All other ports see the traffic between hosts A and B

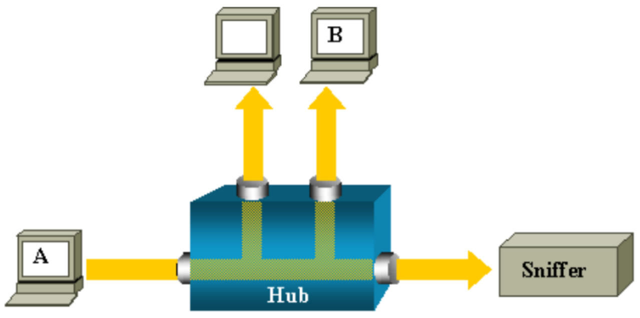


Figure 6.3: Working principle of packet flow in a hub device

On a switch, after the host B MAC address is learned, unicast traffic from A to B is only forwarded to the B port. Therefore, the sniffer does not see this traffic:

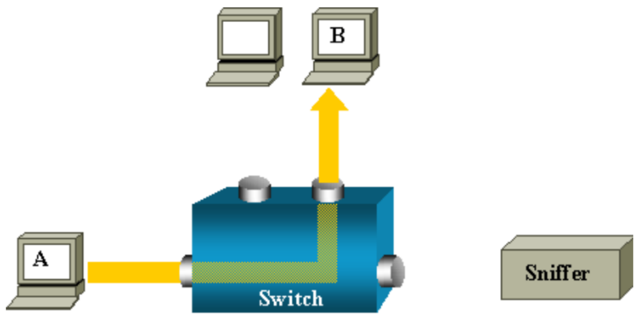


Figure 6.4: Working principle of packet flow in an unmanaged switch device

In this configuration, the sniffer only captures traffic that is flooded to all ports, such as:

- Broadcast traffic
- Multicast traffic with CGMP or Internet Group Management Protocol (IGMP) snooping disabled
- Unknown unicast traffic

Unicast flooding occurs when the switch does not have the destination MAC in its content-addressable memory (CAM) table. The switch does not know where to send the traffic. The switch floods the packets to all the ports in the destination VLAN.

An extra feature is necessary that artificially copies unicast packets that host A sends to the sniffer port:

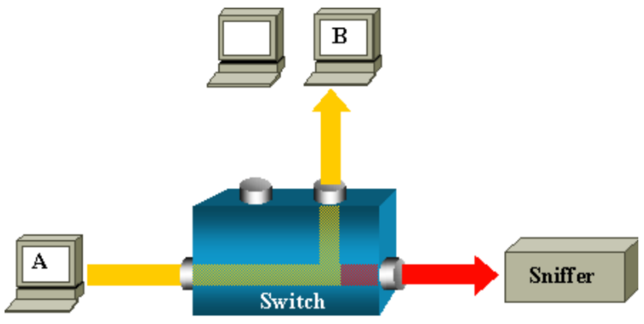


Figure 6.5: Working principle of packet flow in a managed switch device with SPAN configured

In this diagram, the sniffer is attached to a port that is configured to receive a copy of every packet that host A sends. This port is called a SPAN port. The other sections of this document describe how you can tune this feature very precisely in order to do more than just monitor a port.

6.1.2.3. Network Host Devices

A network host is a computer connected to a computer network. A host may work as a server offering information resources, services, and applications to users or other nodes on the network. Hosts are assigned at least one network address.

A computer participating in networks that use the Internet protocol suite may also be called an IP host. Specifically, computers participating in the Internet are called Internet hosts, sometimes Internet nodes. Internet hosts and other IP hosts have one or more IP addresses assigned to their network interfaces. The addresses are configured either manually by an administrator, automatically at startup by means of the Dynamic Host Configuration Protocol (DHCP), or by stateless address auto configuration methods.

Network hosts that participate in applications that use the client-server model of computing, are classified as server or client systems. Network hosts may also function as nodes in peer-to-peer applications, in which all nodes share and consume resources in an equipotent manner.

6.1.2.4. Configurations in the environment

In this simple testbed we designed and implemented two networks, namely Attack Network and Victim-Network. The Victim-Network is a normally configured devices with their default firewall turned on. The router is connected to the internet that was provided by the Internet Service Provider (ISP). On one port of LAN ports the switch is connected. The Attacker-Network has two hosts connected directly to the router while the Victim-Network is set-up in the switch. All the victim hosts are connected to the switch. Setting up the

Attacker Network to the other network was very expensive and near to impossible with the devices that were available to us.

6.1.2.5. IP Configurations of the Network Devices

Device	IP address	Subnet Mask	Gateway	Description
Router	192.168.10.1	255.255.255.0	100.76.0.1	Connects to the internet provided by the ISP
Managed Switch	192.168.10.2	255.255.255.0	192.168.10.1	Connects to the router
Attacker host 1	192.168.10.6	255.255.255.0	192.168.10.1	Kali Linux OS
Attacker host 2	192.168.10.10	255.255.255.0	192.168.10.1	Ubuntu 18 OS
Victim host 1	192.168.10.9	255.255.255.0	192.168.10.1	Windows 10 OS
Victim host 2	192.168.10.7	255.255.255.0	192.168.10.1	Windows 10 OS
Victim host 3	192.168.10.15	255.255.255.0	192.168.10.1	Windows 10 OS
Victim host 4	192.168.10.14	255.255.255.0	192.168.10.1	Android 7.0
Victim host 5	192.168.10.13	255.255.255.0	192.168.10.1	Android 5.0
Victim host 6	192.168.10.11	255.255.255.0	192.168.10.1	Windows 10 OS
Victim host 7	192.168.10.2	255.255.255.0	192.168.10.1	Windows 10 OS
Victim host 8	192.168.10.112	255.255.255.0	192.168.10.1	Windows 10 OS
IDS	192.168.10.100	255.255.255.0	192.168.10.1	Ubuntu 18.04

Table 6.1: IP Configuration of Network Devices and host in a testbed architecture

6.1.2.6. SPAN Configuration in Managed Switch

The whole purpose of using managed switch was to monitor the main uplink port and also all other ports. Monitoring these ports will duplicate all the packets sent and received by the victim hosts and send it to the flow collector which has intrusion detection system. The flow collector will continuously collect the packet and intrusion detection system will analyze each and every packet to detect intrusions, attacks and malware infiltrations.

In our project port 1 of switch is connected to the router. This port acts as the main uplink through which all the packets are sent and received by victim hosts. Port 8 is configured to receive all the packets from port 1 to port 12 except itself. For the configuration it requires to access the switch command line interface using serial communication between the computer and the switch. The IP address, the gateway address and SPAN configuration is done only after accessing the command line interface. After the configuration the switch is connected back. This completes the complete physical hardware setup required to demonstrate real time network intrusion detection.

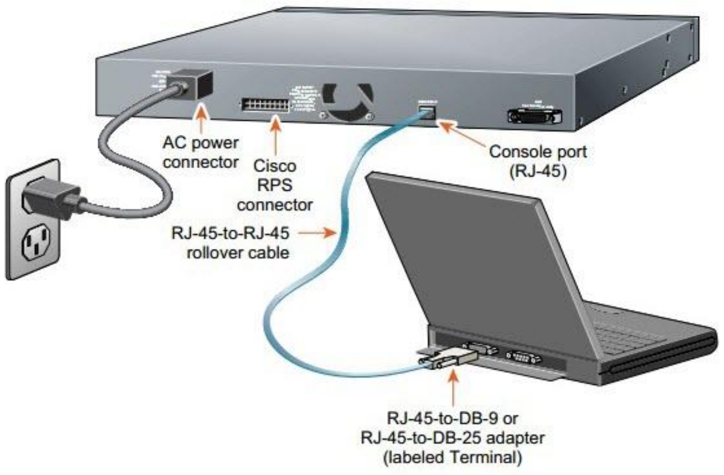

AC power connector
Cisco RPS connector
Console port (RJ-45)
RJ-45-to-RJ-45 rollover cable
RJ-45-to-DB-9 or RJ-45-to-DB-25 adapter (labeled Terminal)

Figure 6.6: Physical connection for SPAN configuration in Cisco catalyst 1900 series

6.2. Data set generation for Network Intrusion Detection System

6.2.1. Available data set

Datasets play an important role in the testing and validation of network anomaly detection methods or Systems. A good quality dataset not only allows us to identify the ability of a method or a system to detect anomalous behavior, but also allows us to provide potential effectiveness when deployed in real operating environments. Several datasets are publicly available for testing and evaluation of network anomaly detection methods and systems.
A taxonomy of network intrusion datasets is shown in following figure [23]:

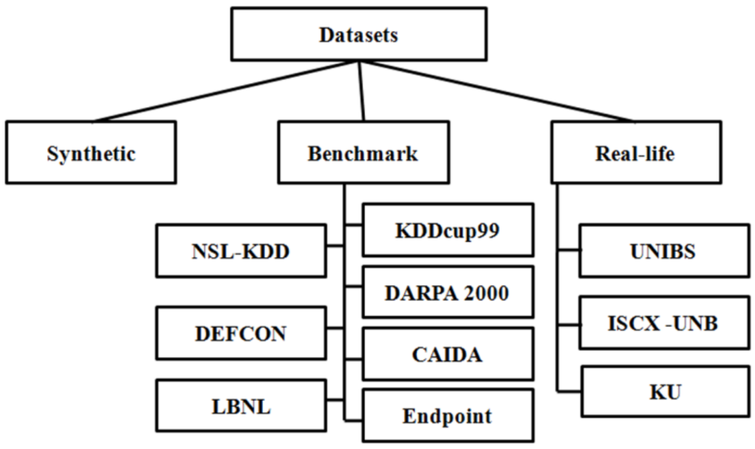


Figure 6.7: Taxonomy of NIDS dataset

6.2.1.1. Synthetic Datasets

Synthetic datasets are generated to meet specific needs or certain conditions or tests that real data satisfy. Such datasets are useful when designing any prototype system for theoretical analysis so that the design can be refined. A synthetic dataset can be used to test and create many different types of test scenarios. This enables designers to build realistic behavior profiles for normal users and attackers based on the dataset to test a proposed system. This provides initial validation of a specific method or a system; if the results prove to be satisfactory, the developers then continue to evaluate a method or a system in a specific domain real-life data

6.2.1.2. Benchmark Datasets

Seven publicly available benchmark datasets generated using simulated environments in large networks were discussed. Different attack scenarios were simulated during the generation of these datasets.

KDD Cup 1999 Dataset: The KDD Cup 1999 dataset is the benchmark dataset for intrusion detection. This dataset is an updated version of the DARPA98, by processing the tcpdump portion the benign and attack traffic are merged together in a simulated environment. This dataset has a large number of redundant records and is studded by data corruptions that led to skewed testing results (Tavallaee et al., 2009). NSL-KDD was created using KDD (Tavallaee et al., 2009) to address some of the KDD's shortcomings (McHugh, 2000).

Each record of the dataset represents a connection between two network hosts according to existing network protocol and is described by 41 attributes (38 continuous or discrete numerical attributes and 3 categorical attributes). Each record of the training data is labeled as either normal or a specific kind of attack. [24]. Denial of Service(DoS), User to Root (U2R), Remote to Local(R2L) and Probes are the attacks found in the KDD Cup 1999 Dataset.

NSL-KDD Dataset: NSL-KDD is a dataset for network-based intrusion detection systems. It is the new version of KDD Cup 1999 dataset. In the KDD Cup dataset, there are a large number of redundant records, which can cause learning algorithms to be biased towards frequent records. To address this issue, one unique copy of each record is kept in the NSL-KDD dataset. Though this dataset is not the perfect representation of real networks, it can be applied as an effective benchmark dataset to compare different intrusion detection methods. Analysis of the KDD dataset showed that there were two important issues with the dataset, which highly affect the performance of evaluated systems often resulting in poor evaluation of anomaly detection methods. To address these issues, a new dataset known as NSL-KDD, consisting of selected records of the complete KDD dataset was introduced. This dataset is also publicly available for researchers and has the following advantages over the original KDD dataset.

- This dataset doesn't contain superfluous and repeated records in the training set, so classifiers or detection methods will not be biased towards more frequent records.
- There are no duplicate records in the test set. Therefore, the performance of learners is not biased by the methods which have better detection rates on frequent records.
- The number of selected records from each difficulty level is inversely proportional to the percentage of records in the original KDD dataset. As a result, the classification rates of various machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of various learning techniques.
- The number of records in the training and testing sets is reasonable, which makes it practical to run experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research groups are consistent and comparable.

DARPA 2000 Dataset: A DARPA project targeted the detection of complex attacks that contain multiple steps. Two attack scenarios were simulated in the DARPA 2000 evaluation contest, namely Lincoln Laboratory scenario DDoS (LLDOS) 1.0 and LLDOS 2.0. To achieve variations, these two attack scenarios were carried out over several network and audit scenarios.

These sessions were grouped into four attack phases:

- Probing
- Breaking into the system by exploiting vulnerability
- Installing DDoS software for the compromised system
- Launching DDoS attack against another target.

DEFCON Dataset: The DEFCON dataset is another commonly used dataset for evaluation of IDSs. It contains network traffic captured during a hacker competition called

Capture The Flag (CTF), in which competing teams were divided into two groups: attackers and defenders. The traffic produced during CTF is very different from real world network traffic since it contains only intrusive traffic without any normal background traffic. Due to this limitation, DEFCON dataset has been found useful only in evaluating alert correlation techniques.

DEFCON-8 dataset created in 2000 contains port scanning and buffer overflow attacks, whereas DEFCON-10 dataset, which was created in 2002, contains port scan and sweeps, bad packets, administrative privilege, and FTP by Telnet protocol attacks

CAIDA Dataset: CAIDA collects many different types of data and makes them available to the research community. CAIDA datasets are very specific to particular events or attacks resources when connecting to Internet servers. This organization has three different datasets, CAIDA OC48, CAIDA DDOS and CAIDA Internet traces 2016. Most of CAIDAs datasets are very specific to particular events or attacks and are anonymized with their payload, protocol information, and destination. These are not the effective benchmarking datasets due to a number of shortcomings.

LBNL Dataset: LBNL's internal enterprise traffic traces are full header network traces without payload. This dataset suffers from heavy anonymization to the extent that scanning traffic was extracted and separately anonymized to remove any information which could identify individual IPs.

6.2.1.3. Real-life Datasets

Real-life datasets created by collecting network traffic on several consecutive days were discussed. The details include both normal as well as attack traffic in appropriate proportions in the authors' respective campus networks (i.e., testbeds).

UNIBS Dataset: The UNIBS packet traces were collected on the edge router of the campus network of the University of Brescia in Italy, on three consecutive working days. The dataset includes traffic captured or collected and stored using 20 workstations, each running the GT (Ground Truth) client daemon. The dataset creators collected the traffic by running tcpdump on the faculty router, which was a dual Xeon Linux box that connected the local network to the Internet through a dedicated 100Mb/s uplink. They captured and stored the traces on a dedicated disk of a workstation connected to the router through a dedicated ATA controller.

ISCX-UNB Dataset: The ISCX-UNB dataset is built on the concept of profiles that include the details of intrusions. The datasets were collected using a real-time testbed by incorporating multi-stage attacks. It uses two profiles - α and β - during the generation of the datasets. α profiles are constructed using the knowledge of specific attacks and " profiles are built using the filtered traffic traces. Real packet traces were analyzed to create α and β profiles for agents that generate real-time traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols. Various multistage attack scenarios were explored to generate malicious traffic.

KU Dataset: The Kyoto University dataset is a collection of network traffic data obtained from honeypots. The raw dataset obtained from the honeypot system consisted of 24 statistical features, out of which 14 significant features were extracted. The dataset developers extracted 10 additional features that could be used to investigate network events inside the university more effectively. The initial 14 features extracted are similar to those in the KDDcup99 datasets. Only 14 conventional features were used during training and testing.

CICIDS2017: It contains benign and the most up-to-date common attacks, which resembles to a real-world data. It consists of the results of the network traffic analysis using CICFlowMeter. It focuses on generation of real-world traffic. B-Profile system is used to profile the abstract behavior of human interactions and a naturalistic benign background traffic was generated. Abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols was built. Data collection was for 5 days. They have cover almost all criteria which previous datasets could not. In the following,

They are listed as follows:

- Complete Network configuration
- Complete Traffic
- Labelled Dataset
- Complete Interaction
- Complete Capture
- Available Protocols
- Attack Diversity
- Heterogeneity
- Feature Set
- Meta-Data

CICIDS2017 Dataset is used for the evaluation of our intrusion detection system classifier in order to check the authenticity, purity and integrity of the dataset generated by ourselves.

6.2.2. Data set generation

Our dataset consists of normal as well as malicious traffic. With the local resources we have collected this dataset. The normal data consists of real world data traffic. The regular sites were visit and regular day-to-day activities were done. We were given the managed switch by the department of CIT. We connected laptops and phones in the home network and collected their network traffic for few days and then we conducted attacks possible for a beginner attacker as we know there are rarely any professional hacker in the college. So these attacks are the possible attacks that is possible for a common person. We have included attacks like probing, port scanning, brute force, Dos and we also included malware attack as in college network malwares prevail and get transferred from PCs to PCs.

6.2.2.1. Packets

A packet is a small amount of data sent over a network, such as a LAN or the Internet. Similar to a real-life package, each packet includes a source and destination as well as the content (or data) being transferred. When the packets reach their destination, they are reassembled into a single file or other contiguous block of data.
Packets are intended to transfer data reliably and efficiently. Instead of transferring a large file as a single block of data, sending smaller packets helps ensure each section is transmitted successfully. If a packet is not received or is "dropped," only the dropped packet needs to be resent. Additionally, if a data transfer encounters network congestion due to multiple simultaneous transfers, the remaining packets can be rerouted through a less congested path [25].

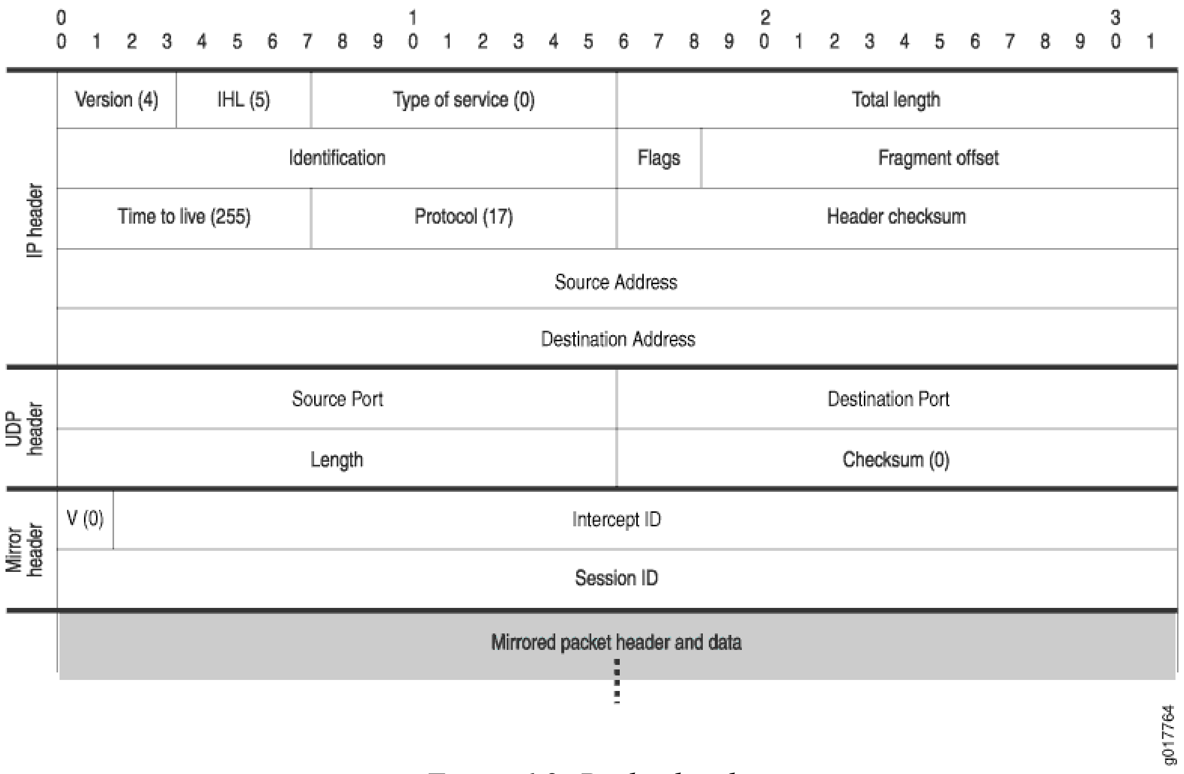


Figure 6.8: Packet header

6.2.2.2. Tools used

For packet capturing and attack simulations different tools were used. These tools are open-source software that are available in the internet. However the use of these tools on other hosts without their permission is strictly illegal and should be used for educational purpose only. The Tools used are done in safe testbed environment without hampering anyone's data inside the network.

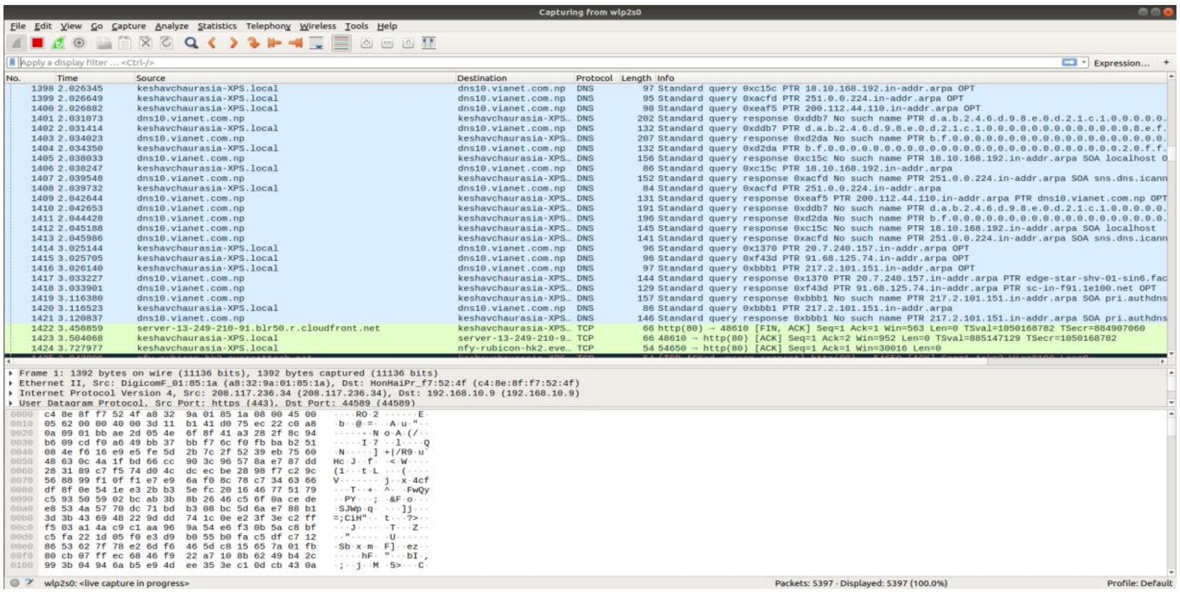
6.2.2.2.1. Wireshark

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues. Wireshark is very similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options.

Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering [27]

Wireshark is a data capturing program that "understands" the structure (encapsulation) of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols. Wireshark uses pcap to capture packets, so it can only capture packets on the types of networks that pcap supports.

Wireshark's native network trace file format is the libpcap format supported by libpcap and WinPcap, so it can exchange captured network traces with other applications that use the same format, including tcpdump and CA NetMaster. It can also read captures from other network analyzers, such as snoop, Network General's Sniffer, and Microsoft Network Monitor.


Capturing from wlp2s0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…
+

```
No.   Time      Source                                  Destination            Protocol Length Info
 1398 2.026345  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       97 Standard query 0xc15c PTR 18.10.168.192.in-addr.arpa OPT
 1399 2.026649  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       95 Standard query 0xacfd PTR 251.0.0.224.in-addr.arpa OPT
 1400 2.026882  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       98 Standard query 0xeaf5 PTR 200.112.44.110.in-addr.arpa OPT
 1401 2.031073  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      202 Standard query response 0xddb7 No such name PTR d.a.b.2.4.6.d.9.8.e.0.d.2.1.c.1.0.0.0.0.0.
 1402 2.031414  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS      132 Standard query 0xddb7 PTR d.a.b.2.4.6.d.9.8.e.0.d.2.1.c.1.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.
 1403 2.034023  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      207 Standard query response 0xd2da No such name PTR b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
 1404 2.034350  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS      132 Standard query 0xd2da PTR b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.
 1405 2.038033  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      156 Standard query response 0xc15c No such name PTR 18.10.168.192.in-addr.arpa SOA localhost O
 1406 2.038247  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       86 Standard query 0xc15c PTR 18.10.168.192.in-addr.arpa
 1407 2.039548  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      152 Standard query response 0xacfd No such name PTR 251.0.0.224.in-addr.arpa SOA sns.dns.icann
 1408 2.039732  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       84 Standard query 0xacfd PTR 251.0.0.224.in-addr.arpa
 1409 2.042644  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      131 Standard query response 0xeaf5 PTR 200.112.44.110.in-addr.arpa PTR dns10.vianet.com.np OPT
 1410 2.042653  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      191 Standard query response 0xddb7 No such name PTR d.a.b.2.4.6.d.9.8.e.0.d.2.1.c.1.0.0.0.0.0.
 1411 2.044428  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      196 Standard query response 0xd2da No such name PTR b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
 1412 2.045188  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      145 Standard query response 0xc15c No such name PTR 18.10.168.192.in-addr.arpa SOA localhost
 1413 2.045986  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      141 Standard query response 0xacfd No such name PTR 251.0.0.224.in-addr.arpa SOA sns.dns.icann
 1414 3.025144  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       96 Standard query 0x1370 PTR 20.7.240.157.in-addr.arpa OPT
 1415 3.025705  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       96 Standard query 0xf43d PTR 91.68.125.74.in-addr.arpa OPT
 1416 3.026140  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       97 Standard query 0xbbb1 PTR 217.2.101.151.in-addr.arpa OPT
 1417 3.033227  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      144 Standard query response 0x1370 PTR 20.7.240.157.in-addr.arpa PTR edge-star-shv-01-sin6.fac
 1418 3.033901  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      129 Standard query response 0xf43d PTR 91.68.125.74.in-addr.arpa PTR sc-in-f91.1e100.net OPT
 1419 3.116380  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      157 Standard query response 0xbbb1 No such name PTR 217.2.101.151.in-addr.arpa SOA pri.authdns
 1420 3.116523  keshavchaurasia-XPS.local               dns10.vianet.com.np    DNS       86 Standard query 0xbbb1 PTR 217.2.101.151.in-addr.arpa
 1421 3.120837  dns10.vianet.com.np                     keshavchaurasia-XPS…   DNS      146 Standard query response 0xbbb1 No such name PTR 217.2.101.151.in-addr.arpa SOA pri.authdns
 1422 3.458859  server-13-249-210-91.blr50.r.cloudfront.net keshavchaurasia-XPS… TCP     66 http(80) → 48610 [FIN, ACK] Seq=1 Ack=1 Win=563 Len=0 TSval=1050168782 TSecr=884907060
 1423 3.504068  keshavchaurasia-XPS.local               server-13-249-210-9…   TCP       66 48610 → http(80) [ACK] Seq=1 Ack=2 Win=952 Len=0 TSval=885147129 TSecr=1050168782
 1424 3.727977  keshavchaurasia-XPS.local               nfy-rubicon-hk2.eve…   TCP       54 54650 → http(80) [ACK] Seq=1 Ack=1 Win=30016 Len=0

▸ Frame 1: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits)
▸ Ethernet II, Src: DigicomF_01:85:1a (a8:32:9a:01:85:1a), Dst: HonHaiPr_f7:52:4f (c4:8e:8f:f7:52:4f)
▸ Internet Protocol Version 4, Src: 208.117.236.34 (208.117.236.34), Dst: 192.168.10.9 (192.168.10.9)
▸ User Datagram Protocol, Src Port: https (443), Dst Port: 44589 (44589)

0000  c4 8e 8f f7 52 4f a8 32  9a 01 85 1a 08 00 45 00   ····RO·2 ······E·
0010  05 62 00 00 40 00 3d 11  b1 41 d0 75 ec 22 c0 a8   ·b··@·=· ·A·u"··
0020  0a 09 01 bb ae 2d 05 4e  6f 8f 41 a3 28 2f 8c 94   ·····-·N o·A·(/··
0030  b6 09 cd f0 a6 49 bb 37  bb f7 6c f0 fb ba b2 51   ·····I·7 ··l····Q
0040  08 4e f6 16 e9 e5 fe 5d  2b 7c 2f 52 39 eb 75 60   ·N·····] +|/R9·u`
0050  48 63 0c 4a 1f bd 66 cc  90 3c 96 57 8a e7 87 dd   Hc·J··f· ·<·W····
0060  28 31 89 c7 f5 74 d0 4c  dc ec be 28 98 f7 c2 9c   (1···t·L ···(····
0070  56 88 99 f1 0f f1 e7 e9  6a f0 8c 78 c7 34 63 66   V······· j··x·4cf
0080  df 8f 0e 54 1e e3 2b b3  5e fc 20 16 46 77 51 79   ···T··+· ^· ·FwQy
0090  c5 93 50 59 02 bc ab 3b  8b 26 46 c5 6f 0a ce de   ··PY···; ·&F·o···
00a0  e8 53 4a 57 70 dc 71 bd  b3 08 bc 5d 6a e7 88 b1   ·SJWp·q· ···]j···
00b0  3d 3b 43 69 48 22 9d dd  74 1c 0e e2 3f 3e c2 ff   =;CiH"·· t···?>··
00c0  f5 03 a1 4a c9 c1 aa 96  9a 54 e6 f3 0b 5a c8 bf   ···J···· ·T···Z··
00d0  c5 fa 22 1d 05 f0 e3 d9  b0 55 b0 fa c5 df c7 12   ··"····· ·U······
00e0  86 53 62 7f 78 e2 6d f6  46 5d c8 15 65 7a 01 fb   ·Sb·x·m· F]··ez··
00f0  80 cb 07 ff ec 68 46 f9  22 a7 10 8b 62 49 b4 2c   ·····hF· "···bI·,
0100  99 3b 04 94 6a b5 e9 4d  ee 35 3e c1 0d cb 43 0a   ·;··j··M ·5>···C·

wlp2s0: <live capture in progress>        Packets: 5397 · Displayed: 5397 (100.0%)        Profile: Default
```

Figure 6.9: User Interface of Wireshark

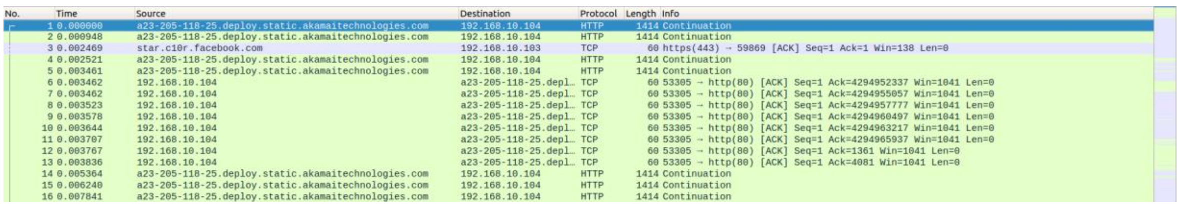
6.2.2.2.2. CICFlowMeter

CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyzer for anomaly detection that has been used in many Cybersecurity datasets such as Android Adware-General Malware dataset (CICAAGM2017), IPS/IDS dataset (CICIDS2017) and Android Malware dataset (CICAndMal2017) [27][28]

It can be used to generate bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence more than 80 statistical network traffic features such as Duration, Number of packets, Number of bytes, Length of packets, etc. can be calculated separately in the forward and backward directions.

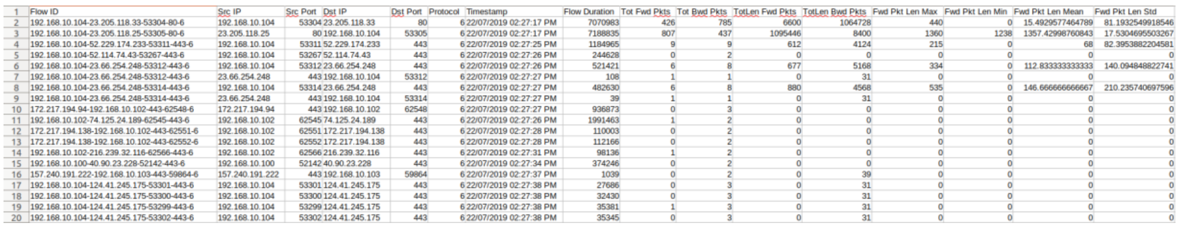
Additional functionalities include, selecting features from the list of existing features, adding new features, and controlling the duration of flow timeout. The output of the application is the CSV format file that has six columns labeled for each flow (Flow ID, Source IP, Destination IP, Source Port, Destination Port, and Protocol) with more than 80 network traffic analysis features.

Note that TCP flows are usually terminated upon connection teardown (by FIN packet) while UDP flows are terminated by a flow timeout. The flow timeout value can be assigned arbitrarily by the individual scheme e.g., 600 seconds for both TCP and UDP. [29]



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation
2	0.000948	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation
3	0.002469	star.c10r.facebook.com	192.168.10.103	TCP	60	https(443) → 59869 [ACK] Seq=1 Ack=1 Win=138 Len=0
4	0.002521	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation
5	0.003461	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation
6	0.003462	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4294952337 Win=1041 Len=0
7	0.003462	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4294955057 Win=1041 Len=0
8	0.003523	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4294957777 Win=1041 Len=0
9	0.003578	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4294960497 Win=1041 Len=0
10	0.003644	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4294963217 Win=1041 Len=0
11	0.003707	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4294965937 Win=1041 Len=0
12	0.003767	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=1361 Win=1041 Len=0
13	0.003836	192.168.10.104	a23-205-118-25.depl…	TCP	60	53305 → http(80) [ACK] Seq=1 Ack=4081 Win=1041 Len=0
14	0.005364	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation
15	0.006240	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation
16	0.007841	a23-205-118-25.deploy.static.akamaitechnologies.com	192.168.10.104	HTTP	1414	Continuation

Figure 6.10: Packet capture in raw format



1	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	Fwd Pkt Len Std
2	192.168.10.104-23.205.118.33-53304-80-6	192.168.10.104	53304	23.205.118.33	80	6	22/07/2019 02:27:17 PM	7070983	426	785	6600	1064728	440	0	15.4929577464789	81.1932549918546
3	192.168.10.104-23.205.118.25-53305-80-6	23.205.118.25	80	192.168.10.104	53305	6	22/07/2019 02:27:17 PM	7188835	807	437	1095446	8400	1360	1238	1357.42998760843	17.5304695503267
4	192.168.10.104-52.229.174.233-53311-443-6	192.168.10.104	53311	52.229.174.233	443	6	22/07/2019 02:27:25 PM	1184965	9	9	612	4124	215	0	68	82.3953882204581
5	192.168.10.104-52.114.74.43-53267-443-6	192.168.10.104	53267	52.114.74.43	443	6	22/07/2019 02:27:26 PM	244628	0	2	0	0	0	0	0	0
6	192.168.10.104-23.66.254.248-53312-443-6	192.168.10.104	53312	23.66.254.248	443	6	22/07/2019 02:27:26 PM	521421	6	8	677	5168	334	0	112.833333333333	140.094848822741
7	192.168.10.104-23.66.254.248-53312-443-6	23.66.254.248	443	192.168.10.104	53312	6	22/07/2019 02:27:27 PM	108	1	1	0	31	0	0	0	0
8	192.168.10.104-23.66.254.248-53314-443-6	192.168.10.104	53314	23.66.254.248	443	6	22/07/2019 02:27:27 PM	482630	6	8	880	4568	535	0	146.666666666667	210.235740697596
9	192.168.10.104-23.66.254.248-53314-443-6	23.66.254.248	443	192.168.10.104	53314	6	22/07/2019 02:27:27 PM	39	1	1	0	31	0	0	0	0
10	172.217.194.94-192.168.10.102-443-62548-6	172.217.194.94	443	192.168.10.102	62548	6	22/07/2019 02:27:27 PM	936873	0	3	0	0	0	0	0	0
11	192.168.10.102-74.125.24.189-62545-443-6	192.168.10.102	62545	74.125.24.189	443	6	22/07/2019 02:27:26 PM	1991463	1	2	0	0	0	0	0	0
12	172.217.194.138-192.168.10.102-443-62551-6	192.168.10.102	62551	172.217.194.138	443	6	22/07/2019 02:27:28 PM	110003	0	2	0	0	0	0	0	0
13	172.217.194.138-192.168.10.102-443-62552-6	192.168.10.102	62552	172.217.194.138	443	6	22/07/2019 02:27:28 PM	112166	0	2	0	0	0	0	0	0
14	192.168.10.102-216.239.32.116-62566-443-6	192.168.10.102	62566	216.239.32.116	443	6	22/07/2019 02:27:31 PM	98136	1	2	0	0	0	0	0	0
15	192.168.10.100-40.90.23.228-52142-443-6	192.168.10.100	52142	40.90.23.228	443	6	22/07/2019 02:27:34 PM	374246	0	2	0	0	0	0	0	0
16	157.240.191.222-192.168.10.103-443-59864-6	157.240.191.222	443	192.168.10.103	59864	6	22/07/2019 02:27:37 PM	1039	0	2	0	39	0	0	0	0
17	192.168.10.104-124.41.245.175-53301-443-6	192.168.10.104	53301	124.41.245.175	443	6	22/07/2019 02:27:38 PM	27686	0	3	0	31	0	0	0	0
18	192.168.10.104-124.41.245.175-53300-443-6	192.168.10.104	53300	124.41.245.175	443	6	22/07/2019 02:27:38 PM	32430	0	3	0	31	0	0	0	0
19	192.168.10.104-124.41.245.175-53299-443-6	192.168.10.104	53299	124.41.245.175	443	6	22/07/2019 02:27:38 PM	35381	1	3	0	31	0	0	0	0
20	192.168.10.104-124.41.245.175-53302-443-6	192.168.10.104	53302	124.41.245.175	443	6	22/07/2019 02:27:38 PM	35345	0	3	0	31	0	0	0	0

Figure 6.11: packet capture file converted to netflow records using CICFlowmeter-V4.0

Features extracted by CICFlowmeter-V4.0

CICFlowmeter-V4.0 extracts more than 80 features from the packet capture files. Following are the features and their descriptions

Feature Name	Feature Description
Flow ID	Flow Id
Src IP	Source address
Src Port	Source Port
Dst IP	Destination address
Dst Port	Destination Port
Protocol	Protocol
Timestamp	Timestamp
Flow Duration	Duration of the flow in Microsecond
Tot Fwd Pkts	Total packets in the forward direction
Tot Bwd Pkts	Total packets in the backward direction
TotLen Fwd Pkts	Total size of packet in forward direction
TotLen Bwd Pkts	Total size of packet in backward direction
Fwd Pkt Len Max	Maximum size of packet in forward direction
Fwd Pkt Len Min	Minimum size of packet in forward direction

Fwd Pkt Len Mean	Mean size of packet in forward direction
Fwd Pkt Len Std	Standard deviation size of packet in forward direction
Bwd Pkt Len Max	Maximum size of packet in backward direction
Bwd Pkt Len Min	Minimum size of packet in backward direction
Bwd Pkt Len Mean	Mean size of packet in backward direction
Bwd Pkt Len Std	Standard deviation size of packet in backward direction
Flow Byts/s	Number of flow packets per second
Flow Pkts/s	Number of flow bytes per second
Flow IAT Mean	Mean inter-arrival time of packet
Flow IAT Std	Standard deviation inter-arrival time of packet
Flow IAT Max	Maximum inter-arrival time of packet
Flow IAT Min	Minimum inter-arrival time of packet
Fwd IAT Tot	Total time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Bwd IAT Tot	Total time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)

Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Len	Total bytes used for headers in the forward direction
Bwd Header Len	Total bytes used for headers in the backward direction
Fwd Pkts/s	Number of forward packets per second
Bwd Pkts/s	Number of backward packets per second
Pkt Len Min	Minimum Length of Packet
Pkt Len Max	Maximum Length of Packet
Pkt Len Mean	Mean Length of Packet
Pkt Len Std	Standard deviation of Length of Packet
Pkt Len Var	Variance in Length of Packet
FIN Flag Cnt	Number of packets with FIN
SYN Flag Cnt	Number of packets with SYN
RST Flag Cnt	Number of packets with RST
PSH Flag Cnt	Number of packets with PSH
ACK Flag Cnt	Number of packets with ACK
URG Flag Cnt	Number of packets with URG
CWE Flag Count	Number of packets with CWE
ECE Flag Cnt	Number of packets with ECE
Down/Up Ratio	Download and upload ratio
Pkt Size Avg	Average size of packet
Fwd Seg Size Avg	Average size observed in the forward direction

Bwd Seg Size Avg	Average size observed in the backward direction
Fwd Byts/b Avg	Average number of bytes bulk rate in the forward direction
Fwd Pkts/b Avg	Average number of packets bulk rate in the forward direction
Fwd Blk Rate Avg	Average number of bulk rate in the forward direction
Bwd Byts/b Avg	Average number of bytes bulk rate in the backward direction
Bwd Pkts/b Avg	Average number of packets bulk rate in the backward direction
Bwd Blk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Pkts	The average number of packets in a sub flow in the forward direction
Subflow Fwd Byts	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Pkts	The average number of packets in a sub flow in the backward direction
Subflow Bwd Byts	The average number of bytes in a sub flow in the backward direction
Init Fwd Win Byts	The total number of bytes sent in initial window in the forward direction
Init Bwd Win Byts	The total number of bytes sent in initial window in the backward direction
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Mean	Mean time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle
Idle Mean	Mean time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

Idle Max	Maximum time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active
Label	Target Label Normal Traffic or Attack
SubLabel	Target Label Normal Traffic or Specific Attack

Table 6.2: Features of dataset extracted by CICFlowMeter

6.2.2.3. Normal Traffic Collection

Normal Traffic was collected after setting up the testbed architecture for IDS. Different host's packets were captured when they were doing their normal operations. Different normal operations performed by the host for normal traffic capture are

- Web Browsing and video streaming
- Composing and sending mail
- FTP from the FTP server
- Remote desktop connection for the work purpose
- SSH connection to remote server

6.2.2.4. Attack Simulation

Attacks are the unauthorized, illegal and forceful penetrations operations done to get access of sensitive information such as username and password. Different attacks that were simulated by us in our project are listed below.

6.2.2.4.1. Probing

It includes information gathering in a network. Probing is making an attempt to gain access to a vulnerable computers and its files through a known or a weak point in the computer system. It includes PING sweep, ICMP sweep, etc. This attack is generally used to know the active devices in the network and also shows the vulnerability in the network. We used Nmap tool that is pre-installed in kali-Linux OS. We ran different kinds of PING sweeps and discovered the hosts which were up. We captured the network traffic in form of PCAPS using the pre-installed tool “Wireshark” in kali Linux.

6.2.2.4.2. Port Scan

Port scan on a network shows which ports are open and listening as well as it reveals the presence firewalls between the sender and the target. There are 65,535 ports in each IP address and we scanned each ports after we selected the target. We used Nmap tool for this purpose. We captured the network traffic in form of PCAPS using the pre-installed tool “Wireshark” in kali Linux. Port scan included TCP SYN scan, **TCP Connect or Vanilla scan,** UDP scan, TCP NULL, FIN, and Xmas scans, TCP ACK scan, FTP Bounce Scan, Windows scan, etc.

6.2.2.4.3. Brute force

It is a trial and error method where the attacker tries various combinations to decode the encrypted data and get access to the private keys and PIN numbers. A lot of tools are available to conduct brute force attack and among them we used THC hydra. We Performed host scan and then targeted an Ubuntu machine with it’s SSH port Open. Then with Hydra we implemented dictionary attack with the username known but password unknown. Using Patator we implemented FTP brute force and MySQL brute force.

6.2.2.4.4. Denial of Service (DoS)

A denial-of-service (DoS) attack is a cyber-attack where an attacker tries to make the system service unavailable to its intended users by interrupting the device's normal functioning. We used hping3 and metasploit to conduct various DoS attacks like SYN flood, UDP flood, and ICMP flood. And we also used LOIC, Slowloris and goldeneye to perform this attack.

6.2.2.4.5. Malware Infiltration

It is simply a computer virus that runs malicious activity in the network once executed. We used msfvenom to create payload with reverse TCP connection for different devices like windows and android phone. We then included the local host to be the attacker IP and port 4444 then we installed the apk and .exe file extensions in the host system. So after installation the listening started and the connection was carried out using metasploit multi handler. Hence we conducted various malicious activities.

6.3. Real Time Malicious URL Detection

A malicious URL is a link created with the purpose of promoting scams, attacks and frauds. By clicking on an infected URL, you can download a malware or a Trojan that can take your devices, or you can be persuaded to provide sensitive information on a fake website. The most common scams with malicious URLs involve spam and phishing. Phishing is a type of fraud used by criminals who try to deceive victims by impersonating well-known and trusted organizations or people. It means that you may receive a malicious URL within an email from a friend if his email account has been compromised or if the criminal is trying to deceive you by spoofing your friend's name and address. Malicious links may also be hidden in supposedly safe download links and may spread quickly through the sharing of files and messages in sharing networks. Just like with emails, websites can also be compromised, which can lead users to click on malicious URLs and provide sensitive information directly to fraudsters. Within the multitude of cyber threats out there, malicious websites play a critical role in today's attacks and scams. The end result can often be downloaded malware, spyware, ransomware, compromised accounts, and all the headaches those threats entail.

There are more than 1.86 billion websites on the internet. Around 1% of these, something like 18,500,000, are infected with malware at a given time each week; while the average website is attacked 44 times every day.

These conventional mechanisms cannot effectively deal with the ever evolving technologies and web- access techniques. Furthermore, these approaches also fall short in detecting the modern URLs such as short URLs, dark web URLs. In this paper, we propose a novel classification method to address the challenges faced by the traditional mechanisms in malicious URL detection. The proposed classification model is built on sophisticated machine learning methods that not only takes care about the syntactical nature of the URL, but also the semantic and lexical meaning of these dynamically changing URLs. The proposed approach is expected to outperform the existing techniques.

The first step in network intrusion is to prevent yourself from accidentally clicking or downloading malicious contents and scripts from the internet itself. Real Time Malicious

URL Detection is developed to warn the network admin and ultimately the user to stop accessing the malicious websites.

In our project we aimed to build a real time malicious URL detection using machine learning model that continuously monitors the browsing activity and checks the URL is malicious or not. Any malicious URL found is saved in the database for further analysis and the user is notified of the malicious website in their mobile phone.

6.3.2. Data set

The data set used in this project is obtained from the source provided by the link https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs/tree/master/data. The data set contains a total of 420464 URLs out of which 344821 URLs are benign URLs and 75643 URLs are marked as malicious.

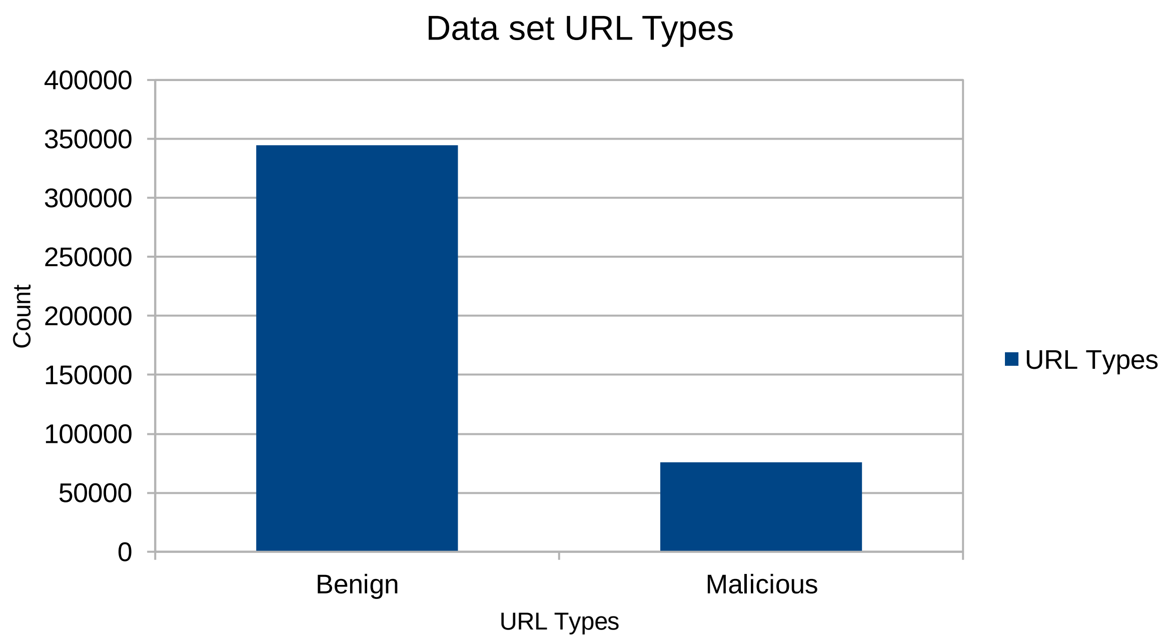


Figure 6.12: Bar chart of data set URL Types

6.3.2. Classification model

6.3.2.1. TF-IDF Vectorizer

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.[1] It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. TFIDF is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf–idf [30].

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Term Frequency

Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, "the brown cow". A simple way to start out is by eliminating documents that do not contain all three words "the", "brown", and "cow", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. However, in the case where the length of documents varies greatly, adjustments are often made (see definition below). The first form of term weighting is due to Hans Peter Luhn (1957) which may be summarized as:[31]

The weight of a term that occurs in a document is simply proportional to the term frequency.

In the case of the term frequency *tf(t,d)*, the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the simplest tf scheme is $tf(t,d) = f_{t,d}$. Other possibilities include [32]

- Boolean "frequencies": $tf(t,d) = 1$ if t occurs in d and 0 otherwise;
- term frequency adjusted for document length : $f_{t,d} \div (number\ of\ words\ in\ d)$
- logarithmically scaled frequency: $tf(t,d) = log\ (1 + f_{t,d})$;[33]
- augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the raw frequency of the most occurring term in the document:

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

Inverse document frequency

The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

.

Term frequency-Inverse document frequency

Then tf–idf is calculated as

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf–idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf–idf closer to 0.

6.3.2.2. Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from logistic unit, hence the alternative names. Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

And it looks like this:

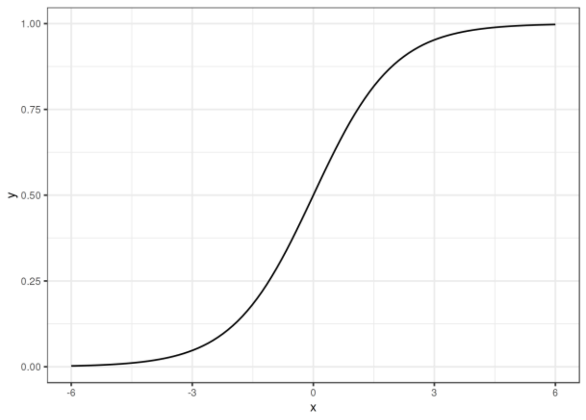


Figure 6.13: The logistic function. It outputs numbers between 0 and 1. At input 0, it outputs 0.5.

The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modeled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

Classification works better with logistic regression and we can use 0.5 as a threshold in both cases.

6.3.2.3. Building Logistic Regression Classifier

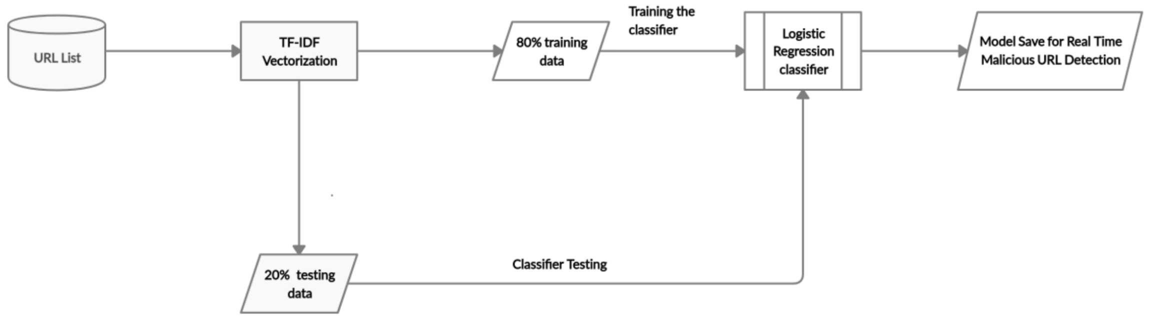
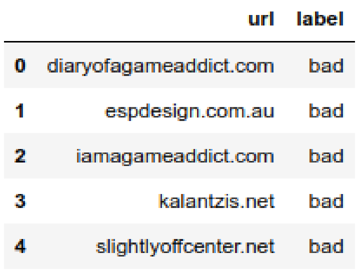


Figure 6.14: Overview of our classifier development, training and testing

Figure 6.14 shows the overview of how the classifier was built. In this task, the URL list was loaded in CSV format. The URL list contains two columns. One column has the name of the URL and the other column has the label that identifies if the URL is malicious or not.



	url	label
0	diaryofagameaddict.com	bad
1	espdesign.com.au	bad
2	iamagameaddict.com	bad
3	kalantzis.net	bad
4	slightlyoffcenter.net	bad

Table 6.2: Table showing list of URLs and their label

The URL List is vectorized using Scikit-learn tf-idf vectorizer. After vectorization whole data set is split into training and testing data set in the ratio 80:20. A logistic regression classifier is trained with training set and tested with training set. The trained model and the vectorizer is the saved for using in real time malicious URL detection.

6.3.2.4. Real Time System for Malicious URL Detection

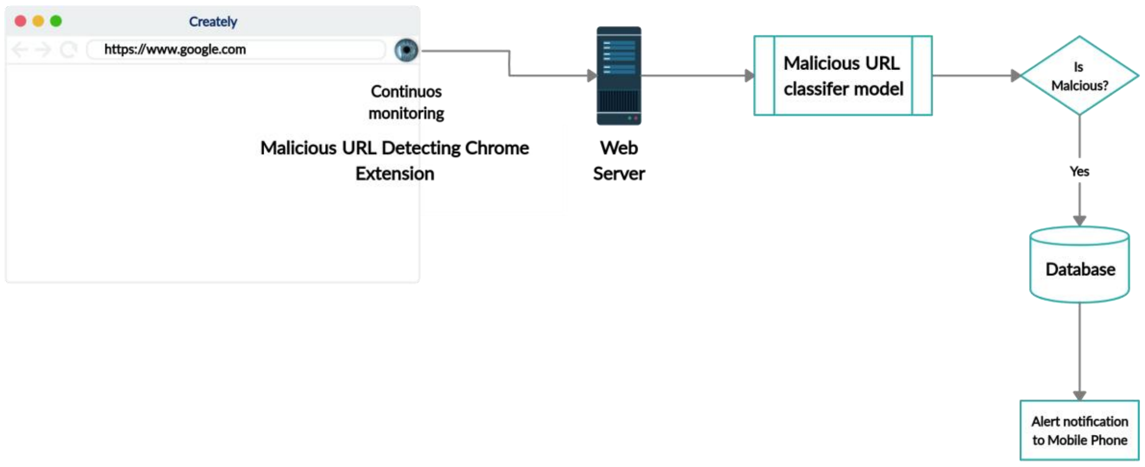

Creately
https://www.google.com
Continuos monitoring
Malicious URL Detecting Chrome Extension
Web Server
Malicious URL classifer model
Is Malcious?
Yes
Database
Alert notification to Mobile Phone

Figure 6.15: real time system for malicious URL detection

6.3.2.4.1. Google chrome extensions

Figure 6.16: Google chrome extension developed for getting URL links

Extensions are small software programs that customize the browsing experience. They enable users to tailor Chrome functionality and behavior to individual needs or preferences. They are built on web technologies such as HTML, JavaScript, and CSS. These are add-ons to the browser which helps in adding more features. An extension must fulfill a single purpose that is narrowly defined and easy to understand. A single extension can include multiple components and a range of functionality, as long as everything contributes towards a common purpose.

For the case of our system we need to continuously retrieve URLs. Real time monitoring of web usage requires the permission to continuously extract URL from the browser. So the extension is programmed with default permissions provided. When the user enters the URL, the extension takes the URL and sends POST request to web server that is continuously listening for the request. The request is sent using JavaScript and asynchronous [34].

6.3.2.4.2. Classifier

The web server after receiving the request it loads the vectorizer and classifier that was built after training the model. The input URL is vectorized using the loaded vectorizer and then passed as input to the classifier. The classifier classifies the URL link if it is malicious or not. If the link is found malicious it is logged into the database for reporting and an alert notification is sent to the mobile user.

6.3.2.4.3. Pusher

Pusher Channels provides real-time communication between servers, apps and devices. Channels is used for notifications, chat, gaming, web-page updates, IoT, and many other systems requiring real-time communication. Pusher sits as a real-time layer between your servers and your clients. Pusher maintains persistent connections to the clients - over Web Socket if possible and falling back to HTTP-based connectivity - so that as soon as your servers have new data that they want to push to the clients they can do, instantly via Pusher. Pusher offers libraries to integrate into all the main runtimes and frameworks. PHP, Ruby, Python, Java, .NET, Go and Node on the server and JavaScript, Objective-C (iOS) and Java (Android) on the client [35].

For the notification system to work, an instance of pusher is created in using a pusher account. Necessary credentials are provided by the pusher to install it in the system. When sending the notification, the pusher uses these credentials to make a pusher request and send notification to the registered devices. Currently in our project only one android device is registered and all the threat and malicious visit detection arrive in the same phone.

This approach has a very low latency and is very close to continuous real time detection. The lag time between loading the classifier every time a request for URL checking is reduced by caching the model in the webserver. This dropped the lag time. The extension built is for the chrome browser and needs to be extended for other browsers such as Firefox, safari as well.

6.4. Real time Network Intrusion Detection

6.4.1. Data collection, cleaning and preprocessing

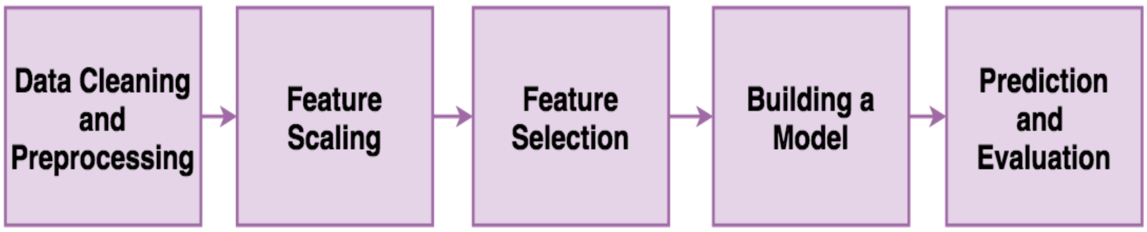


Figure 6.17: Flowchart for data cleaning and preprocessing

After the dataset generation and conversion of packet capture to csv flow records. The data needs to be cleaned and preprocessed.

Data Cleaning and Preprocessing: Basically in this step the dataset has to go through a cleaning process to remove duplicate records, as the dataset used was already cleaned, this step was not required. Next a Pre-processing operation was carried in place because the dataset contains numerical and non-numerical instances. Generally the model defined in the Scikit-learn works well with numerical inputs, so one-hot encoding method was used to make that transformation. This technique transformed each categorical feature with m possible inputs to n binary features, with one active at the time only.

Features scaling: Features scaling is a common requirement of machine learning methods, to avoid the features with large values that may weight too much on the final results. For each feature, we calculated the average, subtracted the mean value from the feature value, and divided the result by their standard deviation. After scaling, each feature had a zero average, with a standard deviation of one.

Features Selection: Feature selection was used to eliminate the redundant and irrelevant data. It is a technique of selecting a subset of relevant features that fully represents the given problem alongside a minimum deterioration of presentation, two possible reasons were analyzed why it would be recommended to restrict the number of features:

Firstly, it is possible that irrelevant features could suggest correlations between features and target classes that arise by chance and do not model the problem correctly. This aspect is also related to over-fitting. Secondly, a large number of features could greatly increase the computation time without any improvement in classification.

The feature selection process started with a univariate feature selection with ANOVA F-test for feature scoring, univariate feature selection analyzes each feature individually to determine the strength of the relationship of the feature with labels. The SelectPercentile method in the sklearn.feature_selection module were used. This method selected features based on a percentile of the highest scores. When, the best subset of features were found, a recursive feature elimination was applied. It constantly builds a model, placing the feature aside and then repeating the process with the remained features until all features in the dataset are exhausted. It is a good optimization method for finding the best performing subset of features. The idea here is to use the weights of a classifier to produce a feature ranking. However, in our dataset feature selection showed significant performance degradation. Thus, we eliminated 5 features and opted for using all the 78 features.

6.4.2. Classifier Model

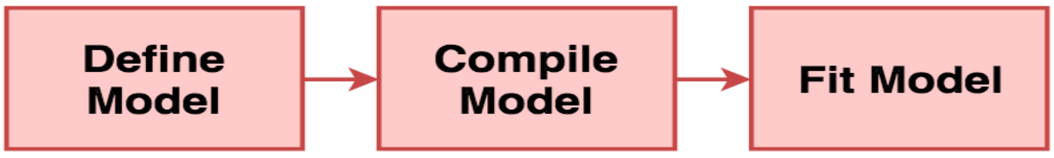


Figure 6.18: Steps involved in building a model

6.4.2.1. Define Model

Artificial Neural Network

Artificial Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us to cluster and classify the data. They behave as a clustering and classification layer on top of the data to be stored and managed. They help to group unlabeled data according to similarities among the inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification. It consists of three layers: input layer, hidden layer and output layer.

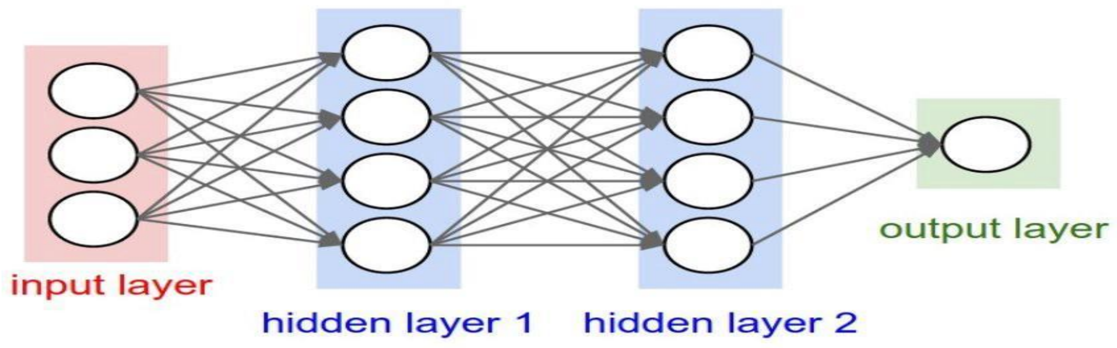


Figure 6.19: Layers of Neural Network

Densely connected neural network layers

Layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives an input from all the neurons present in the previous layer thus, they're said to be densely connected. In other words, the dense layer is a fully connected layer, meaning all the neurons in a layer are connected to those in the next layer.

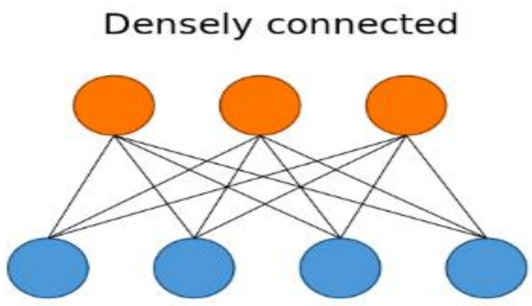


Fig 6.20: Densely Connected Layers

ReLU Activation function

The rectified linear activation function is a piecewise linear function that will output the input directly if the value is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

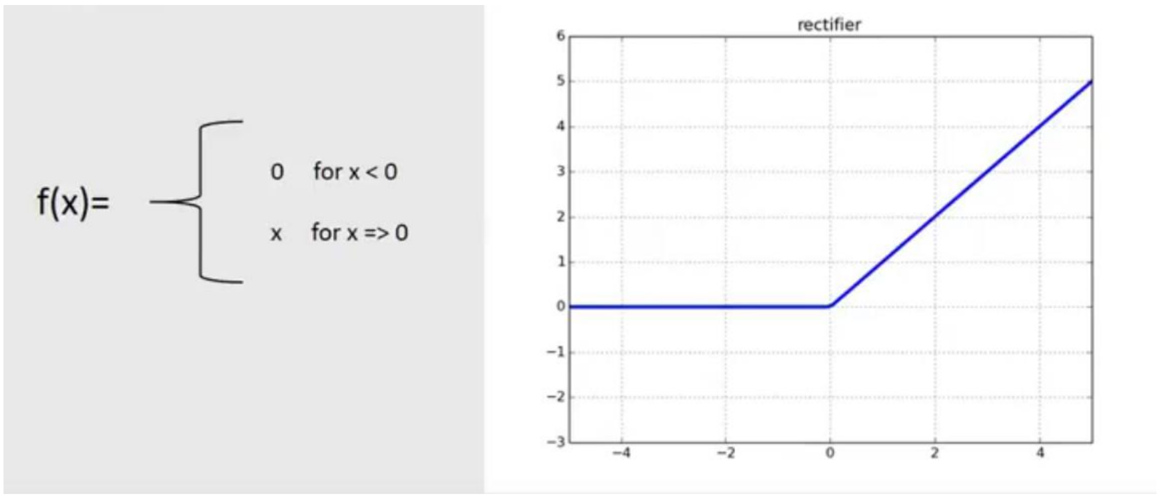


Figure 6.21: Representation of ReLU activation function

We used ReLU because of the following reasons.

- The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem.
- The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

- The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks.

Adam Optimizer

The choice of optimization algorithm for deep learning model can mean the difference between good results in minutes, hours, and days.

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network.

How it works:

1. First, it calculates an exponentially weighted average of past gradients, and stores it in variables V_{dW} & V_{db}(before bias correction) and $V_{dW}^{corrected}$ & $V_{db}^{corrected}$ (with bias correction).
2. Then it calculates an exponentially weighted average of the squares of the past gradients, and stores it in variables S_{dW} & S_{db} (before bias correction) and $S_{dW}^{corrected}$ & $S_{db}^{corrected}$ (with bias correction).
3. Finally updates parameters in a direction based on combining information from "1" and "2".

Algorithm:

1. Initialize V_{dW}, S_{dW}, V_{db} and S_{db} to zero.
2. On iteration T, compute the derivatives dw & db using current mini-batch.
3. Update V_{dW} and V_{db} like momentum.

   $V_{dW} = ß_1 \text{ x } V_{dW} + (1- ß_1) \text{ x } dW$

   $V_{db} = ß_1 \text{ x } V_{db} + (1 – ß_1) \text{ x } db.$
4. Update S_{dW} and S_{db} .

   $S_{dW} = ß_2 \text{ x } S_{dW} + (1- ß_2) \text{ x } dW^2$

$S_{db} = ß_2 \text{ x } S_{db} + (1 – ß_2) \text{ x } db^2$.

5. In Adam optimization implementation, we do implement bias correction.
6.
   $V_{dW}^{corrected} = V_{dW} / (1- ß_1^t)$

   $V_{db}^{corrected} = V_{db} / (1- ß_1^t)$

   $S_{dW}^{corrected} = S_{dW} / (1- ß_2^t)$

   $S_{db}^{corrected} = S_{db} / (1 – ß_2^t)$
7. Update parameters W and b.

   $W = W – \text{learning rate x } (V_{dW}^{corrected} / \text{sqrt}(S_{dW}^{corrected} + ε))$

   $b = b – \text{learning rate x } (V_{db}^{corrected} / \text{sqrt}(S_{db}^{corrected} + ε))$

Where:

- epsilon ‘ε’ is a very small number to avoid dividing by zero (epsilon = 10-8).
- ß1 and ß2 are hyper parameters that control the two exponentially weighted averages. In practice we use the default values for ß1 = 0.9 and ß2 = 0.999.
- Alpha is the learning rate and a range of values to be tested to see what works best for different problems.

We used Adam optimizer because of the following reason

1. Straightforward to implement.
2. Computationally efficient.
3. Little memory requirements.
4. Invariant to diagonal rescale of the gradients.
5. Well suited for problems that are large in terms of data and/or parameters.
6. Appropriate for non-stationary objectives.
7. Appropriate for problems with very noisy/or sparse gradients.
8. Hyper-parameters have intuitive interpretation and typically require little tuning.

We created a Sequential model and added layers one at a time until we were happy with our network topology. We have used a fully-connected network structure with x layers. Fully connected layers are defined using the dense class. We specified the number of neurons in the layer as the first argument, the initialization method as the second argument as init and specified the activation function using the activation argument. We initialized the network weights to a small random number generated from a uniform distribution, in this case between 0 and 0.05 because that is the default uniform weight initialization in Keras.

We used the rectifier (‘**ReLU**‘) activation function on the first two layers and the softmax function in the output layer. Sigmoid and tanh activation functions used to be preferred for all layers. These days, performance achieved is better using the rectifier activation function. The hidden layer has 78 neurons and expects 78 input variables. The output layer has 5 neurons to predict the multiple class attacks.

6.4.2.2. Compile Model

Compiling the model uses the efficient numerical libraries under the covers such as TensorFlow. The backend automatically selects the best way to represent the network for training and making predictions to run on the hardware, such as CPU or GPU or even distributed. When compiling, we specified some additional properties required when training the network to find the best set of weights to make predictions for our problem. We identified the loss function used to evaluate a set of weights, the optimizer used to search through different weights for the network and any optional metrics we would like to collect and report during training. We used the efficient gradient descent algorithm “adam”. Finally, because it is a classification problem, we reported the classification accuracy as the metric.

6.4.2.3. Fit Model

The train set and test set data were divided in 70:30 ratio. The training process ran for a fixed number of iterations through the dataset called epochs, that we set using the epoch argument while fitting the model. Our model ran for 32 epochs.

6.4.2.4. Prediction

The test data was used to make prediction and evaluation of our model, multiple settings was considered such as the accuracy score, precision, recall, f-measure and a confusion matrix. The evaluations are shown in the result section. The classifier model is saved for using it in real time network intrusion detection system.

6.4.3. Real time network Intrusion Detection System

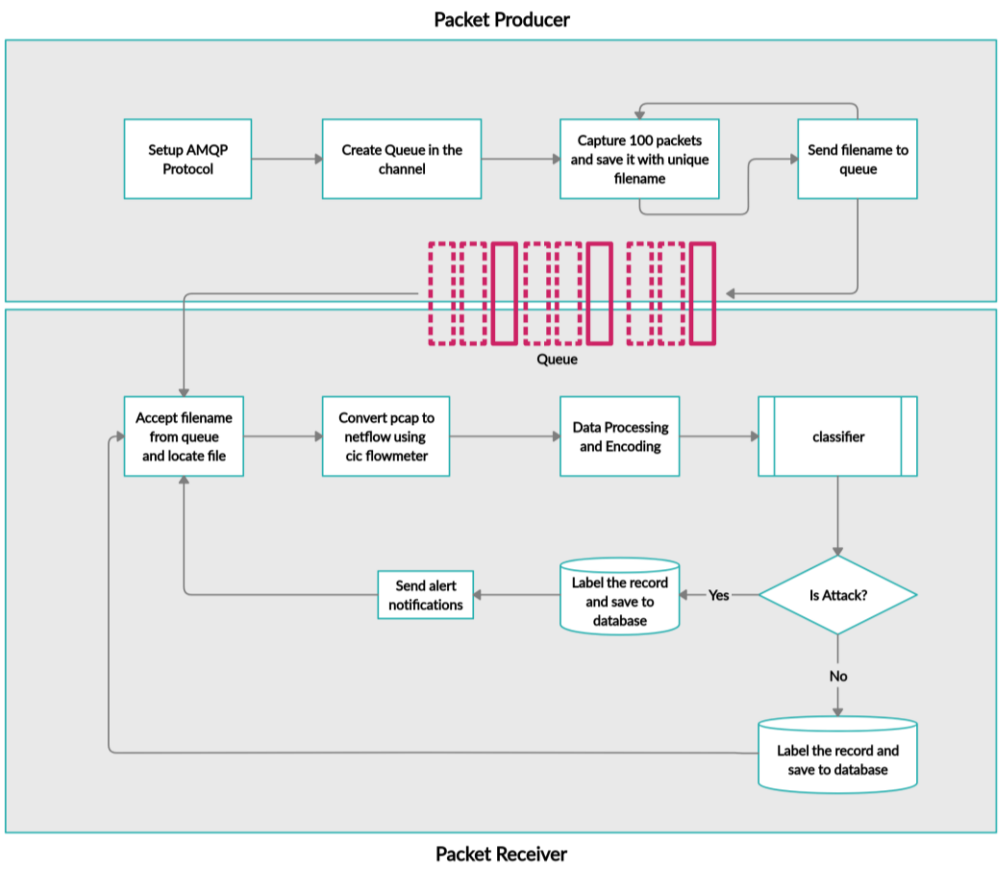

Packet Producer
Setup AMQP Protocol
Create Queue in the channel
Capture 100 packets and save it with unique filename
Send filename to queue
Queue
Accept filename from queue and locate file
Convert pcap to netflow using cic flowmeter
Data Processing and Encoding
classifier
Is Attack?
Yes
Label the record and save to database
Send alert notifications
No
Label the record and save to database
Packet Receiver

Figure 6.22: Flow chart of real time intrusion detection system

6.4.3.1. Advanced Message Queuing Protocol (AMQP)

The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security [36].

AMQP mandates the behavior of the messaging provider and client to the extent that implementations from different vendors are interoperable, in the same way as SMTP, HTTP, FTP, etc. have created interoperable systems. Previous standardizations of middleware have happened at the API level (e.g. JMS) and were focused on standardizing programmer interaction with different middleware implementations, rather than on providing interoperability between multiple implementations.[37] Unlike JMS, which defines an API and a set of behaviors that a messaging implementation must provide, AMQP is a wire-level protocol. A wire-level protocol is a description of the format of the data that is sent across the network as a stream of bytes. Consequently, any tool that can create and interpret messages that conform to this data format can interoperate with any other compliant tool irrespective of implementation language.

6.4.3.2. RabbitMQ

RabbitMQ is an open-source message-broker software (sometimes called message-oriented middleware) that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), Message Queuing Telemetry Transport (MQTT), and other protocols [38].

RabbitMQ is a lightweight, reliable, scalable and portable message broker. But unlike many message brokers familiar to Java developers, it's not based on JMS. Instead, your applications communicate with it via a platform-neutral, wire-level protocol: the Advanced Message Queuing Protocol (AMQP). Fortunately there's already a Java client library and

Spring Source is working on first class Spring and Grails integration - so don't worry about having to do low-level stuff to use RabbitMQ. You can even find AMQP client libraries that expose a JMS interface. But AMQP is sufficiently different in operation from JMS that it might cause headaches for Java developers that are used to the JMS model.

In order to ease the transition, I'll be looking in this post at the basic concepts that underpin AMQP along with three common usage scenarios. By the end, you will hopefully have a good enough understanding to configure RabbitMQ and use it via the APIs provided by Spring and Grails.

Like any messaging system, AMQP is a message protocol that deals with publishers and consumers. The publishers produce the messages, the consumers pick them up and process them. It's the job of the message broker (such as RabbitMQ) to ensure that the messages from a publisher go to the right consumers. In order to do that, the broker uses two key components: exchanges and queues. The following diagram shows how they connect a publisher to a consumer:

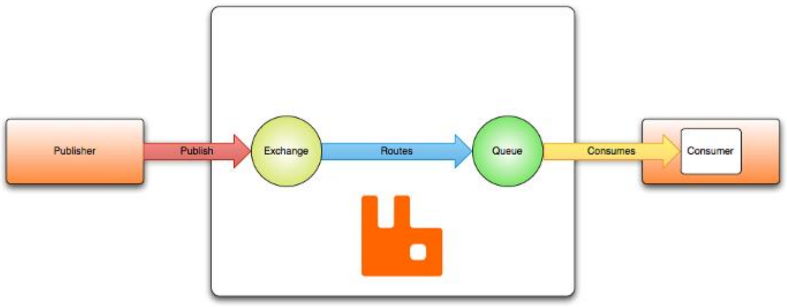


Figure 6.23: Exchange, channel and queue in RabbitMQ server

Examples

This section gives sample programs written in Python (using the pika package) for sending and receiving messages using a queue.

Sending: The following code fragment establishes a connection, makes sure the recipient queue exists, then sends a message and finally closes the connection.

```python
#!/usr/bin/env python
import pika
connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
channel = connection.channel()
channel.queue_declare(queue='hello')
channel.basic_publish(exchange='', routing_key='hello', body='Hello World!')
print(" [x] Sent 'Hello World!'")
connection.close()
```

Figure 6.24: Example code for sending message in RabbitMQ server

Receiving: Similarly, the following program receives messages from the queue and prints them on the screen:

```python
#!/usr/bin/env python
import pika
connection = pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
channel = connection.channel()
channel.queue_declare(queue='hello')
print(' [*] Waiting for messages. To exit press CTRL+C')
def callback(ch, method, properties, body):
    print(" [x] Received %r" % body)
channel.basic_consume(callback, queue='hello', no_ack=True)
channel.start_consuming()
```

Figure 6.25: Example code for receiving message from RabbitMQ server

RabbitMQ server was installed including the dependencies it required and the service was started. RabbitMQ provided us the exchange service required to send the filename of the received pcap files to the process where the pcap files are analyzed. After defining a queue and exchange channel we can send and receive bytes in queuing style.

6.4.3.3. Pyshark

Pyshark is a python wrapper for tshark, allowing python packet parsing using Wireshark dissectors. There are quite a few python packet parsing modules, this one is different because it doesn't actually parse any packets, it simply uses tshark's (Wireshark command-line utility) ability to export XMLs to use its parsing. This package allows parsing from a capture file or a live capture, using all Wireshark dissectors you have installed. Pyshark allows us to capture live packets in ring buffer continuously and save the packet capture files in ".pcap" extensions.

After the creation of a queue, Pyshark commands were used to capture packets at count of 100. Then the captured files were stored as pcap files. It was given a unique name so that the filename doesn't clash with the existing files. After file formation, the filename was passed to the queue for further processing and it started capturing process again. The process runs on infinite loop to continuously sniff packets. There is a very few lag time in seconds as it doesn't check for intrusion immediately for each packets.

In another script, intrusion detection classifier is loaded and the channel is created to listen for messages from the buffer queue provided by RabbitMQ AMQP server. The queue has the feature of callback that is executed whenever it receives a new message from the channel. On receiving the filename in the message, it locates the packet capture file. The packet capture file is unreadable and unusable for the classifier and feature extraction is necessary for the preprocessing. Feature extraction is done by using CIC flowmeter. It takes pcap files as input and in turn outputs the flow records with their features. The flow extracted is bidirectional.

Once the features are extracted, unnecessary fields like Flow Id, Src IP, Dest IP, Src Port, and Timestamp are removed. These features don't contribute anything and is meaningless. After the unnecessary feature removal, the records are then normalized using standard scaler normalization technique. It standardizes by removing the mean and scaling to unit variance [39]. After standardization, target variable is dropped and the data is passed to the classifier to predict for intrusions. On predicting the flow record, it labels the original data and saves each flow record in database. If the flow record is not normal then alert notification is sent to network admin in their mobile phones to check the network dashboard for the flow record and take preventive measures. If the batch pcap has no any anomaly then the pcap is deleted from the system and if it has any record than the pcap is saved for future analysis.

In this way, real time network intrusion detection system has been developed in our project. Though the system is not completely real time and does processing and detection in batch, the batch size being very small and the processing time reduced drastically by using simultaneous packet capturing and intrusion detection, we can say it as real-time.

6.5. Network Traffic Reporting and Visualization

6.5.1. Tools and Techniques

For the Network Data Visualization, the tools used are Django Web Framework for the backend purposes and Plotly.js for the front-end data visualization.

6.5.1.1. Django Web Framework

The Django is a high-level web framework written in python programming language. Django is very simple tool for rapid web application development but yet very powerful tool. It ships with large inbuilt library and sqlite3 database. Django includes dozens of extras you can use to handle common Web development tasks. Django can handle user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

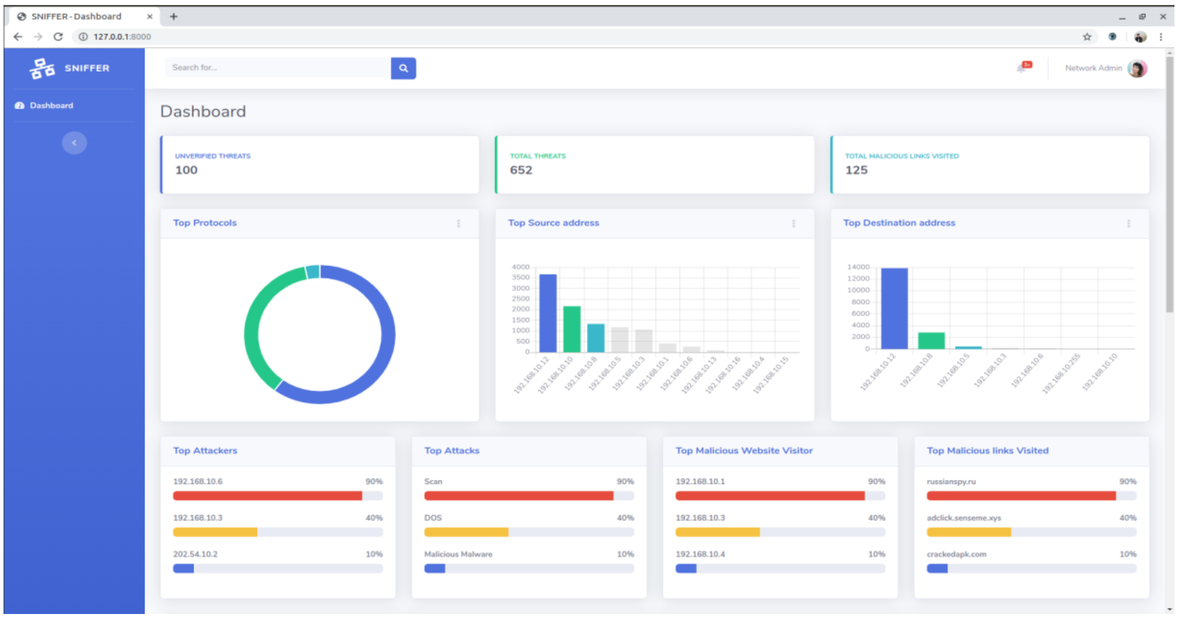
Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords. Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands. Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

6.5.1.2. Plotly.js JS Graphing library

Plotly JS is an open source visualization library in JavaScript. It is built on top of d3.js and stack.gl graphing library, plotly.js is a high-level, declarative charting library. Plotly.js ships with 20 chart types, including 3D charts, statistical graphs, and SVG maps. It also includes the visualization for the financial data, time-series data, geo mapping and contour plotting. It is very flexible library because it is highly customizable. It has high performance visualization and stunning animation functions and many more. The reason we choose this library is its features like highly customizable, high performance, great animation and very compatible with python (in which Django is also written) in context of data representation and manipulation. We have used basic chart like bar graph, pie charts and box plot. Also time series chart for the network flow visualization and advanced chart like parallel coordinates for the multidimensional data visualization.

6.5.2. Reporting Dashboard

The Reporting dashboard has the reports and statistics created from the records collected from the malicious URLs visited and intrusion detections. It allows us to the network administrators to see the overview of the number of attacks, malicious URLS visited inside the network and take preventive actions to safeguard the network.



Dashboard

UNVERIFIED THREATS	TOTAL THREATS	TOTAL MALICIOUS LINKS VISITED
100	652	125

Top Attackers

192.168.10.6	90%
192.168.10.3	40%
202.54.10.2	10%

Top Attacks

Scan	90%
DOS	40%
Malicious Malware	10%

Top Malicious Website Visitor

192.168.10.1	90%
192.168.10.3	40%
192.168.10.4	10%

Top Malicious links Visited

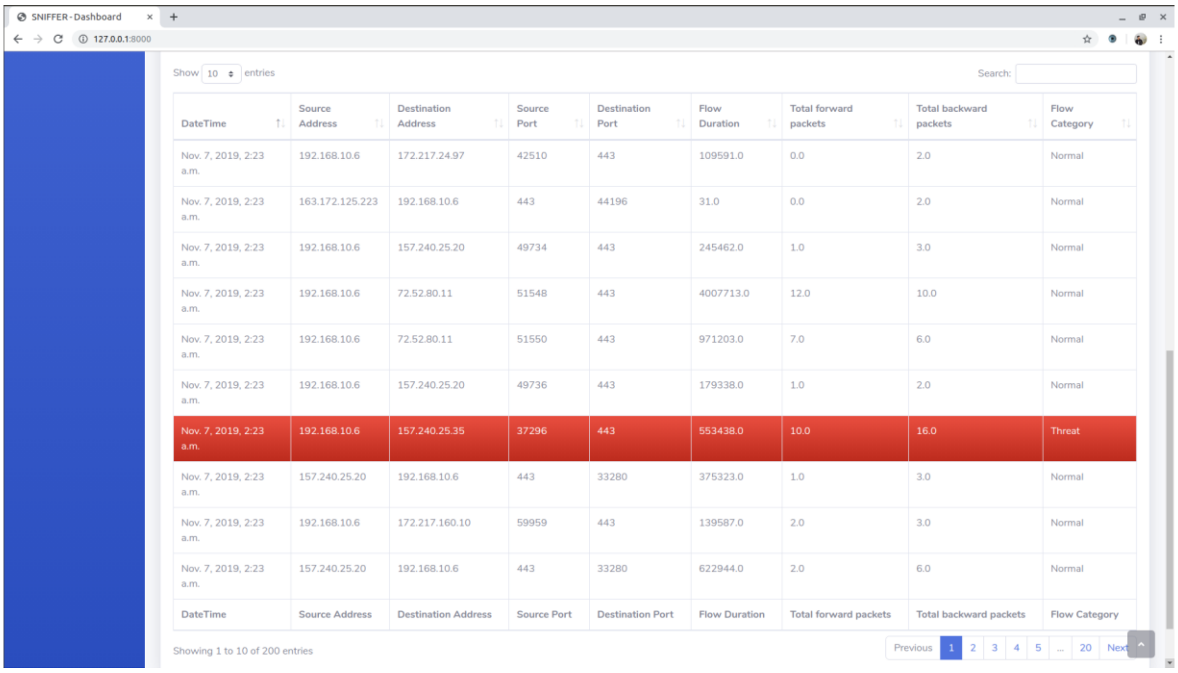
russianspy.ru	90%
adclick.senseme.xys	40%
crackedapk.com	10%

Figure 6.26: Reporting Dashboard

Reporting dashboard includes different overviews as below:

- Total number of unverified threats
- Total number of threats encountered

- Total malicious URLs visit detected
- Top Protocols
- Top Source address
- Top Destination address
- Top attacking IP Address
- Top Attacks encountered
- Top Malicious Website Visitor
- Top Malicious Website URLs



DateTime	Source Address	Destination Address	Source Port	Destination Port	Flow Duration	Total forward packets	Total backward packets	Flow Category
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	172.217.24.97	42510	443	109591.0	0.0	2.0	Normal
Nov. 7, 2019, 2:23 a.m.	163.172.125.223	192.168.10.6	443	44196	31.0	0.0	2.0	Normal
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	157.240.25.20	49734	443	245462.0	1.0	3.0	Normal
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	72.52.80.11	51548	443	4007713.0	12.0	10.0	Normal
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	72.52.80.11	51550	443	971203.0	7.0	6.0	Normal
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	157.240.25.20	49736	443	179338.0	1.0	2.0	Normal
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	157.240.25.35	37296	443	553438.0	10.0	16.0	Threat
Nov. 7, 2019, 2:23 a.m.	157.240.25.20	192.168.10.6	443	33280	375323.0	1.0	3.0	Normal
Nov. 7, 2019, 2:23 a.m.	192.168.10.6	172.217.160.10	59959	443	139587.0	2.0	3.0	Normal
Nov. 7, 2019, 2:23 a.m.	157.240.25.20	192.168.10.6	443	33280	622944.0	2.0	6.0	Normal
DateTime	Source Address	Destination Address	Source Port	Destination Port	Flow Duration	Total forward packets	Total backward packets	Flow Category

Showing 1 to 10 of 200 entries

Figure 6.27: Reporting Dashboard

This table shows the logs from the network intrusion detection system. All the records are shown with their features and targets. These logs are extremely useful when analyzing the network traffic flows.

6.5.3. Visualizations Dashboard

This dashboard captures overall visualization, allowing the analyst to see related flows. The various techniques used in our data visualizations are:

Network Flow Visualization

In 2D Network Flow visualization, the total amount of data flows per minute is depicted. This visualization is sub-divided into two parts. The first one is the total flows according to the IP protocols. This tells us how much amount of data flow in our system according to IP protocols. This can be used to analyze the time to time flow of data. The peaks in the flow diagrams can be analyzed to determine the maximum data flown.

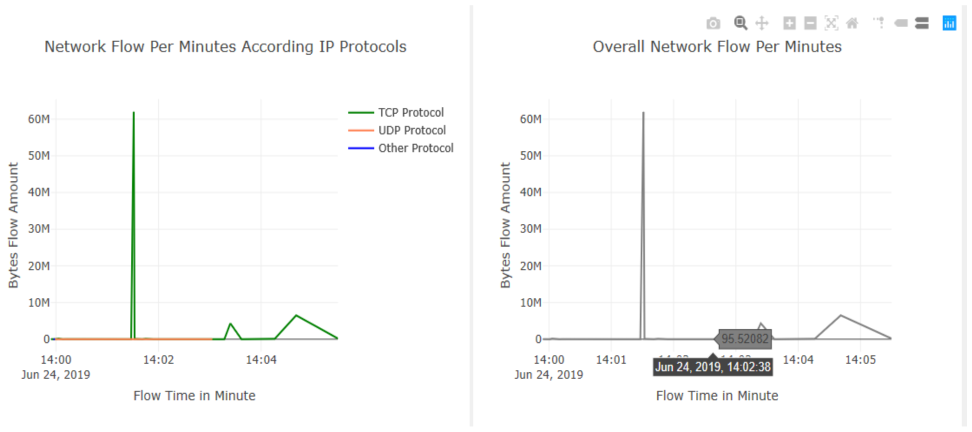


Figure 6.28: Network Flow Visualization

Parallel Coordinates Visualization

Parallel coordinates is a common method of visualizing multi-dimensional geometry and analyzing multivariate data. To show a set of points in an n-dimensional space, a backdrop is drawn consisting of n parallel lines, typically vertical and equally spaced.

Mapping is done from one lines to another with the sets of multi-dimensional datasets attributes values. In our visualization we showed the mapping of Source IP address, Source Port, Destination IP and Destination Port. The range of color is assigned for each lines while mapping. This can tell us which port at source or destination is used most or the least and concentrate more on that port for further analysis. Also, we can visualize the most used IP addresses.

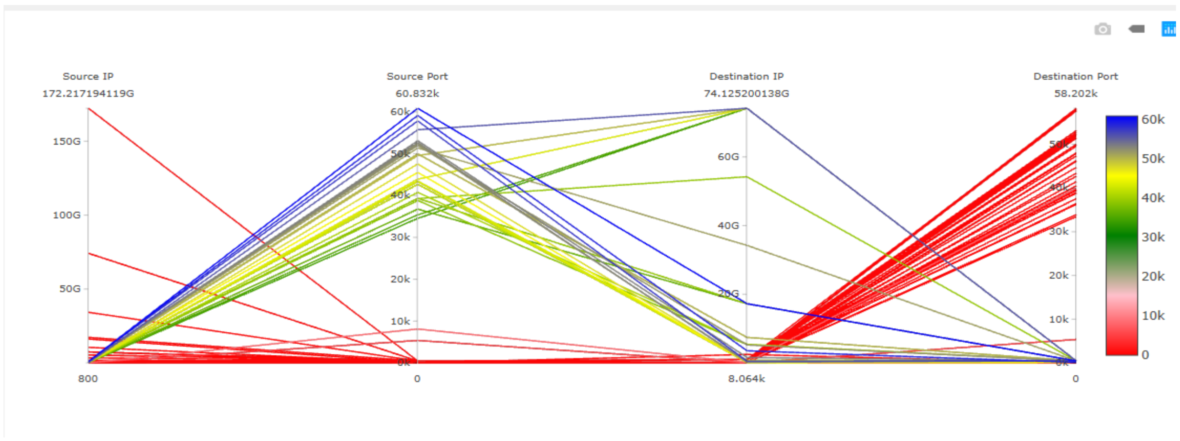


Figure 6.29: Parallel Coordinate Visualization

Box Plot Visualization

Box plot is another data visualizing technique which depicts the mean, median, lower quartile, upper quartile, maximum value and minimum values in the given datasets. It can be used to inspect how our data is distributed and thus used to measure the quality of data too. We plotted the box plot for the total bytes' flow and the total packets flow attributes from database. The various parameters of box plot like mean, median, etc. can be recorded in the database for our specific attributes in the datasets. Then comparing the values of parameters from time to time can reveals whether the data flown through the system was normal or threat. Also, if the maximum value in the datasets is far larger than the average value then this might indicate the something not usual is causing data distribution different than normal one. This is very useful for overall data visualization of system as well as data comparison for two or more values of an attributes for the same another attribute.

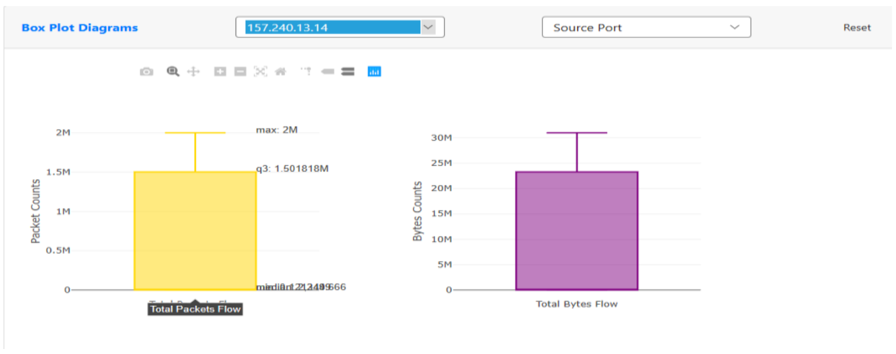


Figure 6.30: Box Plot Visualization

Port Scan visualization

Port Scan visualization is a technique which counts the number of times the specific port used for the flow of data through the system. It is one of a method for anomaly detection in the system. We can manually tell from the visualization that some person is doing port scanning. This visualization can tell us the unusual data flow in the system for specific port. If the port counts is large for a specific port than usual one it can classified as the threat to the system. Not only it can be threat but also may the port had been used for specific huge task like database related tasks.

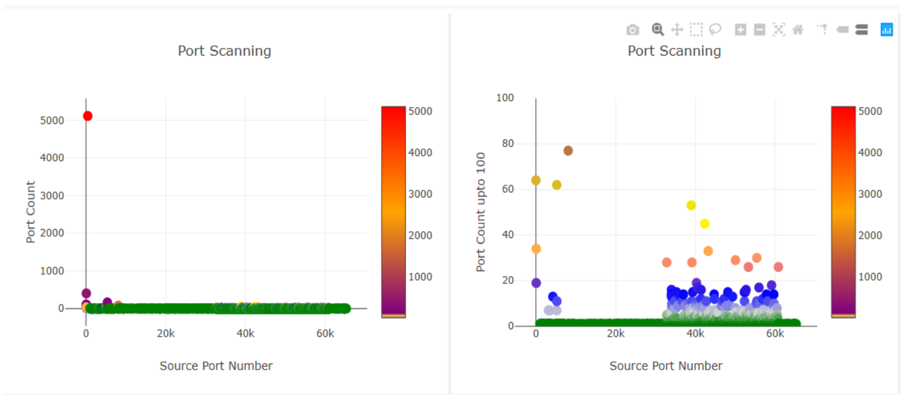


Figure 6.31: Port Scan Visualization

7. RESULT

7.1. Evaluation Metrics

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. True positive and true negatives are the observations that are correctly predicted. We want to minimize false positives and false negatives.

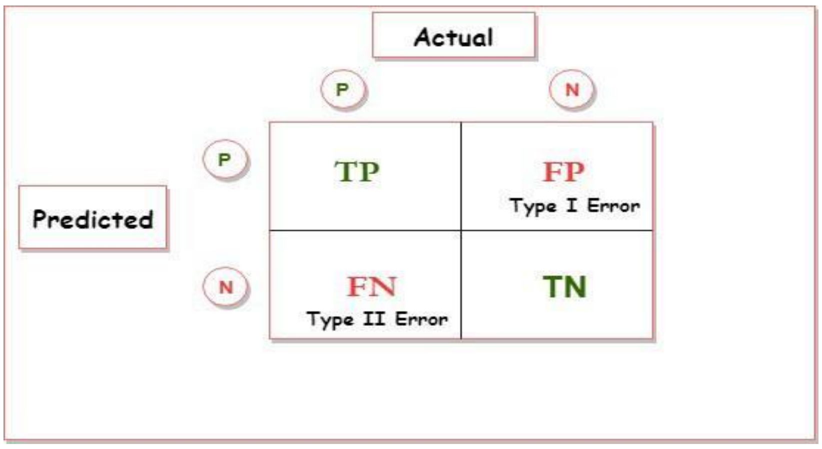


Figure 7.1: Confusion Matrix Representation

True Positives (TP)

These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes.

True Negatives (TN)

These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.

False Positives (FP)

When actual class is no and predicted class is yes.

False Negatives (FN)

When actual class is yes but predicted class in no.

Accuracy

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when there are symmetric datasets where values of false positive and false negatives are almost same. Therefore, the parameters should be observed to evaluate the performance of the model.

Accuracy = TP+TN/TP+FP+FN+TN

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate.

Precision = TP/TP+FP

Recall (Sensitivity)

Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? We have got recall of 0.631 which is good for this model as it's above 0.5.

Recall = TP/TP+FN

ROC curve

A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve, or ROC curve. It is a plot of the false positive rate in x-axis versus the true positive rate y-axis for a number of different threshold values between 0.0 and 1.0.

The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive. The true positive rate is also referred to as sensitivity.

The false positive rate is calculated as the number of false positives divided by the sum of the number of false positives and the number of true negatives.

It is also called the false alarm rate as it summarizes how often a positive class is predicted when the actual outcome is negative. The false positive rate is also referred to as the inverted specificity.

The ROC curve is a useful tool for a few reasons:

1. The curves of different models can be compared directly in general or for different thresholds.
2. The area under the curve (AUC) can be used as a summary of the model skill.

The shape of the curve contains a lot of information, including what we might care about most for a problem, the expected false positive rate, and the false negative rate.

To make this clear:

1. Smaller values on the x-axis of the plot indicate lower false positives and higher true negatives.
2. Larger values on the y-axis of the plot indicate higher true positives and lower false negatives.

Generally, skillful models are represented by curves that bow up to the top left of the plot. A model with no skill is represented at the point [0.5, 0.5]. A model with no skill at each threshold is represented by a diagonal line from the bottom left of the plot to the top right and has an AUC of 0.5. A model with perfect skill is represented at a point[0.0 ,1.0]. A model with perfect skill is represented by a line that travels from the bottom left of the plot to the top left and then across the top to the top right.

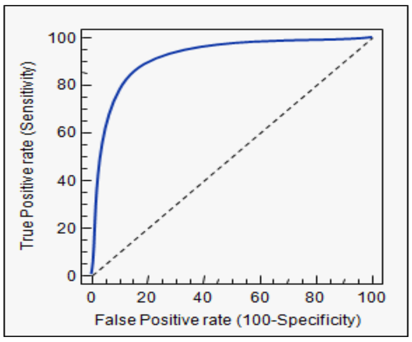


Fig 7.2: ROC curve diagram

7.2. Evaluation of Malicious URL Detection Classifier

The accuracy of the malicious URL detection classifier was found to be 96.44%. It was able to classify most of the URL links correctly. The evaluation of the model was done using confusion matrix.

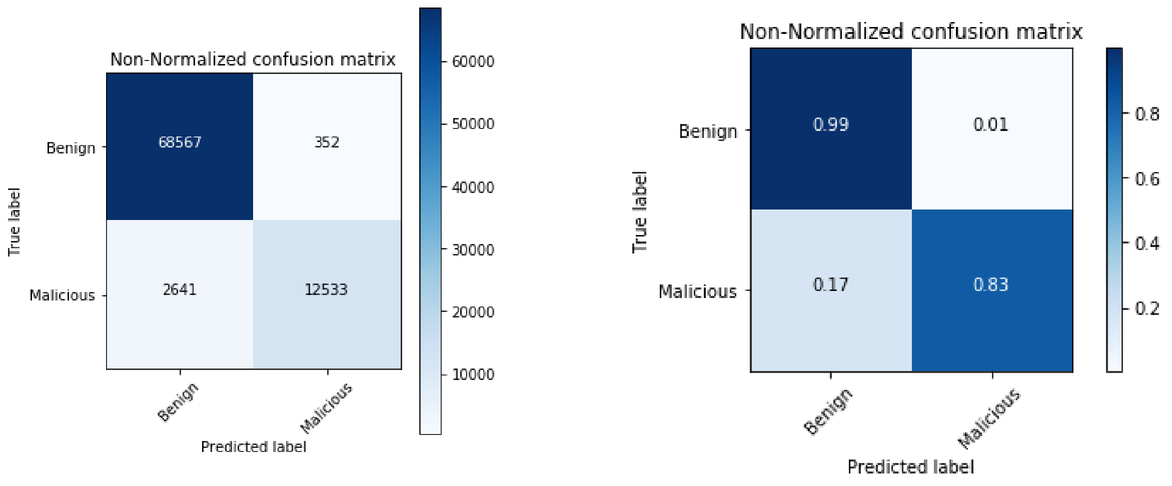


Figure 7.3: confusion matrices for evaluation of malicious URL Detection classifier

The result showed good results. The model was able to classify 0.99% of the benign URLs correctly and 0.83% of the malicious URLs were identified correctly.

Classification Report

	Precision	Recall	F1-score	support
Benign	0.96	0.99	0.98	68919
Malicious	0.97	0.83	0.89	15174
Accuracy			0.96	84093
Macro avg	0.97	0.91	0.94	84093
Weighted avg	0.96	0.96	0.96	84093

Table 7.1: classification report of malicious URL Detection classifier

ROC curve

The area under curve value for this model is 0.990.

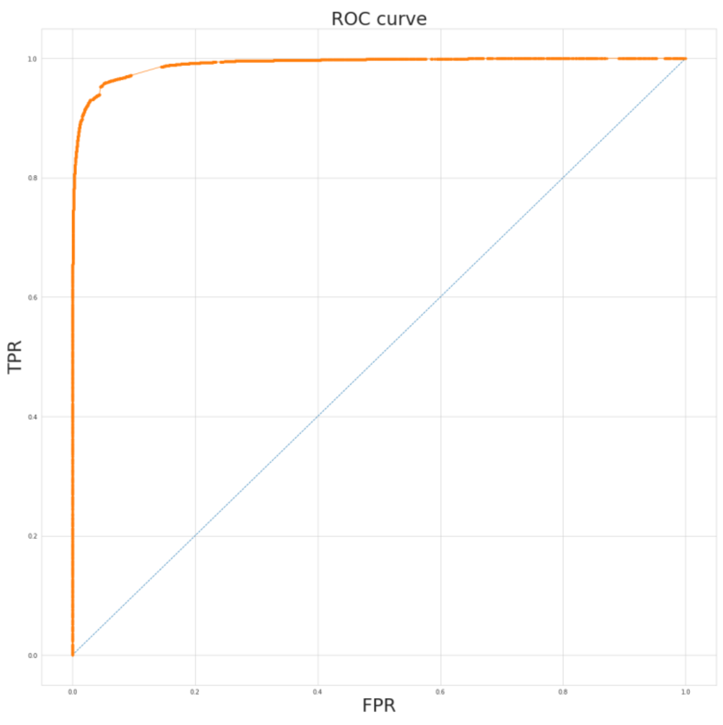


Figure7.4: ROC Curve of URL Detection Classifier

7.3. Real time malicious URL Detection

Open opening the chrome browser with the extension enabled we opened a safe site. The Web server received the link that we opened and predicted the type of URL using the loaded classifier.

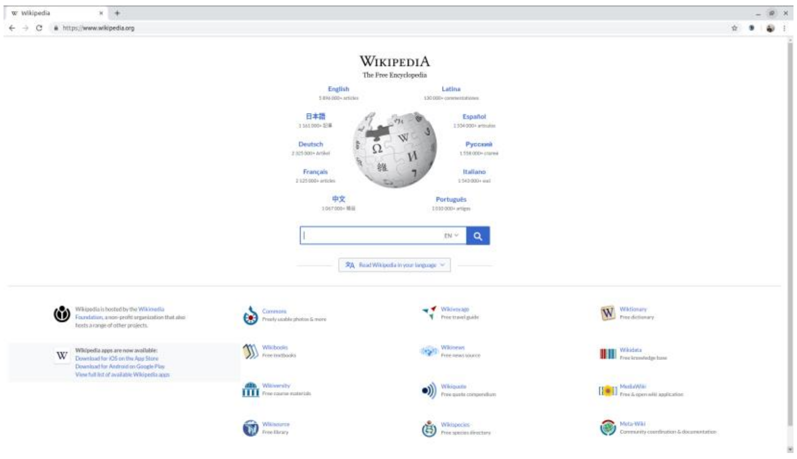
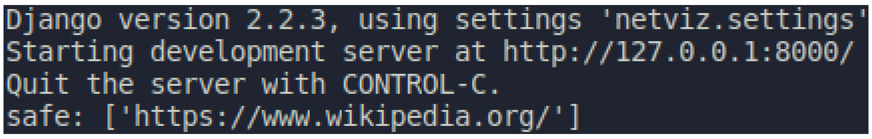


Figure 7.5: Safe URL browsing in Chrome



```
Django version 2.2.3, using settings 'netviz.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
safe: ['https://www.wikipedia.org/']
```

Figure 7.6: Output from the webserver

We can see that as we browse a safe URL https://wikipedia.org/ it was marked as safe and didn't raise any alert notification.

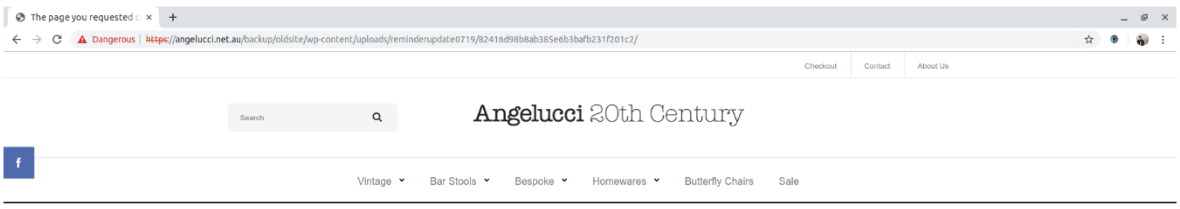
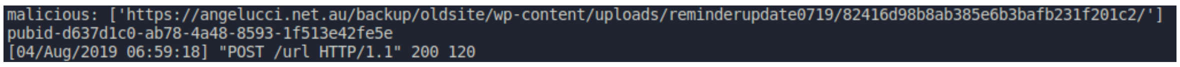
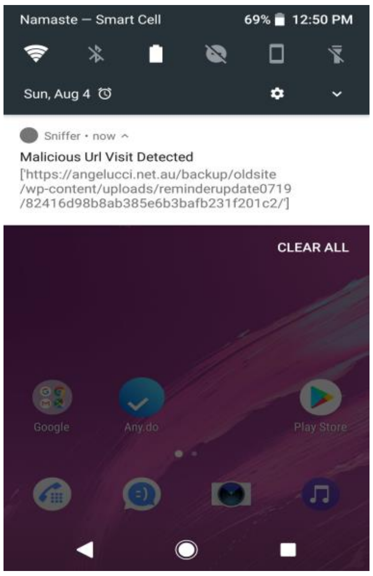


Figure 7.7: Malicious URL browsing in Chrome



```
malicious: ['https://angelucci.net.au/backup/oldsite/wp-content/uploads/reminderupdate0719/82416d98b8ab385e6b3bafb231f201c2/']
pubid-d637d1c0-ab78-4a48-8593-1f513e42fe5e
[04/Aug/2019 06:59:18] "POST /url HTTP/1.1" 200 120
```

Figure 7.8: Output from the web server


Namaste — Smart Cell 69% 12:50 PM

Sun, Aug 4

Sniffer • now

Malicious Url Visit Detected
['https://angelucci.net.au/backup/oldsite/wp-content/uploads/reminderupdate0719/82416d98b8ab385e6b3bafb231f201c2/']

Figure 7.9 Alert Notification in mobile phone

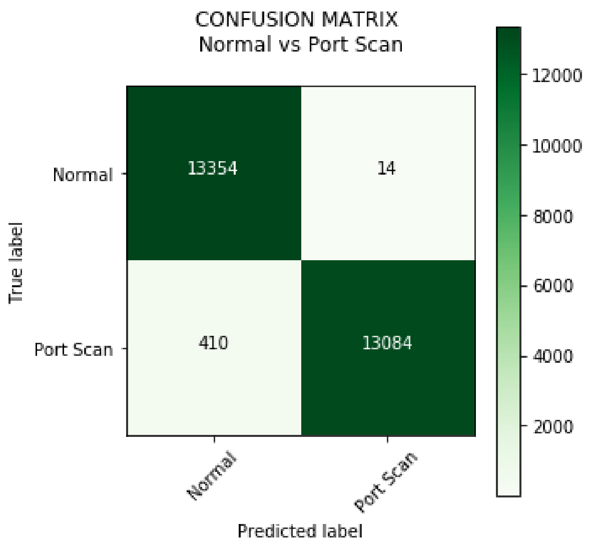
When we tried to browse phishing URL it was classified as malicious and the incident report was sent to the mobile as alert notifications. This proof-of-concept worked successfully in detecting malicious URL browsing.

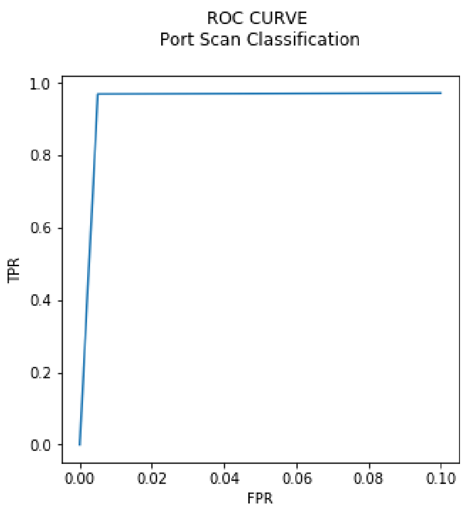
7.4. Evaluation of Network Intrusion Detection Classifier

7.4.1. Binary Classification for Each Attack

Binary classification for normal vs each attack was carried out to see how each attack is handled by the neural network. Figure 7.10 shows the confusion matrix and Roc curve for all the attacks Port Scan, DoS , SSH Brute Force and Malware after being passed through a binary classification neural network. The attacks were classified with minimal false positives and minor occurrence of false negatives . The area under the curve for all the attacks in Roc curve approached to 0.9. Table 7.2 shows the evaluation: accuracy, precision, recall and f1-score for each attack category with maximum results.

Confusion Matrices of Binary Classification for each attack





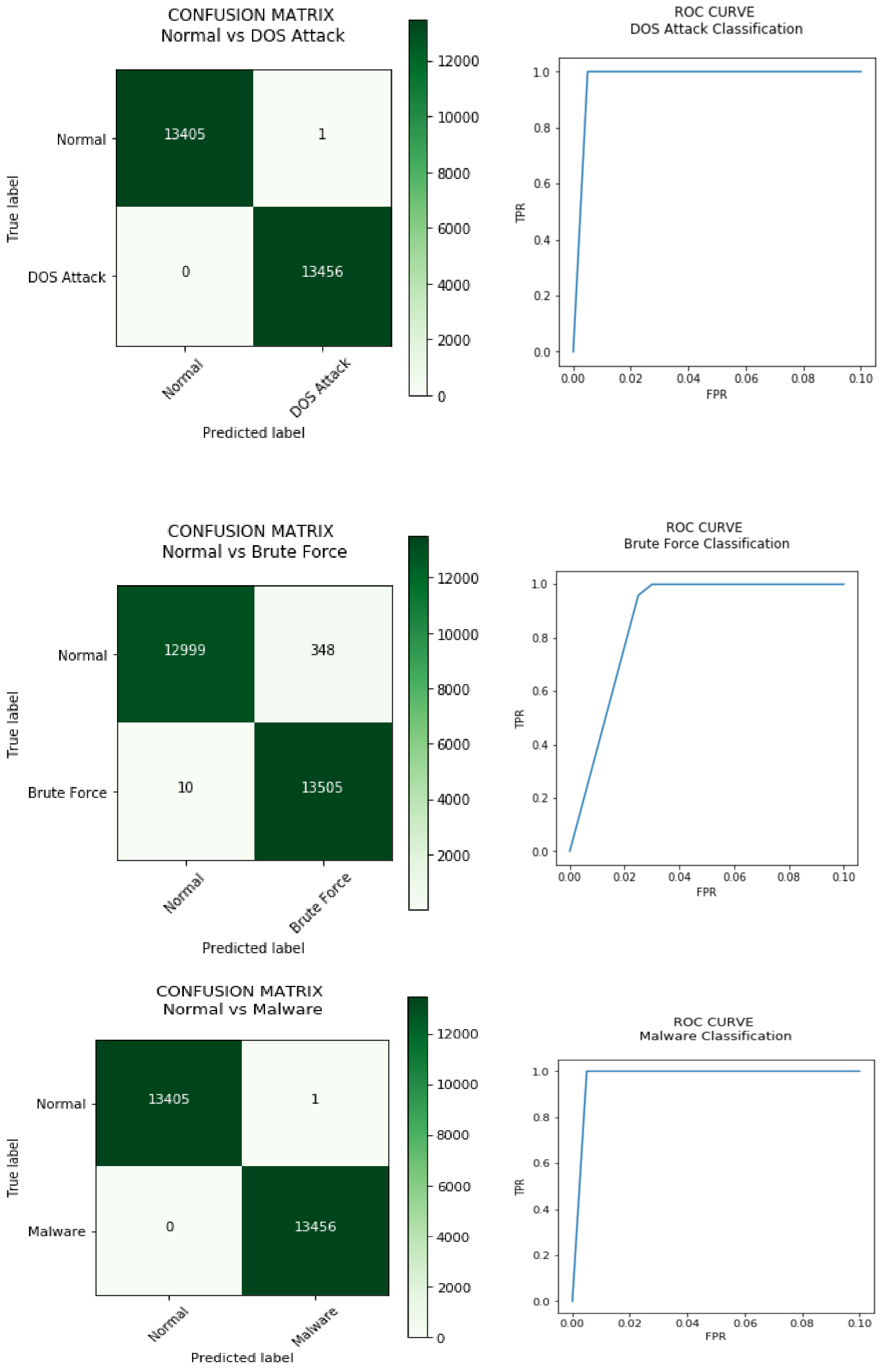


Figure 7.10: Confusion matrix and ROC curve of binary classification for each attacks

Classification Report of binary classification of each attack

Attack Category	Accuracy	Precision	Recall	F1-Score
Port Scan	0.988	0.986	0.990	0.980
DOS	0.990	0.990	1.000	0.996
Brute Force	0.986	0.975	0.992	0.987
Malware	0.990	0.990	1.000	0.990

Table 7.2: classification report of binary classification of each attack

7.4.2. Multiclass Classification

Multiclass classification is a process where all the attributes of the target class is classified distinctly. Multiclass classification was performed using a fully connected sequential neural network with 5 nodes in the output layer. The model was trained until optimal values of the parameters were obtained after 23 epochs with minimum loss as shown in the model loss curve. Figure 7.12 shows the loss curve for train set and validation set with ripples. Evaluation of the 30% test data on our neural network model showed more false negatives than false positives as shown in Figure 7.11. Normal traffic was falsely classified as threat while few threats were classified as normal traffic. The attacks were not misclassified. Table 7.3 shows the evaluation : precision, recall f1-score and accuracy of the model on test set . The accuracy obtained was 93.38%.

Confusion Matrix

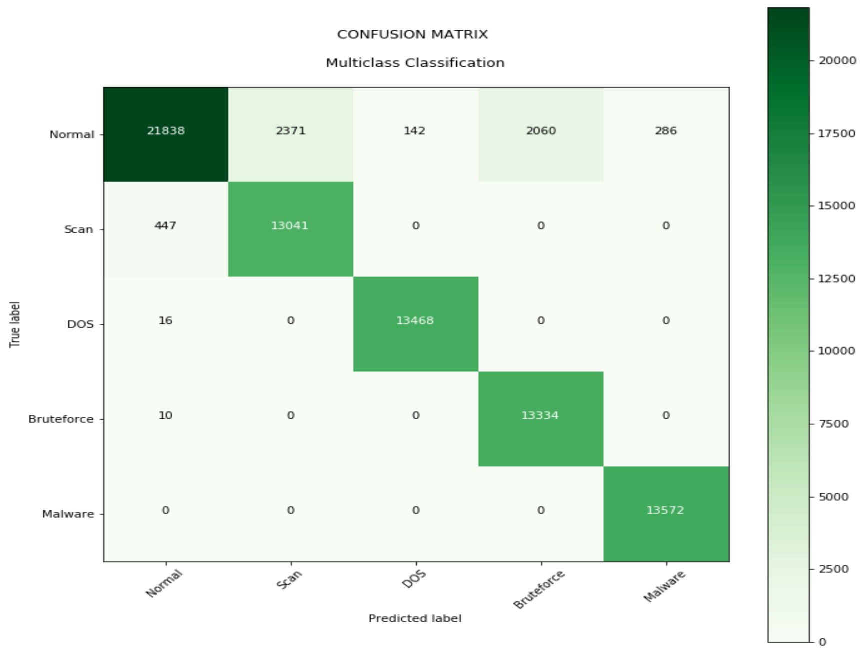


Figure 7.11: Confusion Matrix of Multiclass Classification

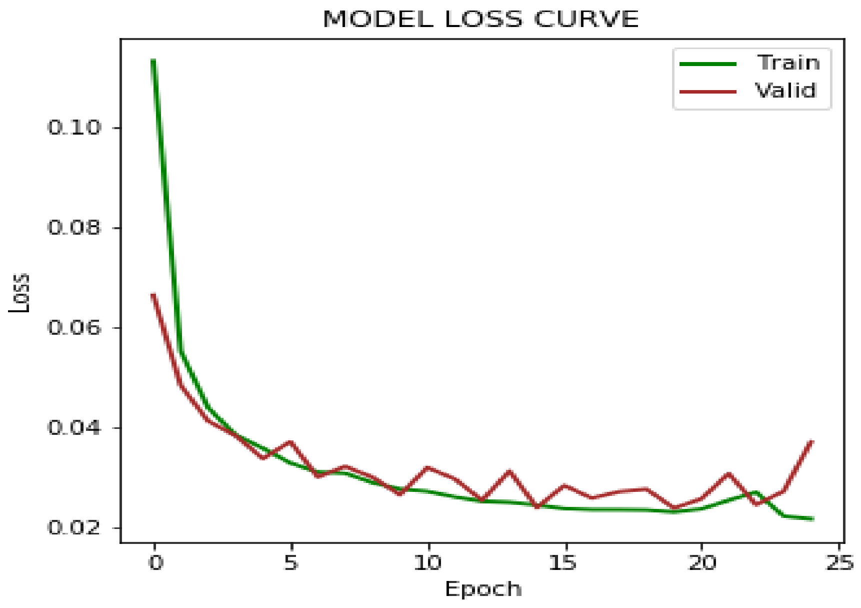


Figure 7.12: Model loss curve

Classification Report for Multiclass Classification:

	Precision	Recall	F1-Score	Support
Normal	0.98	0.82	0.89	26697
Scan	0.85	0.97	0.90	13488
DOS	0.99	1.00	0.99	13484
SSH Brute	0.87	1.00	0.93	13344
Malware	0.98	1.00	0.99	13572
Accuracy				
Macro	0.93	0.96	0.94	80585
Weighted	0.94	0.93	0.94	80585
Accuracy	93.38%			

Table 7.3: classification report for multiclass classification

7.4.3. NIDS Evaluation on CIC IDS 2017 Dataset

The neural network model that was trained with the self generated dataset was used to predict the target values of the standard CIC IDS 2017 dataset. The standard dataset comprised of 8036 data with normal , DoS, Port scan, SSH Brute Force having support 2000 each and Malware having support 36. The confusion matrix of multiclass classification of the standard dataset in shown in the Figure 7.13. The evaluation showed maximum error while classifying DoS attack with few false positives, false negatives and misclassification of the attacks. Since, the model was trained with minimum malware attacks, malware wasn't classified significantly. Table 7.4 shows the evaluation : precision, recall f1-score and accuracy of the model ons standars data set . The overall accuracy obtained was 79.82%.

Confusion Matrix

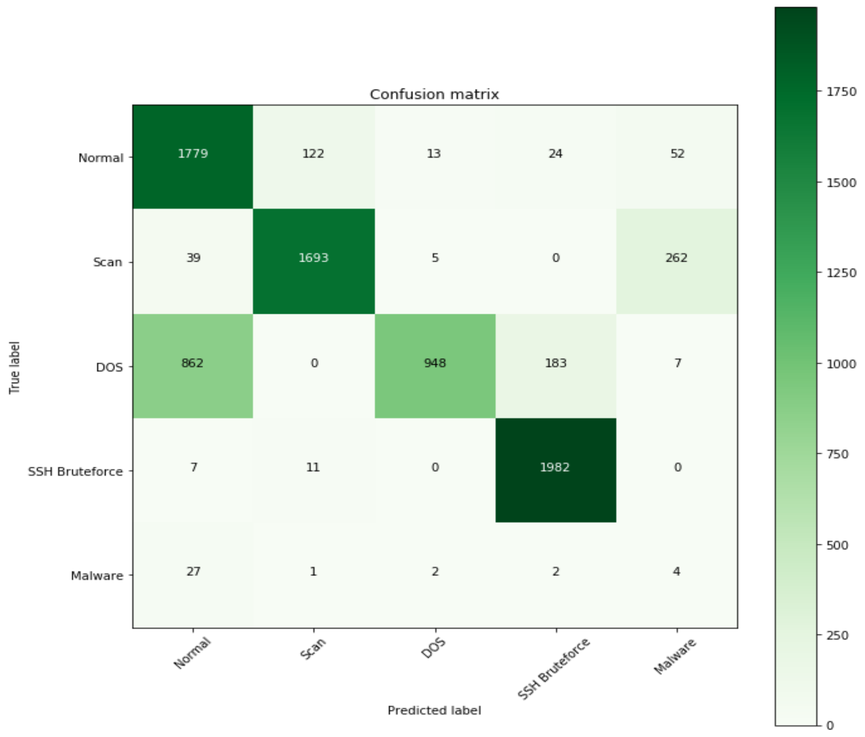


Figure 7.13: Confusion Matrix of Multiclass Classification in CIC IDS 2017 Dataset

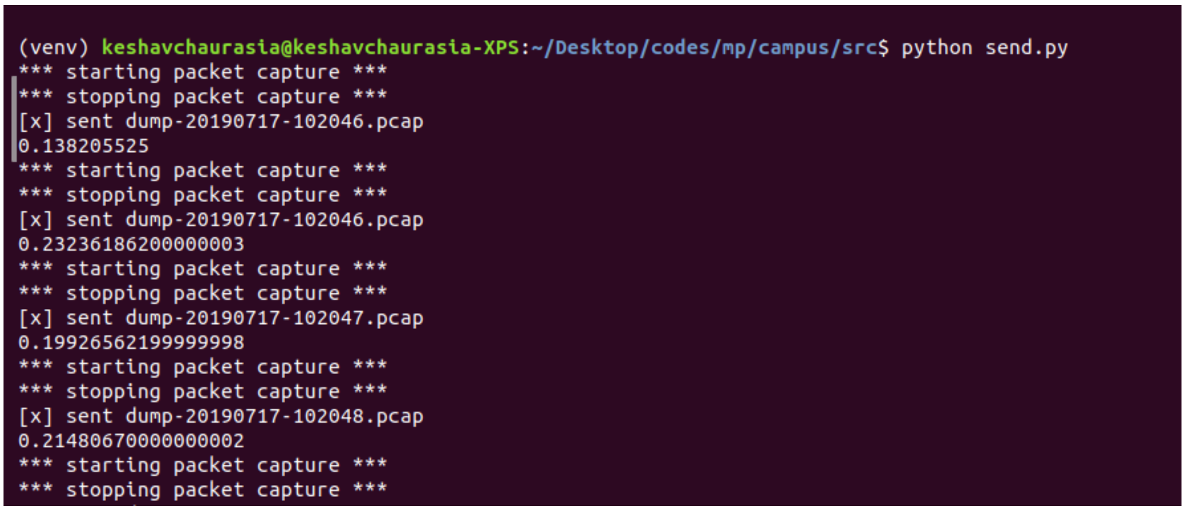
Classification Report for CIC IDS 2017 Dataset:

	Precision	Recall	F1-Score	Support
Normal	0.66	0.89	0.76	1990
Scan	0.93	0.85	0.88	1999
DOS	0.98	0.47	0.64	2000
SSH Brute	0.90	0.99	0.95	2000
Malware	0.01	0.11	0.02	36
Macro	0.93	0.96	0.94	8025
Weighted	0.94	0.93	0.94	8025
Accuracy	79.82%			

Table 7.4: classification report for multiclass classification in CIC IDS 2017 datase

7.5. Real time Network Intrusion Detection

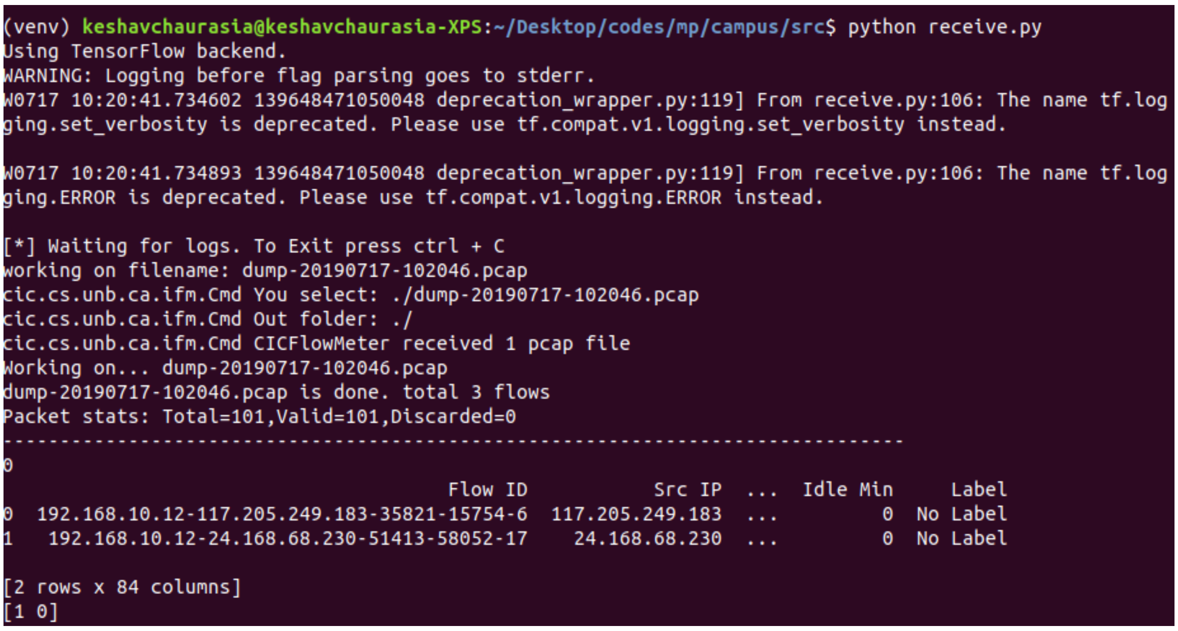
Packet Capturing



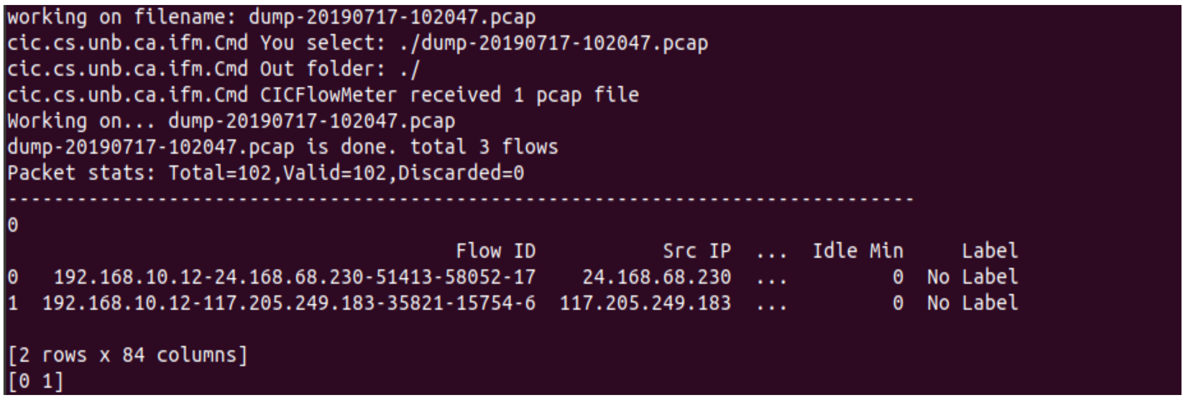
```
(venv) keshavchaurasia@keshavchaurasia-XPS:~/Desktop/codes/mp/campus/src$ python send.py
*** starting packet capture ***
*** stopping packet capture ***
[x] sent dump-20190717-102046.pcap
0.138205525
*** starting packet capture ***
*** stopping packet capture ***
[x] sent dump-20190717-102046.pcap
0.23236186200000003
*** starting packet capture ***
*** stopping packet capture ***
[x] sent dump-20190717-102047.pcap
0.19926562199999998
*** starting packet capture ***
*** stopping packet capture ***
[x] sent dump-20190717-102048.pcap
0.21480670000000002
*** starting packet capture ***
*** stopping packet capture ***
```

Figure 7.14: Output of real time packet capturing

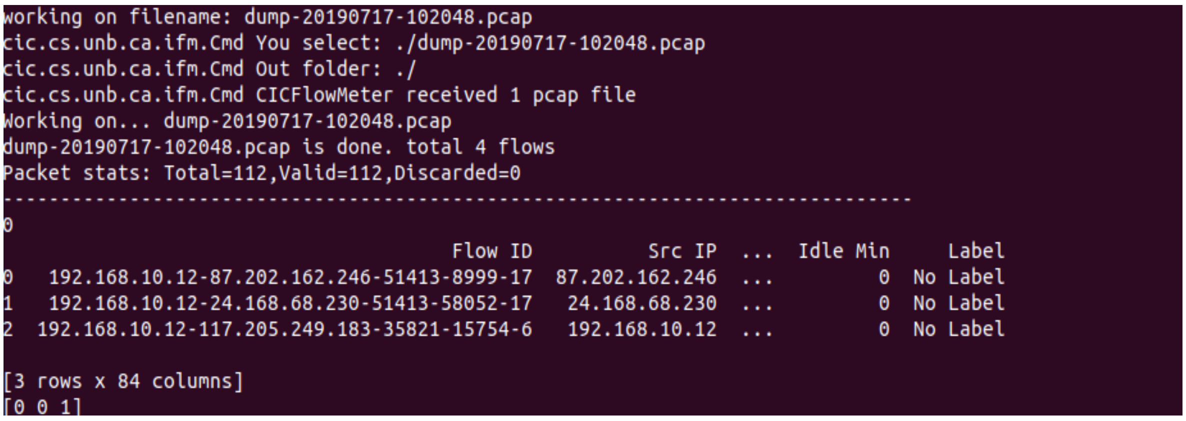
Packet Receiving and Classification



```
(venv) keshavchaurasia@keshavchaurasia-XPS:~/Desktop/codes/mp/campus/src$ python receive.py
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0717 10:20:41.734602 139648471050048 deprecation_wrapper.py:119] From receive.py:106: The name tf.log
ging.set_verbosity is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

W0717 10:20:41.734893 139648471050048 deprecation_wrapper.py:119] From receive.py:106: The name tf.log
ging.ERROR is deprecated. Please use tf.compat.v1.logging.ERROR instead.

[*] Waiting for logs. To Exit press ctrl + C
working on filename: dump-20190717-102046.pcap
cic.cs.unb.ca.ifm.Cmd You select: ./dump-20190717-102046.pcap
cic.cs.unb.ca.ifm.Cmd Out folder: ./
cic.cs.unb.ca.ifm.Cmd CICFlowMeter received 1 pcap file
Working on... dump-20190717-102046.pcap
dump-20190717-102046.pcap is done. total 3 flows
Packet stats: Total=101,Valid=101,Discarded=0
-------------------------------------------------------------------------------
0
                                       Flow ID           Src IP  ...  Idle Min     Label
0  192.168.10.12-117.205.249.183-35821-15754-6  117.205.249.183  ...         0  No Label
1   192.168.10.12-24.168.68.230-51413-58052-17    24.168.68.230  ...         0  No Label

[2 rows x 84 columns]
[1 0]
```



```
working on filename: dump-20190717-102047.pcap
cic.cs.unb.ca.ifm.Cmd You select: ./dump-20190717-102047.pcap
cic.cs.unb.ca.ifm.Cmd Out folder: ./
cic.cs.unb.ca.ifm.Cmd CICFlowMeter received 1 pcap file
Working on... dump-20190717-102047.pcap
dump-20190717-102047.pcap is done. total 3 flows
Packet stats: Total=102,Valid=102,Discarded=0
----------------------------------------------------------------------------
0
                                       Flow ID           Src IP  ...  Idle Min     Label
0   192.168.10.12-24.168.68.230-51413-58052-17    24.168.68.230  ...         0  No Label
1  192.168.10.12-117.205.249.183-35821-15754-6  117.205.249.183  ...         0  No Label

[2 rows x 84 columns]
[0 1]
```



```
working on filename: dump-20190717-102048.pcap
cic.cs.unb.ca.ifm.Cmd You select: ./dump-20190717-102048.pcap
cic.cs.unb.ca.ifm.Cmd Out folder: ./
cic.cs.unb.ca.ifm.Cmd CICFlowMeter received 1 pcap file
Working on... dump-20190717-102048.pcap
dump-20190717-102048.pcap is done. total 4 flows
Packet stats: Total=112,Valid=112,Discarded=0
----------------------------------------------------------------------------
0
                                       Flow ID          Src IP  ...  Idle Min     Label
0   192.168.10.12-87.202.162.246-51413-8999-17  87.202.162.246  ...         0  No Label
1   192.168.10.12-24.168.68.230-51413-58052-17   24.168.68.230  ...         0  No Label
2  192.168.10.12-117.205.249.183-35821-15754-6   192.168.10.12  ...         0  No Label

[3 rows x 84 columns]
[0 0 1]
```

Figure 7.15: Output of real time NIDS

8. CONCLUSION

The data set generated in test bed environment was used in the network intrusion detection system to train the artificial neural network. It detected attacks Port Scan, Dos, Brute force and malware infiltration with an accuracy of 93.38%. The model showed an accuracy of 79.82% when validated using real life data set CIC IDS 2017 developed by Canadian Institute of Cybersecurity(CIC). The real time malicious URL detection system showed an accuracy of 96.44%

9. LIMITATIONS

1. The system is only detects common atacks like port scan, brute force, DoS and malware infiltrations.
2. The system detects attacks but it is not 100% accurate in classifying the attack types.
3. The dataset generated is in testbed environment and consists of only internal intrusion attacks.

10. FUTURE ENHANCEMENTS

1. New attacks can be generated in the dataset generation process and the data set can have external intrusions.
2. Neural Network model can be trained to detect the new atacks.
3. False alarm rate can be minimized.
4. A much vivid and analytical visualization dashboards can be developed.

REFERENCES

1. A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5G mobile and wireless communications: The vision of the metis project," IEEE Commun. Mag., vol. 52, no. 5, pp. 26–35, May 2014.

2. Cisco Visual Networking Index: Forecast and Methodology 2015–2020, published at www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

3. M. Alvarez, N. Bradley, P. Cobb, S. Craig, R. Iffert, L. Kessem, J. Kravitz, D. McMilen, and S. Moore, "IBM X-force threat intelligence index 2017," IBM Corporation, pp. 1–30, 2017.

4. Internet Of Things Statistics from 2019 To Justify The Rise Of Iot
Christo Petrov - https://techjury.net/stats-about/internet-of-things-statistics/

5. C. Kolias, G. Kambourakis, and M. Maragoudakis, "Swarm intelligence in intrusion detection: A survey," Computers & Security, vol. 30, no. 8, pp. 625–642, 2011.

6. A. G. Fragkiadakis, V. A. Siris, N. E. Petroulakis, and A. P. Traganitis, "Anomaly-based intrusion detection of jamming attacks, local versus collaborative detection," Wireless Com-munications and Mobile Computing, vol. 15, no. 2, pp. 276–294, 2015.

7. L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning,"APSIPA Transactions on Signal and Information Processing, vol. 3, 2014.

8. K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," NIST special publication, vol. 800, no. 2007, 2007.

9. Immadisetti Naga Venkata Durga Naveen, Manamohana K, Rohit Verma,"Detection of Malicious URLs using Machine Learning Techniques",International Journal of Innovative Technology and Exploring Engineering (IJITEE)

10. Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2011). Learning to detect malicious urls. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 30.

11. G.P. Zhang, Neural networks for classification: A survey, IEEE Trans.Syst. Man Cybern. C 30 (4) (2000) 451–462.

12. S. Rostami, D. O'Reilly, A. Shenfield, N. Bowring, A novel preference articulation operator for the evolutionary multi-objective optimisation of classifiers in concealed weapons detection, Inform. Sci. 295 (2015)494–520.

13. T. Auld, A.W. Moore, S.F. Gull, Bayesian neural networks for internet traffic classification, IEEE Trans. Neural Netw. 18 (1) (2007) 223–239.

14. S. Dreiseitl, L. Ohno-Machado, Logistic regression and artificial neural network classification models: A methodology review 35 (5–6) (2002)352–359.

15. A. Adebiyi, J. Arreymbi, C. Imafidon, A neural network based security tool for analyzing software, in: Doctoral Conference on Computing, Electrical and Industrial Systems, Springer, 2013, pp. 80–87.

16. G. Liu, F. Hu, W. Chen, A neural network ensemble based method for detecting computer virus, in: 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering, Vol. 1, Aug 2010, pp. 391–393.

17. J. Wu, D. Peng, Z. Li, L. Zhao, H. Ling, Network intrusion detection based on a general regression neural network optimized by an improved artificial immune algorithm, PLOS ONE 10 (3) (2015) 1–13

18. B. Wardman, G. Shukla and G. Warner, ;"Identifying vulnerable web-sites by analysis of common strings in phishing URLs,"; 2009 eCrime Researchers Summit, Tacoma, WA, 2009, pp. 1-13.

19. Xiang et al., "A Feature-Rich Machine Learning Framework for Detecting Phishing WebSites, ACM Transactions on Information and System Security "2011.

20. O'Hara, J. (2007). "Toward a commodity enterprise middleware" (PDF). ACM Queue. 5 (4): 48–55. doi:10.1145/1255421.1255424.

21. "Overview Of Key Routing Protocol Concepts: Architectures, Protocol Types, Algorithms and Metrics". Tcpipguide.com. Archived from the original on 20 December 2010. Retrieved 15 January 2011.

22. Thayumanavan Sridhar (September 1998). "Layer 2 and Layer 3 Switch Evolution". cisco.com. The Internet Protocol Journal. Cisco Systems. Retrieved 2014-08-05.

24. "Packet and Flow Based Network Intrusion Dataset", Prasanta Gogoi1, Monowar H. Bhuyan1, D.K. Bhattacharyya1, and J.K. Kalita2

25. Christensson, P. (2018, May 31). Packet Definition. Retrieved 2019, Aug 4, from https://techterms.com

26. "Wireshark." Wikipedia, Wikimedia Foundation, 28 July 2019, en.wikipedia.org/wiki/ Wireshark.

27. Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In

Proceedings of the 2nd International Conference on Information Systems Security and Privacy(ICISSP 2016) , pages 407-414, Rome , Italy

28. Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun and Ali A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features", In the proceeding of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS, Porto, Portugal, 2017

29. https://www.unb.ca/cic/research/applications.html#CICFlowMeter

30. Breitinger, Corinna; Gipp, Bela; Langer, Stefan (2015-07-26). "Research-paper recommender systems: a literature survey". International Journal on Digital Libraries. 17 (4): 305–338. doi:10.1007/s00799-015-0156-0. ISSN 1432-5012.

31. Luhn, Hans Peter (1957). "A Statistical Approach to Mechanized Encoding and Searching of Literary Information" (PDF). IBM Journal of Research and Development. 1 (4): 309–317. doi:10.1147/rd.14.0309.

32. Manning, C.D.; Raghavan, P.; Schutze, H. (2008). "Scoring, term weighting, and the vector space model" (PDF). Introduction to Information Retrieval. p. 100. doi:10.1017/CBO9780511809071.007. ISBN 978-0-511-80907-1.

33. "TFIDF statistics | SAX-VSM".

34. “What are extensions? - Google Chrome”, Developer.chrome.com, 2017. Online. Available: https://developer.chrome.com/extensions. Accessed: 30- Mar- 2017.

35. “What is pusher?- Google Chrome”, https://pusher-community.github.io/real-time-laravel/introduction/what-is-pusher.html

36. Vinoski, S. (2006). "Advanced Message Queuing Protocol" (PDF). IEEE Internet Computing. 10 (6): 87–89. doi:10.1109/MIC.2006.116.

37. Which protocols does RabbitMQ support?

38. “What is RabbitMQ?”, https://spring.io/blog/2010/06/14/understanding-amqp-the-protocol-used-by-rabbitmq/

39. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html