COMMUNITY DETECTION OF ANOMALIES IN LARGE-SCALE NETWORK WITH DEEP LEARNING

by

Adefolarin Bolaji

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Technology

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. John Springer, Chair

Department of Computer and Information Technology

Dr. James Eric Dietz

Department of Computer and Information Technology

Dr. Jin Kocsis

Department of Computer and Information Technology

Dr. Vetria Byrd

Department of Computer Graphics Technology

Approved by:

Dr. Kathryne Newton

This work is dedicated to my parent Adesola and Sarah Bolaji, and my God-given wife Adebunmi Elizabeth Odefunso; for how the Lord has used them to make me who I am today.

ACKNOWLEDGMENTS

I wish to acknowledge the God of all grace through my Lord and Savior Jesus Christ. “for in Him I live, move, and have my being…” Acts 17:28. The Lord has been so good to me and my family, all the glory on my life and work belongs to Him and Him alone.

I appreciate Dr. John Springer, my advisor, for believing in me, and for giving me a chance. I also acknowledge the help and untiring assistance from my Ph.D. committee, Dr. Vetria Byrd, Dr. James Eric Dietz, and Dr. Jin Kocsis. Don and Cheryl Brier were God-sent to my family. They stood by us as parents during our studies. The following people made a significant impact on my life during my assistantship role at the graduate school, Mummy Debbie Fellure, Ashlee Messersmith (my buddy), Dr. James Mohler, Christal Musser, Jenny Matson, Wei Kao, Victoria Chu, Dr. Julius Eason, Dean Linda Mason, and all the graduate school staff.

I appreciate Kayode and Bunmi Adeagbo for their role and support in making this become a reality. All our family friends in Colorado such as the Jimos, Ajibolas, Talabis, and Amusats. Dr. Dan & Seun Falola, Dr. Gbenga & Adesola Olatunde are also appreciated for standing with us during this program. I thank God for the lives of Prof. Kola and Dr. Mary Ajuwon, they have played very significant roles in my family during this period at West Lafayette. The families of Pastor Ademola and Sis. Bukola Dada’s & Mr. Kenny and Mrs. Folake Ayano, Mr. Deji & Mrs. Lanre Oyewumi, and Aunty Anita Amiaya have been of immense help to us during this program.

I acknowledge the African Christian Fellowship leaders & members for their spiritual and moral support always. Nigerian Student Association at Purdue (which was founded by me and few others) is also appreciated. The families of Mr. Femi Babatunde, Dr. Evidence Matangi, Mr. Chinonso Etumnu, Dr. Toba Omotilewa, Dr. & Dr. Mrs. Jay Uche, and Dr. Saheed Osho are appreciated for all the help they provided to my family at Purdue. Sis OpeAdura Osunbami is especially recognized for her immense help. I also acknowledge Rev Dr & Mrs. Gabriel Kehinde, Prof Femi Ajayi, Dn Kayode & Prof Mrs. Temitope Oyedepo, Prof Longe Olumide, and Prof Seun Kolade for all the guidance and help they provided to me. The family of Prof Tayo and Dr. Lola Adedokun & Pastor Kola and Mrs. Peju Olugbenro are recognized for their input during this period.

I appreciate the family of my siblings, Rev Johnson & Titilade Adeleke, Mr. Kehinde & Yemisi Bolaji, Dr. Tola & Dr. Adeola Bolaji, Bidemi & Lola Ogundare for their strong support. I also appreciate my Aunty, Mrs. Funke Adebimpe for her support always. I acknowledge Mr. M.O.

Raji & Mr. Lateef Adebimpe for their input on my career. I appreciate Gracelink Computech and her associates which include Niyi Odefunso, Ganiyu Waheed, Samson Babalola, Dele Farayola, Ezekiel Babarinde, Engr Damola Eesuola, Mr. Tayo Lawal, Mr. Tajudeen Fasesan, and Mr. Ally Sulaimon for their great help during this time.

I also do appreciate the families of my in-law for their encouragement, worthy of note is the strong help provided by my mother-in-law, Mrs. Paulina Odefunso, and her children, Pastor Wale & Mrs. Peju Adeyemo, Banke, and Niyi Odefunso. I appreciate the family of Pastor Joel & Adeola Oke, our spiritual mentor. I acknowledge the families of Pastor Sanmi Oyedeji, Dr. Felix Adeoluwa Olanrewaju, Segun Ajisafe, Dr. SamGoforth Ajani, Pastor Thomas Egbeleke, Pastor Gbolahan Oyelakin, and Dr. & Mrs. Ade Durodola for their constant follow up and support.

Finally, I appreciate Adebunmi Elizabeth Ojochenemi Odefunso, my beloved wife, for being a strong pillar to my success in this program. I appreciate God for my children Smith, Donald & Catherine, they made me happy always with their outstanding performances.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CAIDA	Center for Applied Internet Data Analysis
D.A.T.A	Discovery Advancements Through Analytics (Laboratory at Purdue).
DDoS	Distributed Denial of Service
HPC	High Performance Machine
IDS	Intrusion Detection Systems
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
MDL	Minimum Description Length
MOGA-OCD	Multi-Objective Genetic Algorithm in Overlapping Community Detection
NIDS	Network Intrusion Detection Systems
AI	Artificial Intelligence
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
CDoA	Community Detection of Anomalies

GLOSSARY

Graph – "A mathematical representation of a set of objects and their relations. We denote a graph G as an ordered pair G = (V, E) where V represents the set of objects (also called nodes or vertices) and E represents the set of relations (also called edges, links or connections)" (Araujo, 2017, p.9).

Community – "The division of network nodes into groups within which the network connections are dense, but between which are sparser" (Newman & Girvan, 2004, p. 1).

Modularity – "The extent, relative to a null model network, to which edges are formed within the modules instead of between the modules" (Barber, 2007, p. 1).

ABSTRACT

The detection of anomalies in real-world networks is applicable in different domains; the application includes, but is not limited to, credit card fraud detection, malware identification and classification, cancer detection from diagnostic reports, abnormal traffic detection, identification of fake media posts, and the like. Many ongoing and current researches are providing tools for analyzing labeled and unlabeled data; however, the challenges of finding anomalies and patterns in large-scale datasets still exist because of rapid changes in the threat landscape.

In this study, I implemented a novel and robust solution that combines data science and cybersecurity to solve complex network security problems. I used Long Short-Term Memory (LSTM) model, Louvain algorithm, and PageRank algorithm to identify and group anomalies in large-scale real-world networks. The network has billions of packets. The developed model used different visualization techniques to provide further insight into how the anomalies in the network are related.

Mean absolute error (MAE) and root mean square error (RMSE) was used to validate the anomaly detection models, the results obtained for both are 5.1813e-04 and 1e-03 respectively. The low loss from the training phase confirmed the low RMSE at loss: 5.1812e-04, mean absolute error: 5.1813e-04, validation loss: 3.9858e-04, validation mean absolute error: 3.9858e-04. The result from the community detection shows an overall modularity value of 0.914 which is proof of the existence of very strong communities among the anomalies. The largest sub-community of the anomalies connects 10.42% of the total nodes of the anomalies.

The broader aim and impact of this study was to provide sophisticated, AI-assisted countermeasures to cyber-threats in large-scale networks. To close the existing gaps created by the shortage of skilled and experienced cybersecurity specialists and analysts in the cybersecurity field, solutions based on out-of-the-box thinking are inevitable; this research was aimed at yielding one of such solutions. It was built to detect specific and collaborating threat actors in large networks and to help speed up how the activities of anomalies in any given large-scale network can be curtailed in time.

CHAPTER 1. INTRODUCTION

1.1 Nature of the Problem

Anomalies are often called outliers because they deviate from normal, standard, or expected patterns (Farias, Fabregas, Dormido-Canto, Vega, & Vergara, 2020). Anomalies in a network are not easy to identify until they have caused significant problems. The continuous and rapid increase in network traffic volumes is making the prevalence and sophistication of attacks more visible (Do and Gadepally, 2020).

The use of traditional intrusion detection and intrusion prevention systems have initially slowed down the rate of cyber-attacks, but the systems are limited in terms of the kinds of attack they can handle. For example, the recent form of distributed denial of service attacks requires more intelligent and robust systems to handle them, especially due to the high volume and rate of attacks experienced in very short durations. Strategies involving different techniques such as Big data, Artificial Intelligence (AI) are evolving rapidly. They are currently being used to overcome the current limitations of traditional intrusion detection and intrusion prevention systems (Kai, Singtel & Balachandran, 2020).

The provision of robust cybersecurity solutions requires the coordination of both machine and human endeavors. For example, the increase in the number of fileless malware attacks as indicated in the CrowdStrike Global Threat Report requires attention and a quick solution (CrowdStrike Inc., 2020). The fileless malware attacks are also known as “living-off-the-land” (LotL) attacks.

These kinds of frequent changes in the attack landscape weaken and delay progress in the cybersecurity domain; many of the existing solutions have been rendered useless or irrelevant by the changes in the attack landscape. This research is proposing a community detection of anomalies (CDoA) solution that makes use of existing anomaly and community detection algorithms in identifying relationships and patterns of anomalies in a large-scale network. A simplified diagram of anomaly and community detection procedure is as shown in Figure 1.1

The diagram shows that a hidden *Change in Network Property* will influence the behavioral pattern of the network traffic from normal to abnormal. For example, packets with manipulated Time-To-Live (TTL) can cause abnormal behavior in the network. This change in network

property would influence the process that generates the *Observed Network Graphs* which represents the network property's factual information. The summarized graphs which are based on the Graph Statistics become the inputs to some *Anomaly Detector* system. The output of the detector will lead to conclusions as to when the change in the network property is abnormal. The assumption in this scenario is that a 1:1 mapping exists between the graph statistics and the network property. That is, no other property of the network is affecting the observed statistical values.

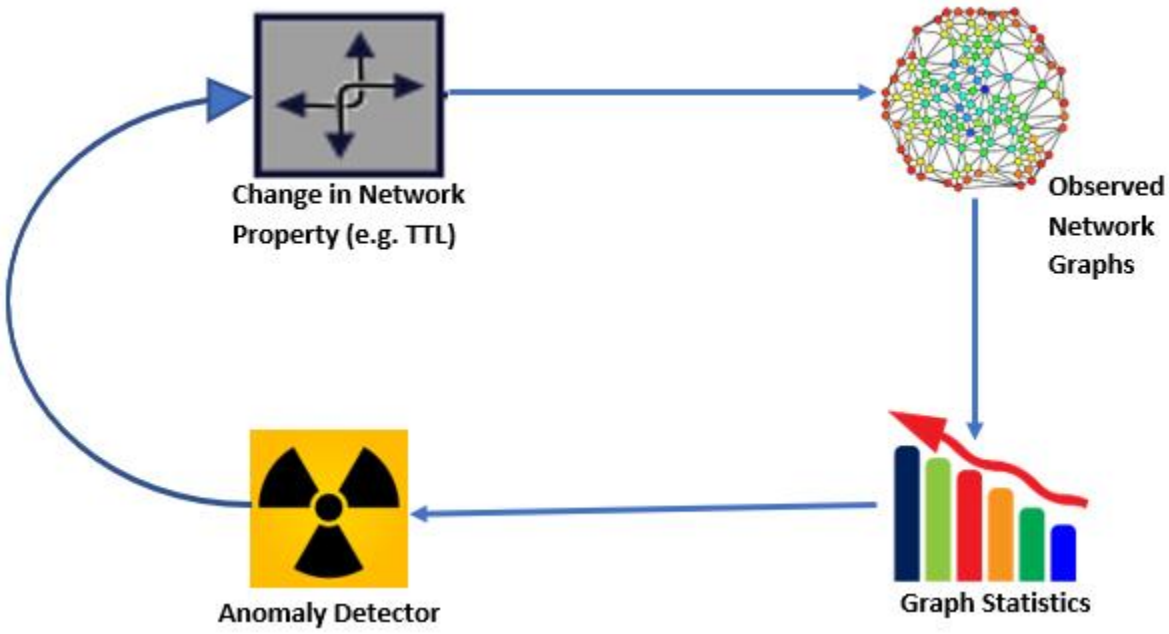


Figure 1.1: A simplified diagram of anomaly and community detection process

1.2 Statement of Problem

Recent growth in cyber-attacks threatens personal credibility, security of nations, and the rights of ownership on intellectual properties globally. Network security-related incidences are great threats to private/personal data which include cards' credentials, personal health and financial records, and other confidential information. All these are hijacked or stolen daily. Growth in cybercriminal activities brings destruction to innovations, disruption to the trading of stocks and government agencies by using ransomware to steal sensitive and crucial data (Zurier, 2021).

The rate at which devices are now being connected on the internet is related to the rate at which adversaries and cyber-attacks increase. The adversaries are launching more attacks by the

day and getting more skillful and successful in their plots (Hoque, Bhuyan, Baishya, Bhattacharyya, & Kalita, 2014).

Available research indicated the inability of the existing defenses to keep commensurable pace with the recent cyber-attacks (Reddy & Reddy, 2014). Sophisticated methods that combine human and artificial intelligence (AI) in detecting anomalies and patterns are required as the world experience data explosion (bigdata) through the proliferation of connected devices and automation (Berman, Buczak, Chavis, & Corbett, 2019). To address this problem, a robust technique for detecting communities of anomalies in large-scale networks with deep learning techniques was implemented.

1.3 Research Questions

The research answered the following related questions on community detection of anomalies with deep learning techniques:

(i) Given a large-scale network with millions or billions of nodes, what level of accuracy of anomaly detection can be achieved with the CDoA model using deep learning long short-term memory (LSTM)?

(ii) Can we identify strong communities of anomalies in a large-scale network using the CDoA model with the Modularity metric as a measure?

1.4 Hypothesis

The following hypotheses were proposed for this study:

1. **H_0**: The use of deep learning LSTM to detect anomalies in large-scale Internet traffics dataset with more than 20 billion packets will yield more than 10% each of mean absolute error and root mean square error, using time step size, batch size, and epoch of 100, 50 and 100 respectively.

H_A: The use of deep learning LSTM to detect anomalies in large-scale Internet traffics dataset with more than 20 billion packets will yield less than 10% each of loss, mean absolute error, validation loss, and validation mean absolute error using time step size, batch size and epoch of 100, 50 and 100 respectively.

2. **H_0**: When the quality function is used as a measure, the average modularity value of identified communities of anomalies in large-scale Internet traffic with billions of packets will be less than 0.6.

   H_A: When the quality function is used as a measure, modularity values of identified communities of anomalies in large-scale Internet traffic with billions of packets will be higher than 0.6.

1.5 Significance of Problem

Recent attacks on large cyber networks demonstrate a need for enhanced and combined solutions for network segmentation and monitoring. This approach will help cybersecurity specialists to alienate specific sections of network communities for proper investigation and control of network-based cyber threats.

The community detection of anomalies (CDoA) model will help cybersecurity specialists to provide efficient and scalable solutions to prevalent cyber threats which include Botnets, LotL, and distributed denial of service attacks (DDoS). Some of these cyberattacks are gaining momentum with the use of proliferating IoT devices.

1.6 Statement of Purpose

From works of literature, deep learning techniques have not been featured frequently in community detection processes but have in anomaly detection (Gao, Song, Wen, Wang, Sun, Xu, & Zhu 2020; Farias *et al.*, 2020; Maimo, Gomez, Clemente, Perez, & Perez, 2018; Malhotra, Vig, Gautam, & Agarwal, 2015; Shipmon, Gurevitch, Piselli, & Edwards, 2017). Recently, deep neural networks have gained widespread attention with methods such as kernel machines in numerous important applications. Both feedforward (acyclic) neural networks (FNNs) and recurrent (cyclic) neural networks (RNNs) have been very popular in deep neural networks (Schmidhuber, 2015). According to Alla &Adari (2019), anomaly detection types include supervised anomaly detection, semi-supervised anomaly detection, and unsupervised anomaly detection. All these include the development of suitable models for detecting anomalies, training the models, and testing the models with targeted datasets.

To evaluate these models, a confusion matrix is often used along with accuracy, recall, F1 score, and precision (Alla &Adari, 2019); a 2x2 confusion matrix has historically been sufficient to evaluate anomaly detection models using true positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Deep learning anomaly detection models that have been developed and used successfully include Isolation Forest, Support Vector Machine (SVM), Convolutional Neural Network (CNN), Variants of Auto-Encoder, Deep Belief Networks (DBN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) (Berman, Buczak, Chavis, & Corbett, 2019). A study on "time series anomaly detection" shows RNN to be more effective in handling false positives (Shipmon, Gurevitch, Piselli, & Edwards, 2017). However, strengthening defensive measures against threat actors requires additional efforts such as combining the RNN with a community detection approach. This gave birth to the CDoA model.

Community detection generally employs the use of graph theory to solve complex problems (Liao, Deng & Wang, 2019). For example, using partitional clustering essentially involves the identification of different communities in a given network and minimizing the loss function according to the distances that exist among the points and/or identified communities (Fortunato, 2010). This technique is utilized by k-means, minimum-k clustering, k-medoids, and other classical algorithms. Its drawback is its requirement for the prior specification of a certain number of clusters as inputs; in a real-world network, this may not be possible. A hierarchical clustering algorithm was developed to overcome this drawback. The algorithm has two popular classes known as Agglomerative and Divisive algorithms (Fortunato, 2010). The agglomerative class was used for community detection in this study. It recursively merged nodes with high similarity to form communities. Using this together with the LSTM anomaly detection model significantly helped in solving high-dimensional and complex security challenges.

1.7 Scope

This research's scope was based on the following:

1. Detect anomalies in a given real-world large-scale cyber network using a deep learning algorithm.
2. Identify subnetworks of anomalies in the given network using a community detection algorithm.

3. Formulate an extended modularity metric to define the quality function for the identified subnetworks.
4. Confirm relationships among the anomalies in the identified subnetworks.
5. Evaluate CDoA Model.

1.8 Assumptions

The following assumptions were made for this study:

1. The same execution environment was used for the anomaly and community detection in the given large-scale network dataset.
2. IP address represents the network nodes which could represent either source nodes or target nodes in this study.
3. Center for Applied Internet Data Analysis (CAIDA) dataset was used to represent a real-world large-scale network.
4. The network traffic dataset was not manipulated under any circumstances.
5. The size and number of the community of anomalies detected were not known a priori.
6. The variety and size of the network traffic anomalies detected were not known a priori
7. The results of this study which was carried out with the use of existing algorithms in related studies are genuine, and this study was based on this assumption.

1.9 Limitation and Delimitation

The limitations and delimitation of this research were:

1. One limitation of this study is that the speed of execution and analysis for this study varied based on the available high-performance machine (HPC) infrastructure provided by Purdue University.
2. The only metric that was taken into consideration to evaluate the strength/structure of the communities that were discovered is modularity.
3. Community detection in this research is limited to only overlapping and non-overlapping networks.
4. The network traffic anomalies discovered were limited to the traces provided by CAIDA.
5. Weighted and directed graphs only were used in this study.

6. Multiple links from a node to the same destination were treated as redundant and were therefore considered as a single link.

1.10 Summary

The statement of problems and the research questions that were answered in this dissertation were presented in this chapter. The chapter started with the introduction and explained the scope, purpose statement and the significance of this research. The chapter also identified specific limitations, delimitations, and assumptions of this study.

CHAPTER 2. REVIEW OF LITERATURE

This chapter presents an overview of relevant works of literature that were reviewed in this research. It gives a general and summarized introduction to the concepts of anomalies and detection approaches. The chapter zoomed in on Botnet (as an example of malware attack that is common in a large-scale network), IoT (Internet of Things) devices, and network traffic anomalies. Deep learning approaches to anomaly detection, community structures/types, community detection techniques, and algorithms were also discussed.

2.1 Introduction

Application of community detection of anomalies in large networks has the possibility of alleviating cyber-attacks in the global arena. This application leveraged existing deep learning anomaly detection algorithms, especially those that have been successfully used to address problems in given real-world complex networks.

2.2 Anomaly detection in Large-scale Network

Anomalies in any domain of human endeavors are usually not friendly. An anomaly was defined intuitively as "a surprising or unusual occurrence" and as a deviation from the normal (Marteau, 2021; Farias *et al.*, 2020; Noble & Cook, 2003 p.632). A system with outliers in its data output (or input) generally raises concern that requires further investigation. Some systems use anomaly scores to determine if anomalies exist in a network based on previously set thresholds on an intrusion detection system (Marteau, 2021; Peddabachigari, Abraham, Grosan, & Thomas, 2007).

Every standard system is expected to behave and exhibit some predefined set of functions, deviation from such expectations are regarded as anomalies. An adding machine that generates number-three as the result of the addition of double number-two (i.e. 2+2 = 3) is anomalous in its operation. Anomalies are usually caused by hidden factors that can be discovered by further diligent research. Further investigation into the causes of anomalies can expose the factors and reasons behind such anomalies. For example, graph solution has been used in community detection

to investigate and detect anomalous subgraphs in graphs that have single attributes (Jie, Wang, Chen, Li & Wu, 2020; Shao, Li, Chen & Chen, 2018; Noble & Cook, 2003, p.634).

Depending on the field of consideration and point of view, anomalies can be categorized and expressed based on various related variables and thresholds. According to this thesis, anomalies were categorized to be based on expectation, impact, and time. In other related studies, anomalies have also been categorized to be based on a data point, pattern, and context. Anomaly detection styles are supervised anomaly detection, semi-supervised anomaly detection (Marteau, 2021), and unsupervised anomaly detection (Alla and Adari, 2019). The category of anomalies that is related to this study is based on nodes and edges and the style adopted is semi-supervised anomaly detection.

The sensitive nature of data handling in the present age requires anomalies to be prevented at all costs. In situations where prevention is not possible, greater efforts should be made to detect or notice anomalies on time before they cause irreversible or untold problems in any network of concern. Computer networks have become an essential tool since their inception; they have been playing significant roles in the daily activities of almost everybody in the world today. A computer network was described as a model of human transactive memory (Wegner, 1995, p.319). Computer networks have been very significant in data and resource sharing. For example, the use of mobile communication technologies for telecommunication purposes is based on a computer network; the voice or text signals from the phones are sent as network traffic on dedicated networks. According to Katz & Aakhus, 2001, p.3, "Mobile communication technologies are already modifying well-established communication patterns, amplifying and substituting for them." Maintaining a computer network and its traffic against the disruption that could be caused by anomalies requires constant effort and research.

2.2.1 Network traffic anomalies

Network traffic anomalies have the potential to disrupt and destabilize the operation and activities of any organized system. Network traffic anomalies can be traced to faults in the network devices or intentional disruptions on the devices through cyber-attacks such as DDOS (Zhou & Li, 2019). Cyber-attacks are usually carried out with malicious intent by adversaries that are within or without an organization. Cyber-attacks threaten businesses, government entities, and individuals' Intellectual Property (IP) and Personal Identity Information (PII) for espionage and monetary gain.

Variants of cyber-attacks are still evolving and becoming more sophisticated and efficient in the last 30 years. These threats are growing and becoming more prevalent every day (Gupta, Tewari, Jain, & Agrawal, 2016). Cyber-attacks are successful when a cyber threat is used to exploit known vulnerabilities in any of the devices on the network. The world of cyber networks is evolving fast in recent times because of diverse IoT devices that are being manufactured and connected on the Internet. As the network evolves, the data generation increases, as a result, the need for network and data security increases. An example of the common attacks that thrive on IoT is known as the Botnet attack (Hussain, Abbas, Fayyaz, Shah, Toqeer & Ali, 2020).

2.2.2 Anomalies caused by Botnet Attack and IoT devices

Internet of Things (IoT) was reported to have fueled one of the biggest attacks in Internet history; "A giant botnet made up of hijacked internet-connected things like cameras, lightbulbs, and thermostats was used to launch the largest Distributed Denial of Service (DDoS) attack ever against a top security blogger in October 2016." (Greene, 2016). A botnet is a cyber-attack that makes use of networked compromised devices to perform malicious remote operations in an organized environment. It is a group of compromised computers that are running one or more computer application programs that are being controlled and manipulated only by the owner of the software source called Bot Controller or Herder (Sikorski & Honig, 2012).

A botnet attack consists of a master that covertly controls the activities of compromised devices or systems. A Bot Herder or Bot Master gives commands to the compromised devices which are referred to as a robot, bot, zombie, or a drone, the devices are usually compromised via a Trojan (Marcus, 2013). "Some attackers have botnets of thousands of compromised machines under their control and use the IP addresses of the compromised hosts as an underground Internet currency, with stealth routines to hide them from prying eyes" (Rhodes-Ousley, 2013).

Information gathering on Botnet involves "the process of creating a blueprint or map of an organization's network and systems, which are attackable by a botnet. It also involves determining the target systems, applications, or physical locations of the botnet target." This will then lead to the use of non-intrusive methods in gathering information about the botnet (Kimberly, 2010). All the operations performed by Botnet are generally covert by nature.

Botnets have been known to cause many of the catastrophic cyber-attacks in recent times (Greene, 2016). Anomalies are not uncommon in any environment where botnets are in operation.

In other words, botnets are one of the significant sources of network anomalies. The major problem of a botnet is that it can be in operation on a network for long without being noticed or discovered except there are strong measures for its discovery and prevention.

Security researchers have warned for years that poor security for IoT devices could have serious consequences. “Botnets made up of compromised IoT devices are capable of launching distributed denial-of-service attacks of unprecedented scale” (Constantin, 2016). Recent Juniper Research revealed that the number of IoT-connected sensors and devices will exceed 50 billion in 2022, up from 21 billion estimated for 2018; this will be a rise of about 140% within four years (Sorrell, 2018). Figure 2.1 shows the split distribution of 51 Billion IoT connected units by 8 key regions in 2022. North America and West Europe take the largest shares of the distribution while Africa & Middle East and the rest of Asia Pacific will not exceed the 1 billion-barrier by the end of the shown forecast period (Sorrell, 2018).

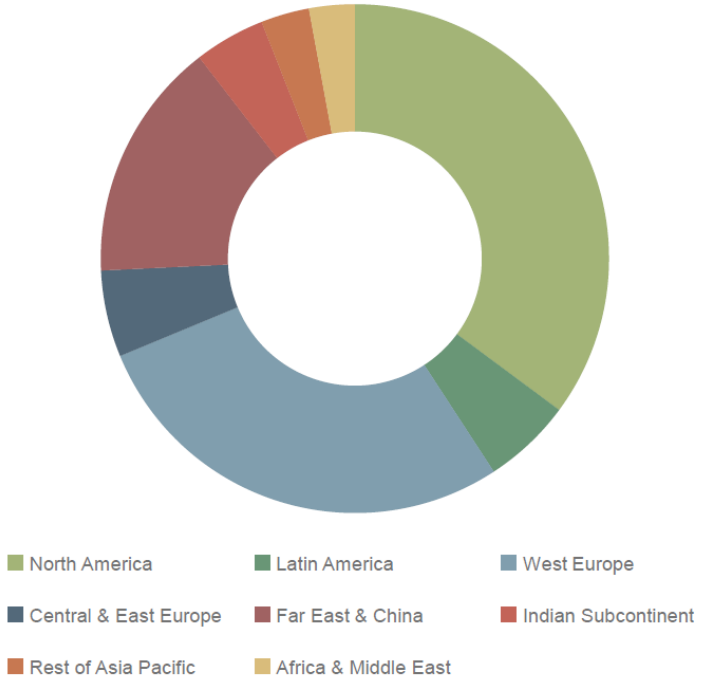


Figure 2.1: The split distribution of 51 Billion IoT connected units by eight key regions in 2022. Source: (Sorrell, 2018)

According to Grau (2015, p. 52), IoT devices can be subjected to various attacks that are categorized into three parts; the ‘take control’ kind, meaning unauthorized applications or actions

that are not permitted by the owner is executed and this may lead to serious incidents. The 'steal information' kind of attack means that hackers sniffed data transmitted and gain private information like location and personal data. The third way of attempting an attack on the IoT device is to disrupt its services; this prevents the IoT device from executing a normal operation, stopping its function, causing incidents like a non-stoppable auto vehicle, or insulin pump not acting while it should.

Research on anomaly detection is very important because it is aimed at solving many problems that exist in various application domains. Many of the techniques that were developed over time for detecting anomalies have been somewhat specific to application domains, while some were developed for more generic purposes (Chandola, Banerjee & Kumar, 2007).

Several reports of IoT devices being routinely hacked and used as weapons in launching big-scale cyber-attacks due to poor security measures and insecure encryption mechanisms in IoT infrastructures have called for a proactive measure in tackling security-related issues with IoT devices generally. For instance, the reported massive DDoS attack that occurred on October 21, 2016, against Dyn servers "brought down much of America's internet. It affected many sites including Twitter, Spotify, PayPal, the Guardian, Netflix, Reddit, CNN, and many other websites" (Raj, 2016). This section describes the previous works on anomaly detection and IoT.

Gendreau & Moorman (2016) proposed a safeguard solution to networks by detecting unauthorized intruders within the constraints of each type of device or subnetwork ahead of information leakage incidents. The proposed solution presented "a survey of Intrusion Detection Systems (IDS) using the most recent ideas and methods proposed for the IoT." The survey tried to separate "IDS platform differences and the current research trend towards a universal, cross-platform distributed approach." This was done by historical examination of intrusion detection systems to provide better understanding and illustration.

IoT devices have some vulnerabilities to different types of attacks, such as routing/insider attacks. Bostani and Sheikhan (2017) proposed "a novel real-time hybrid intrusion detection framework. The framework consists of anomaly-based and specification-based intrusion detection modules for detecting two well-known routing attacks in IoT". The two routing attacks are known as a sinkhole and selective-forwarding attacks. The specification-based intrusion detection agents and anomaly-based intrusion detection agents were used; the agent employed the unsupervised optimum-path forest algorithm for projecting clustering models by using incoming data packets.

The agent used in this study was based on the MapReduce architecture (Bostani and Sheikhan, 2017).

One of the increasingly popular solutions applicable for network intrusion detection systems (NIDS) is the Neural Networks (Mirsky, Doitshman, Elovici & Shabtai, 2018). Neural networks have a very suitable capability of learning complex patterns and behaviors, this capability has made them a useful tool for differentiating between normal traffic and network attacks. One of the major limitations of neural networks is the number of resources needed to train them in a supervised manner. Mirsky *et al.* (2018) tried to overcome the limitation of neural networks by proposing "Kitsune: a plug and play NIDS, which can learn to detect attacks on the local network, without supervision, and in an efficient online manner".

The "need to develop new methods for detecting attacks launched from compromised IoT devices and differentiate between an hour and millisecond long IoT-based attacks" led to another study by Meidan, Bohadana, Mathov, Mirsky, Breitenbacher, Shabtai & Elovici (2018). The study proposed and empirically evaluated a novel network-based anomaly detection method. The method "extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic emanating from compromised IoT devices". Anomaly-based approaches to intrusion detection are one of the major approaches in use today. It has the potential to detect a zero-day attack and other new forms of attack (Harish, 2016)

2.2.3 Anomaly Detection approaches in Network Traffic

Many methods have been used to address the issue of anomaly detection in network traffics. For example, deep learning methods have been used for anomaly detection in social networks by utilizing multimodal data and multidimensional networks (Chaabene, Bouzeghoub, Guetari & Ghezala, 2021). Another method for anomaly detection in a network is graph-based. Existing research has classified graph-based anomaly detections into two broad categories, "white crow" and "in-disguise" as shown in Figure 2.2 (Chen, Hendrix & Samatowa 2011, p. 2). Detected nodes, edges, subgraphs are classified as white crow anomalies while unusual patterns in the network which comprises uncommon nodes and entity alterations are the indicators of an in-disguise class of anomalies.

The CDoA model focused on the later path of anomaly detection. Further research on white crow anomalies has been carried out to identify various types of anomalies (Moonesinghe & Tan,

2006; Sun. Qu, Chakrabarti, & Faloutsos, 2005; Hautamäki, Kärkkäinen, & Fränti, 2004; Noble & Cook, 2003; Lin & Chalupsky, 2003). In-disguise anomalies were further researched by Shetty and Adibi (2005) and Eberle & Holder (2007).

The significant role of computing networks has required the need to develop means of protecting specific networks and their components. Anomalies in the network traffic are indicators of the existence of background or not-so-obvious challenges facing the network. The majority of the network anomalies are usually caused by cyber-attacks even in situations where preventive measures have been put in place on the network. Such preventive measures include the use of firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), and other security measures. Many methods are now being used to detect network traffic anomalies on top of these preventive measures (Marteau, 2021).

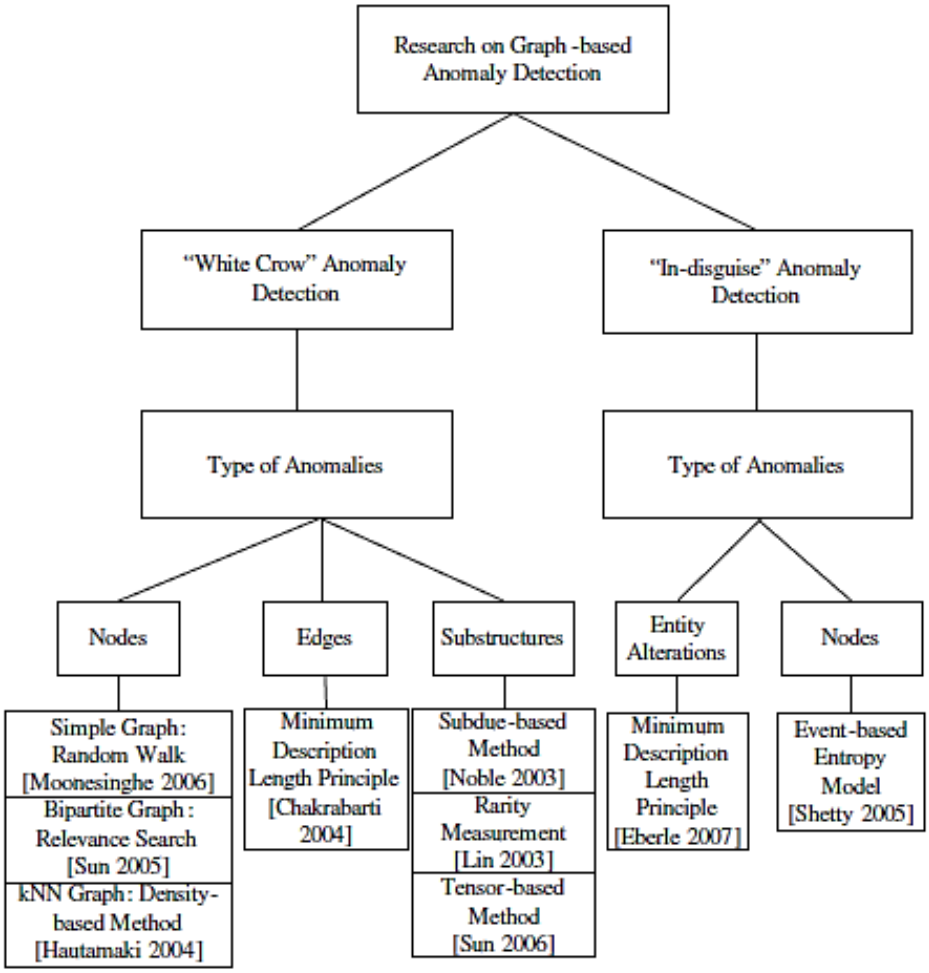


Figure 2.2: A summary of the various research directions in graph-based anomaly detection. (Source: Chen, Hendrix & Samatowa 2011, p. 2)

The recent growth of the internet of connected things has led to surprisingly large data generation, transportation, and storage. The generated data from the large-scale network of several connected 'things' or devices have produced huge data that is now popularly being referred to as 'Big data'. Big data has been very useful for several purposes such as weather forecast, anomaly detection, and so on (Kai, Singtel & Balachandran, 2020; Bendre, Thool & Thool, 2015). To ensure the integrity of big data that is generated by the community of connected things, the community itself must be properly observed. One of the best ways to effectively manage and monitor large-scale networks is the use of anomaly detection.

Chandola, Banerjee & Kumar (2007) provided a survey on available anomaly detection techniques. The provided survey gave a more structured and comprehensive overview of various anomaly detection research by grouping existing techniques into different categories. The underlying approach that was adopted by each technique served as the basis for the grouping. The survey identified key assumptions of each of the techniques that were used to differentiate between normal and anomalous behavior. The goal of the survey was to provide a better understanding of the different directions on anomaly detection research. It also discussed how developed techniques could be used in other domains that are different from the ones where they were created.

Such developed techniques in recent times include the use of artificial intelligence in addressing the problems of anomalies in large datasets. Anomalies in a large data set can be detected using different AI approaches and methods, one of the efficient approaches being used in recent times is deep learning.

2.3 Anomaly Detection with Deep Learning

Defining deep learning is challenging because it has been changing form gradually in the past decade. It could be defined as "neural networks that have a large number of parameters and layers in one of the following fundamental network architectures, unsupervised pre-trained networks, convolutional neural networks, recurrent neural networks and recursive neural networks" (Patterson and Gibson, 2017).

Deep learning autoencoders have been used for anomaly detection in recent studies. Fan, Zhang & Li (2020) used dual autoencoder for anomaly detection on attributed networks. Deep autoencoders have been used to detect anomalous network traffics emanating from compromised

IoT devices (Meidan *et al.*, 2018). KitNet algorithm, an ensemble of autoencoders which is based on a neural network was used to develop a plug-and-play network intrusion detection system that can collectively differentiate between normal and abnormal patterns (Mirsky, Doitshman, Elovic & Shabtai, 2018). Recently, deep neural networks have gained wide-spread attention, because it outperforms alternative machine learning methods such as kernel machines in numerous important applications. Both feedforward (acyclic) neural networks (FNNs) and recurrent (cyclic) neural networks (RNNs) have been very popular in deep neural networks (Schmidhuber, 2015).

A study on Time series anomaly detection was carried out by Shipmon, Gurevitch, Piselli, & Edwards, (2017). The study used an RNN model to detect and predict anomalies in time series. Apart from RNN, DNN and LSTM were used in the anomaly detection and all of the models have the same performance on the dataset that was used. RNN was more effective in handling false positives encountered in the study.

Wang & Paschalidis (2017) conducted a study on a two-stage approach for detecting the presence of botnet and identifying nodes that are compromised and controlled by Botnet. Anomalies were detected by observing large deviations of an empirical distribution. The study used a flow-based approach to estimate the histogram of quantized flow and a graph-based approach to estimate the degree distribution of node interaction graphs. The study was very effective in detecting anomalies based on large deviations results, but the drawback is in the inability of the approach to detect small deviations that could also be anomalous.

Several other deep learning-related anomaly detection methods were presented by Chalapathy & Chawla (2019). The study was based on a survey that presented a structured and comprehensive overview of research methods in deep learning-based anomaly detection. It also assessed the effectiveness of the adoption of the methods for anomaly detection across various application domains. Recurrent neural network shares the same family with FNNs, RNNs' major difference from FNN is that they can send information over time-steps (Patterson and Gibson, 2017). According to Schmidhuber (2015), RNNs are regarded as the deepest of all neural networks among others, they are more powerful than FNNs in terms of computation and can create and process memories of arbitrary sequences of input patterns in principle.

A confusion matrix is used to evaluate the performance of RNN and other deep learning models using accuracy, recall, and precision. A 2x2 confusion matrix is enough to evaluate anomaly detection models using true positive (TP), True Negative (TN), False Positive (FP), and

False Negative (FN) (Wu, Wei & Feng, 2020). Deep learning anomaly detection models that have been developed and used successfully include Isolation Forest, Support Vector Machine (SVM), Convolutional Neural Network (CNN), Variants of Auto-Encoder, Deep Belief Networks (DBN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) (Chalapathy & Chawla, 2019). A study on "time series anomaly detection" shows RNN to be more effective in handling false positives (Shipmon, Gurevitch, Piselli, & Edwards, 2017).

One of the most used deep learning approaches in the detection of anomalies is the RNN. A type of RNN known as long short-term memory (LSTM) networks was used by Malhotra, Vig, Gautam & Agarwal (2015) for anomaly detection in time series. The study trained a network on non-anomalous data, the trained network was used as a predictor over several time steps. Multivariate Gaussian distribution was used in the study to assess the existence of anomalous behavior.

2.4 Prediction of Anomaly and its connections

The last part of this study will focus on the discovery of anomalous nodes' relationships and forecasting of potential anomalous connections. Anomaly prediction involves the use of existing data to forecast the possibilities of anomalies on a network. Data sources today are heterogeneous and characterized by different types of entities and relations that could be leveraged for dataset enrichment (Araujo, 2017).

Tan, Gu & Wang (2010) presented a novel adaptive runtime anomaly prediction system. The system was called 'ALERT'. The system aimed to raise anomaly alerts in advance of occurrence to provide for just-in-time anomaly prevention. To achieve this, a novel context-aware anomaly prediction scheme was proposed, the ALERT system was meant to improve prediction of accuracy in dynamic hosting infrastructures.

Araujo (2017), proposed the modeling of heterogeneous graphs as Coupled Tensors to predict the evolution of some interaction in a network. The ability to study interactions between edges and vertices in networks will provide strong leverage for identifying and predicting anomalies in such networks. Deep learning will make such observation easier in large-scale networks. Forecasting the tensors jointly was expected to generate better predictions than when the tensors are considered independently. TensorCast method was proposed for this study and it achieved over 20% higher precision in top-1000 queries. It also doubled the precision when finding

new relations than comparable alternatives. The method was tested in datasets with over 300M interactions and it scaled well with the input size at (E + N log N). The main advantage of TensorCast is its ability to be simultaneously contextual and time-aware.

LSTM network for anomaly detection models behaviors that are dependent on time or sequence. "The output of a neural network layer at a time (**t**) to the input of the same network layer at a time (**t**)+1. It is more efficient than typical RNN because it solves the vanishing gradient problem that exists in typical RNN" (Alla & Adari, 2019). LSTM belongs to the gated recurrent unit (GRU) class of RNN; its components are forget gate (F_t), input gate (I_t), output gate (O_t), and memory cell. A detailed LSTM Network is as shown in Figure 2.3.

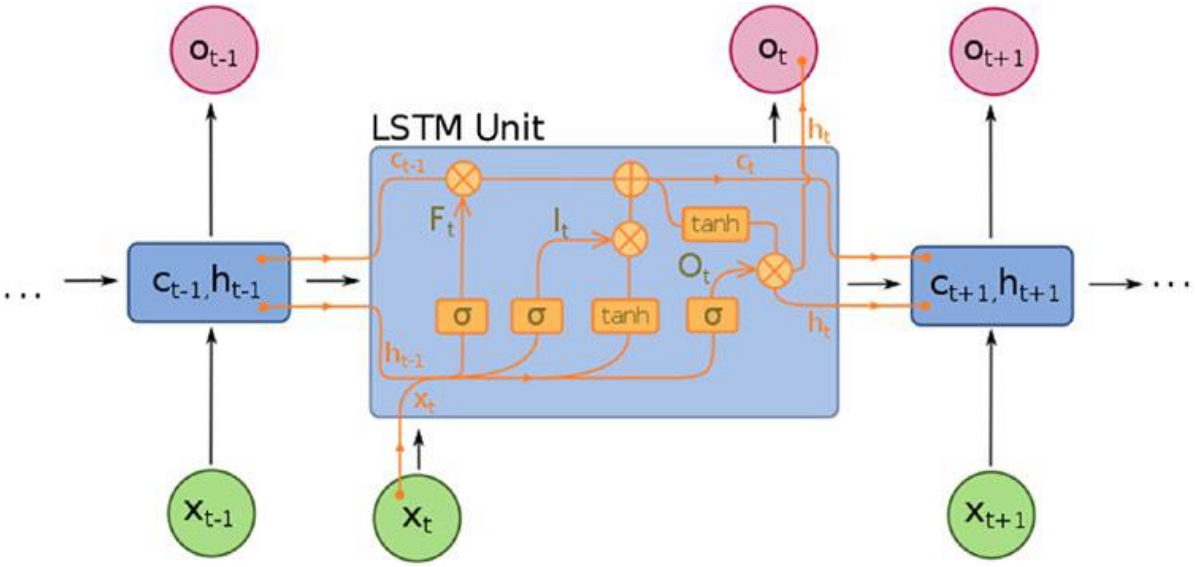


Figure 2.3: A detailed LSTM Network.
(Source: commons.wikimedia.org)

This study will implement LSTM in developing the CDoA model because of its strong efficiency in capturing the long-term dependencies across a large number of instants of time. The first part of the network is the forget gate, it decides the amount of information from a prior stage to be remembered or forgotten. The input gate is responsible for the decision of the amount of information to be passed to the current stage by using activation functions known as sigmoid and tanh. The output gate decides the amount of information the hidden state of a particular stage can retain and pass to the next stage. To strengthen defensive measures against threat actors, the CDoA

model requires additional efforts such as combining the LSTM with community detection approach in this study.

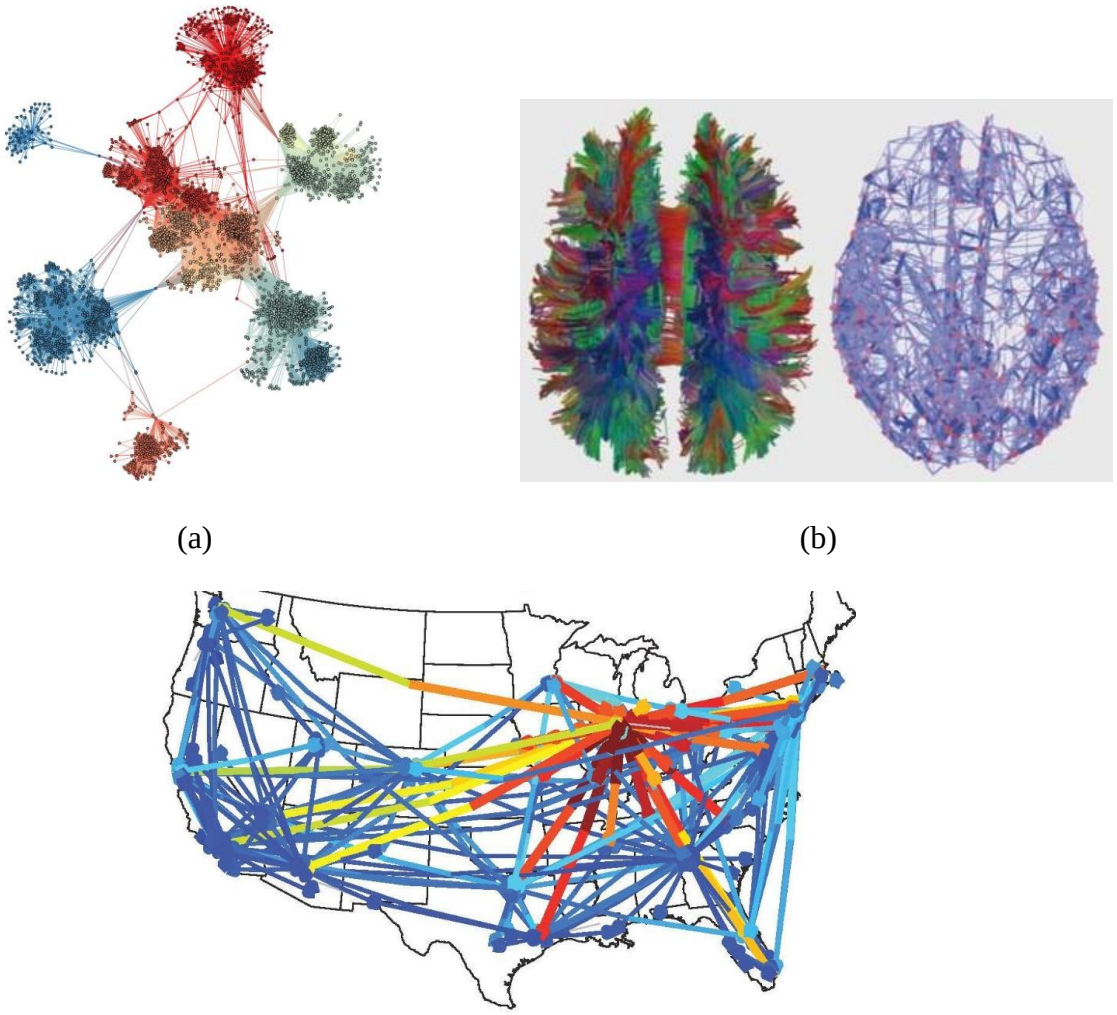
2.5 Community Detection

Community detection has gathered increasing attention lately because of the significance of its application to different domains of human endeavors. Community detection aims to partition networks edges and nodes into sets of clusters, this is to make related nodes within the same cluster more densely connected to themselves than to those in other clusters (Sun, He, Huang, Sun, Li, Wang, He, Sun, and Jia, 2020). Its application has been observed to have a significant impact in computer science, biology, sociology, physics, and other science and social science disciplines (Chakraborty, Dalmia, Mukherjee, & Ganguly, 2017).

Community detection is regarded as an ill-defined concept according to Fortunato (2010) because the nature of the communities is not known in advance. Barabasi (2016) defined community detection as subgraphs that are connected densely in a network and further classified communities as either strong or weak.

Community detection is based on the use of graphs in solving real-world problems. Graphs have been used over several years to model relationships between entities; such entities can be represented with real or abstract objects as shown in Figure 2.4a, b & c. Graphs have been used in biology, geography, and computing to model the relationship between neuron and their synapses, roads and their links, and compute nodes and their networks respectively (Fortunato, 2010).

The aim of using community detection for anomalies is to discover patterns and relationships of anomalies in a given network. Consider a network graph G(V, E), where V represents a set of vertices or nodes in the graph and E represents a set of edges or links between the nodes in the network. Every edge in the set of edges of the network can be represented by $e_{i,j} \in [0, 1]$ where i and j are the endpoints of that edge, forming the $|V| \times |V|$-sized matrix *E*. Graph can also be represented by a weighted graph G = (V,W) where matrix W replaces matrix E. Rather than having 0-1 values, cells w_{ij} are continuous variables. The weight of the edge is the magnitude of communication or relationship between the two endpoints of the edge. An exponentially or hyperbolically large number of possible sub-communities can be found in a given network. These sub-communities can be called subgraphs of any network represented by a graph solution.



(a) (b)

(c)

Figure 2.4: The use of graphs for modeling real-life scenarios.
(a) Social network (b) Neuron and synapses relationship (c) Roads and their links.
(Sources: Creusefond *et al.*, 2017; Petersen, 2015)

The simplest measurable statistics from a network are the node and edge count. The node count |N| is the number of nodes and the edge count |E| represents the number of edges. For this research, edge count refers to |W|, the total weight of the edges

In consideration of the network modeling, modularity function is used in a graph to determine the number of edges that are internal or external with respect to the original graph and to evaluate the difference in the number of such edges if null graph (random graph with identical

degree distribution as the original graph). For the given graph, the null graph was expected to be the most appropriate quality function that outputs the maximum modularity with the graph.

Modularity according to Newman (2006) is therefore defined as:

$$Q_{ud} = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{d(i)d(j)}{2m} \right] \delta_{\omega_i, \omega_j}, \qquad \text{(Eq. 2.1)}$$

Where A_{ij} is the adjacency matrix, d_i and d_j are the degree of nodes i and j, ω is the community of node, m is the number of edges on the graph, $\delta_{\omega i,\omega j}$ is the Kronecker delta function, the function returns 1 if $\omega i = \omega j$, or if i and j belong to the same community, and 0 if otherwise. The value of modularity lies between -1 and 1. A modularity value that is closer to 1 indicates a strong community structure while it shows a weak community structure if the values get closer to -1.

According to Araujo (2017), other metrics for measuring communities' quality are:

(a) Normal Cut or Conductance: it gives the best communities to the ones that are densely linked and connected with few edges to the rest of the network (Shi and Malik 2000).

(b) Partitioning: this method involves the process of splitting the graph into two groups of predefined sizes and repeated application to find groups of similar sizes. It is related to the label propagation method (Gregory, 2009). This method is not considered good for community detection because they require the definition of the number of groups and their sizes beforehand (Fortunato, 2010).

(c) Random-Walk: This method groups nodes of the network by their score when doing a random walk with restart (RWR) (Tong, Faloutsos, & Pan, 2008). This method is similar to PageRank (Page, Brin, Motwani & Winograd, 1999).

(d) Spectra: This method uses eigenvalues to partition a graph (Gkantsidis, Mihail, & Zegura, 2003). It has been well extended over time, its extension includes spectral clustering methods in the presence of node-attributes (Günnemann, Färber, Boden, & Seidl, 2014).

(e) Generative Models: These "models start by representing the network as a group of communities, then rely on inference methods to learn the most appropriate parameters to fit the model to the network" (Araujo, 2017, p. 29). An example of this is block modeling (Wasserman & Faust, 1994).

(f) Information Theory: Information-theoretic modules were proposed by Rosvall and Bergstrom (2007). The proposed modules try to maximize mutual information. Some approaches rely on this module at their core.

2.6 Community Structure

Community structure was defined as "the division of network nodes into groups within which the network connections are dense, but between which they are sparser" (Newman & Girvan, 2004, p.1). The efficiency of identified community structures in a given network can be measured with modularity metric (Q) as expressed in Equation 1. It is to be noted that networks in the real world contain more than a single subnetwork. Modularity in a random graph is different from that of a non-random graph. For random graph, Q = 0, but for non-random, Q lies between 0.3 and 0.7.

The formation of communities was based on the structural or functional similarities among the vertices in the network (Newman, 2004). Research on community structures in networks has a very wide and rich history (Newman & Girvan, 2004). Proper understanding of how communities are formed with nodes in a network will give a broad and deeper view of the network structure formation through the interaction of the nodes with identical nature (Mahoney, Dasgupta, Lang, & Leskovec, 2009; Faltings, Leyton-Brown, & Ipeirotis, 2012; Abbe & Sandon, 2015; Benson, Gleich, & Leskovec, 2016).

Chakraborty *et al.* (2017) illustrated a toy example to illustrate community structure as shown in Figure 2.5. The example was used to categorize communities in the real-world networks into different types as follows:

(a) Nonoverlapping or disjoint, (e.g. a lecturer teaching classes in a school from Mondays to Thursday and consulting for industries on Saturday and Sunday)

(b) Overlapping (e.g. a staff functioning in different committees in the same department)

(c) Hierarchical (e.g. team members in an organization being supervised by team leaders who are in turn supervised by departmental leaders and so on)

(d) Local (e.g. a faculty in a college having uneven interactions between certain members of staff within a department in a University)

This research is considering large-scale networks which contain millions or billions of nodes, which can be regarded as complex networks. The community structure of such networks can be narrowed down to overlapping and non-overlapping.

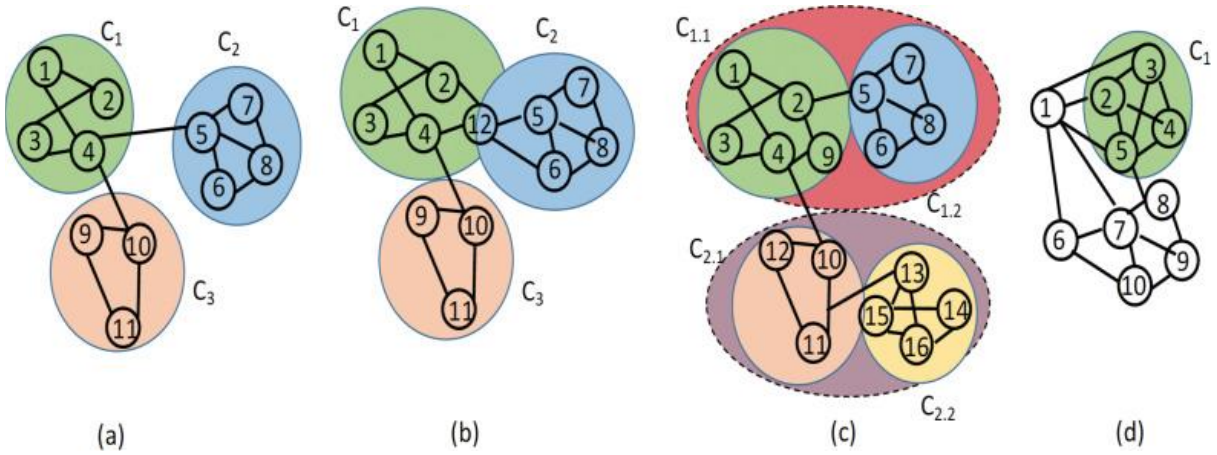


Figure 2.5: Illustration of different types of communities with toy examples.
(a) non-overlapping (b) overlapping (c) hierarchical and (d) local.
(Source: Chakraborty et.al, 2017)

The structure is considered overlapping when nodes in the given network can exhibit the characteristics of more than one community at a time while it is considered non-overlapping when nodes in the network belong to a single community per time.

2.7 Overlapping Community Detection

According to Orgaz, Salcedo-Sanz, and Camacho (2018), overlapping communities exist in real-world networks. Multi-Objective Genetic Algorithm (MOGA-OCD) can be used to detect overlapping communities. The algorithm used measures that are related to the network connectivity for the detection of overlapping communities. It used a phenotype-type edge information encoding and a new fitness function that focused on optimizing two classical objectives in a community detection problem. Although this approach generated a good result when used, the drawback is that modularity was not considered in both the internal and external metrics that were used in the approach. Using extended modularity density as a metric will enhance the possibilities of effective analyses of community structure.

Finding overlapping community structures is also important in realistically analyzing large networks such as the social network. A link clustering-based memetic algorithm for overlapping community detection can be used to optimize the modularity density function (Li and Liu, 2018). The algorithm can detect groups of links that are densely connected on the weighted line graph

that is modeling the network. The algorithm then maps the link communities to node communities using a novel genotype representation. Though the algorithm has a very good state-of-the-art performance, it requires high execution time.

According to Zhou, Liu, Wang & Li (2017), a density-based link clustering algorithm for overlapping community detection in networks is a good approach to solving problems related to excessive overlap. It can improve the accuracy of detecting overlapping communities in networks. The only setback of the algorithm is that it cannot handle weighted and directed networks.

The high cost of computation is associated with providing the optimum solution to community detection in large datasets that are generated by social networks. One novel approach that was proposed to address the problem of detecting overlapping communities in a large dataset is the use of a parallel community forest model and sequential Nash equilibrium for large datasets implemented in parallel with spark (Sarswat & Guddeti, 2018). Markov chain clustering algorithm could also be used to detect overlapping communities in both real and artificial networks (Deng, Ma & Li, 2018).

This project plans to set up an overlapping community detection experiment by using modularity in measuring the quality of a community structure and changes in the community structure in relation to network traffic anomalies. This will be carried out with a sampled large-scale network dataset. It is expected that the use of an efficient community detection algorithm in the development of CDoA will overcome some of the drawbacks of the existing overlapping community detection algorithms.

2.8 Community Detection Techniques and Algorithms

Several community detection techniques and algorithms have been devised and used in different spheres of network science. For example, Nonnegative matrix factorization (NMF), is one of the emerging standard frameworks (Ye, Li, Lin, Chen, & Zheng, 2018). It has been employed widely for overlapping community detection, its operation is based on the factorization of the adjacency matrix into low-rank factor matrices to obtain node's soft community membership (Wang, Wang, Zhu & Ding, 2011). The drawback of NMF is that it requires a very difficult task of post-processing the real-valued factor matrix by a manual threshold specification.

A "discrete overlapping community detection pseudo supervision" approach was proposed by Ye, Chen, Zheng, Li & Yu (2019). The project used Discrete Nonnegative Matrix Factorization

(DNMF). The operation of this framework is by seeking a discrete (binary) community membership matrix directly. DNMF does not need post-processing, to assign explicit community memberships to nodes. Another strength of DNMF is that it is robust. The robustness is enhanced by its ability to "incorporate a pseudo supervision module to exploit the discriminative information in an unsupervised manner." After a thorough evaluation of DNMF using both synthetic and real-world networks, DNMF was reported to have "the ability to outperform state-of-the-art baseline approaches".

Dynamic complex networks are known for community structures that change over time or frequently. One of the recent dynamic community detection algorithms that were introduced to capture such dynamics of network community structure is "a detailed analysis of the dynamic community detection algorithms," carried out by Singh, Haraty, Debnath & Choudhury (2020). The research tested dynamic algorithms such as quick community adaptation (QCA), BatchInc, GreMod, and learning-based targeted revision (LBTR) on small, medium, and large real-world network datasets. The research determined that some of the algorithms were best suited in terms of performance on certain networks than on the others. The comparative analysis will guide anyone who needs to choose the best dynamic community detection algorithms for various sizes of networks in the real world.

Deshmukh (2018) summarizes some of the previously used community detection algorithms in this section. Louvain algorithm was formulated on heuristic technique constructed with modularity optimization and is useful for the discovery of high modularity clusters in large-scale networks; it completely unrolls hierarchical community structure of the network (Blondel, Guillaume, Lambiotte & Lefebvre, 2008).

Community detection employs the use of graph theory to solve complex problems. Graph partitioning technique was developed by Fortunato (2010), which divides "vertices into groups of a predetermined size such that edges lying between the groups are minimized." The algorithms for this technique are unsuitable for detecting communities because they cannot reveal information about the structure of the community.

Partitional clustering technique essentially involves "identifying different clusters in a network and minimizing the loss function based on the distances between the points and/or identified clusters" (Fortunato, 2010). This technique is utilized by k-means, minimum-k clustering, k-medoids, and other classical algorithms. Its drawback is its requirement to specify

the number of clusters as inputs; in a real-world network, this may not be possible. A hierarchical clustering algorithm was developed to overcome this drawback. The algorithm has two popular classes known as Agglomerative and Divisive algorithms (Fortunato, 2010).

The Fast algorithm was developed as an agglomerative hierarchical clustering method (Newman 2004). Newman-Girvan's modularity metric was observed by Chakraborty *et al.* (2017, p. 31) as "the most popular and widely accepted metric in the literature of community analysis". It lays the foundation for many other metrics that are being used in community detection. Clauset-Newman-Moore algorithm tried to overcome the time-consuming limitation of the Newman Fast algorithm (Newman, 2004) by focusing on maintaining and updating the matrix of modularity value instead of tracking the adjacency matrix and calculating modularity value every time. The algorithm achieved a better running time as proposed (Clauset, Newman & Moore, 2004).

The Walktrap algorithm uses a hierarchical clustering approach and has an improved run time complexity (Pons & Latapy, 2006). Infomap algorithm's formulation was based on analysis of information flow in a given network. As a method it uses, random walks on the given network to unroll the community structure of the network. Communities in the given network are identified using an optimal compression of the network structure (Rosvall & Bergstrom, 2007). Label propagation algorithm: In this algorithm, all the nodes are assigned unique labels that indicate the community that each of the nodes belongs. Nodes determine their community based on their neighbors' community labels (Raghavan, Albert & Kumara, 2007).

Community detection and graph clustering methods were classified into five broad classes by Papadopoulos, Kompatsiaris, Vakali, and Spyridonos (2011). The classes are:

(a) Cohesive subgraph

(b) Vertex clustering comprises of spectral clustering (Donetti & Munoz, 2004; Von Luxburg 2006); (Wasserman & Faust, 1994); Walktrap (Pons & Latapy 2006).

(c) Community quality optimization

(d) Divisive is based on the following works, seminal algorithm (Girvan & Newman 2002)

(e) Model-based class spin model (Reichardt and Bornholdt, 2006), and statistical inference (Hastings, 2006).

Important methods of traditional clustering include partitional clustering, neural network clustering, and multidimensional scaling (MDS); a respective example of each is k-means

clustering, self-organizing maps, and singular value decomposition (SVD), and principal component analysis (PCA) (Gan, Ma & Wu, 2007).

Kernighan-Lin algorithm was proposed in 1970 as a heuristic procedure for partitioning electronic circuits into boards. The focus of the algorithm was to optimize Q, to achieve this, it uses the subset swap method. It begins by partitioning a graph into two predefined sizes, it then iteratively swaps the subsets that contain equal numbers of vertices between the two partitioned groups. It is best used as a supplement to high-quality partitions that are obtained by the use of other methods (Porter, Onnela & Mucha, 2009).

Community detection algorithms can also be classified into global and local algorithms. Global assumes that the whole network structure is known and available, it defines communities with respect to the whole graph. Local algorithms assume no previous knowledge of the network, it starts from examining "some given seed nodes and expand them" to the network. Identified local communities "can be aggregated to uncover the global community structure of the network". Local algorithms can result in many redundant communities and become costly in terms of computation if the algorithm starts naively from each of the nodes in the given network (Moradi, 2014).

2.9 Implementation Plan

An agglomerative class of community detection algorithms will be used to build a CDoA model in this study. The agglomerative hierarchical algorithm recursively merges nodes with high similarity to form communities (Nugraha, Perdana, Santoso, Zeniarja, Luthfiarta & Pertiwi, 2018; Babichev, Taif & Lytvynenko, 2016). Using this together with the LSTM anomaly detection model will significantly help in solving high-dimensional and complex security challenges. PageRank algorithm is an example of an agglomerative algorithm that will be used in this study. PageRank can be used to solve large network graph problems (Page, Brin, Motwani & Winograd, 1999). A modified page rank algorithm has been used successfully in the area of information retrieval and sequencing of pages on the Web (Usha & Nagadeepa, 2018; Sen, Chaudhary & Choudhury, 2017)

The problem of community detection of the anomalies will be implemented according to the Minimum Description Length (MDL) principle. The principle will use an approach that will group labeled nodes that are connected by edges with the main aim of minimizing the total description length of the labeled large network (consisting of millions/billions of nodes). To reduce the problem size and speed up the iterations, this part will be achieved with the implementation of

a variation of the PageRank algorithm on PySpark because it is naturally parallelizable and will scale easily to a massive dataset.

In providing answers to the research questions of this study, this research identified the sub-communities of anomalies in the large-scale network and formulate 'individual modularity values' for each of the subgraphs. This will help in incorporating all the possible subnetworks of the given network in the composite modularity metric evaluation process. For example, in a university campus scenario where several computers are being compromised due to various cyber-attacks, the cyber-attacks can be due to worm infestation, botnet, distributed denial of service (DDoS), or phishing. There is no single cybersecurity solution that can resolve all the cyber threats at once. The ability to detect the available communities of threat actors in such a network will significantly help the cybersecurity specialist to handle the related attacks in groups, and also to provide appropriate solutions to each one of the detected communities of threats.

A community structure with high modularity value among the threat actors will help the specialists to identify and rank the cyber-attacks in the order of severity, danger, complexity, and complication. The approach of community detection with modularity density metrics can also be used in a real-time network where network data changes with high velocity; time-series data apply to such problems.

Clustering of available anomalies in a large-scale network into related communities will make it easier to evaluate the strength and severity of such anomalies in given large-scale networks. The formation of a community in this setting can be established when there is frequent interaction among individual anomalies within a group than to the anomalies outside the group (Aditya, Dhuli, Sashrika, Shivani & Jayanth, 2020; Bedi & Sharma, 2016, p. 116).

2.10 Conclusion

The literature review for this research provides a justification basis to address the research questions. The components of the community detection of anomalies (CDoA) model were explained with relevant pieces of literature, various techniques and algorithms were discussed and a strong basis for evaluation criterion was systematically mentioned. The review explored the importance of using modularity metrics to measure the result of this study. It gave a clearer picture of community detection of anomalies as specified in the research questions.

Understanding the pattern of relationship among compromised nodes in a botnet-infested network will serve as a good example of how a community detection approach can be used to classify anomalies into various clusters for further investigation. This study will use the CDoA model to identify anomalies in the network traffic dataset and study the significance of available anomalies and changes in the underlying community structure of the network. Analysis of how the variables of this study have been successfully used in other studies was presented in this chapter. The result of this study will assist cybersecurity specialists in providing scalable solutions to cyber-attacks in a large-scale network.

CHAPTER 3. METHODOLOGY

This chapter explains the methodology and the overall research framework that was adopted during this research. Approaches and procedures for implementing the techniques of CDoA were discussed. Lastly, the source of the dataset, variables used, and the environmental setup for the execution was described at the end of this chapter for replicability purpose.

3.1 Research Framework

In real-world large networks, communities change with time. Developing a model that can recognize minute changes and understand the cause and source of such changes will go a long way in helping to analyze the activities of anomalies and their effects on the community structure of large networks. Consistent observation of the pattern of network traffics and its impact on the community structure will expose anomalies in the community. This kind of observation can be achieved by using deep learning techniques that have been tested in other domains of big data analytics. This study utilized the combination of exploratory and constructive research in answering the research questions. Essentially, this study tried to address the following research questions:

(i) Given a large-scale network with millions or billions of nodes, what level of accuracy of anomaly detection can be achieved with the CDoA model using deep learning long short-term memory (LSTM)?

(ii) Can we identify strong communities of anomalies in a large-scale network using the CDoA model with the Modularity metric as a measure?

In other words, the study investigates if the use of CDoA - a combination of deep learning technique for anomaly detection and community detection, can help identify anomalies in large-scale networks, and group the anomalies into communities based on their behaviors or pattern of communication. The quality metric that I used to evaluate the community structure is modularity. The change in modularity values of the sub-communities was used to detect changes in underlying community structures of the network.

3.2 Research methodology and experimental setup

The set of procedures that I followed in this research comprises data collection, data preprocessing, data training using deep learning algorithm, model generation with LSTM, and anomaly identification and separation based on TTL property of the network traffic data set. Community detection with Louvain and PageRank algorithms were also implemented. The workflow is as shown in Figure 3.1 and the processes are highlighted as follows.

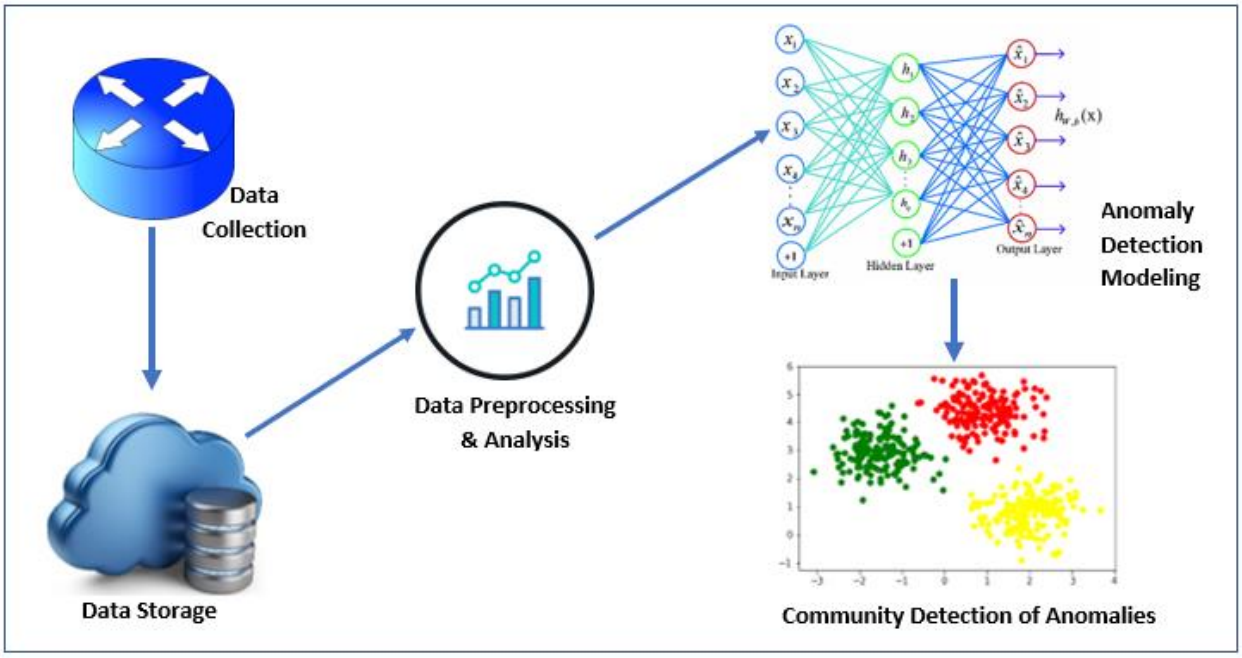


Figure 3.1: Community Detection of Anomalies Workflow

3.3 Location of Study

I carried out this study at the D.A.T.A laboratory of Purdue Polytechnic Institute, Purdue University, West Lafayette, Indiana. The HPC resources (Gilbreth) provided by the University were used for this study. Table 3.1 shows the detailed specification of the Purdue Gilbreth community cluster.

Table 3.1: Purdue Gilbreth Community Cluster Specification

Front-Ends	**Number of Nodes**	**Cores per Node**	**Memory per Node**	**GPUs per node**
With GPU	2	20	96 GB	1 P100
Sub-Cluster	**Number of Nodes**	**Cores per Node**	**Memory per Node**	**GPUs per node**
A	4	20	256 GB	2 P100
B	16	24	192 GB	2 P100
C	3	20	768 GB	4 V100
D	8	16	192 GB	2 P100
E	16	16	192 GB	2 V100
F	5	40	192 GB	2 V100

The Operating System that runs on each node is CentOS 7. For job and resource management, it uses "Moab Workload Manager 8 and TORQUE Resource Manager 5 as the portable batch system (PBS)". Each node has 100Gbps InfiniBand interconnects and at least 192GB of RAM (www.rcac.purdue.edu).

I carried out this research under the supervision of Dr. John Springer.

3.4 Data Collection

Anonymized passive network traffic traces dataset from the Center for Applied Internet Data Analysis (CAIDA) Equinix-nyc monitor was used in the implementation of this framework because of its robustness. The location of the capturing monitor was at an "Equinix data center in New York, New York and it was connected to an OC192 backbone link (9953 Mbps) of a Tier1 ISP between New York, NY and Sao Paulo, Brazil". The data for this research was based on direction label A (Sao Paulo to New York), captured on 01/17/2019 between 13:00 UTC and 14:00 UTC (www.caida.org). Trace statistics of the dataset used for this research are as shown in Table 3.2. The distribution is as shown in Figure 3.2.

Table 3.2: Trace statistics for the CAIDA dataset used in this research.
(Source: “https://www.caida.org/data/passive/trace_stats/”)

Duration	1hour and 02 minutes
First timestamp	1547729950.467105016
Last timestamp	1547733671.460902311
Total number of packets	2366419918
Total number of IPv4 packets	2339350262
Total number of native IPv6 packets	27069656
Total number of tunneled IPv6 packets	602

CAIDA’s passive and active measurement infrastructures provide visibility into the behavior of the Internet globally. Collected data are curated, archived, and shared. CAIDA’s anonymized the data by using CryptoPan prefix-preserving anonymization and store the data in pcap format. The size of the trace is about 640GBytes.

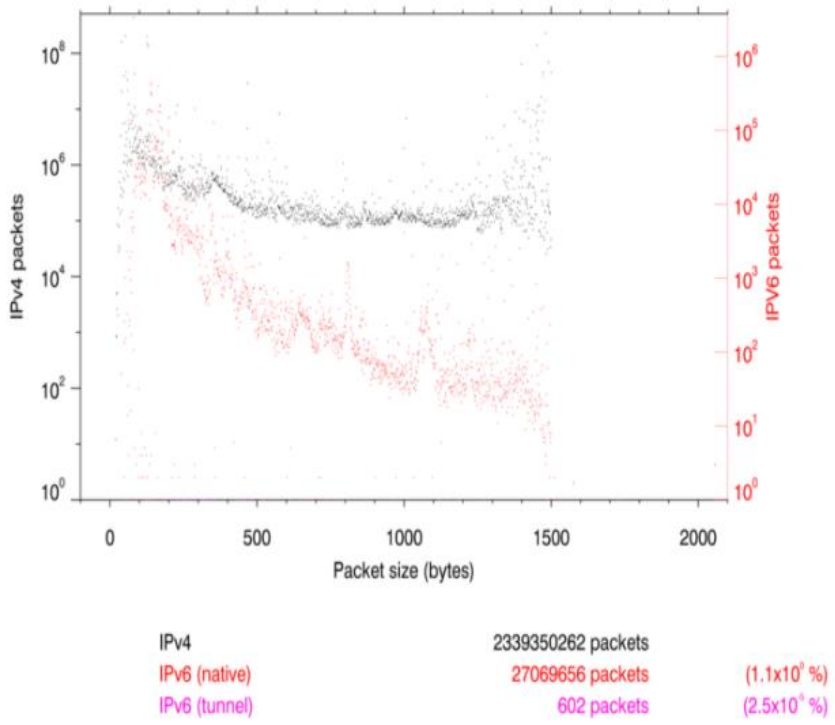


Figure 3.2: The distribution function of packet size for equinix-nyc.dirA.20190117-130000.UTC.
(Source: “https://www.caida.org/data/passive/trace_stats/nyc-A/2019/equinix-nyc.dirA.20190117-130000.UTC.df.xml”)

3.5 Data Storage

Data storage for this work was provided by Purdue D.A.T.A. Laboratory. Multiple data storage facilities were used for different levels of data wrangling and processing. The storage used includes:

i. Purdue Fortress HPSS Archive: A large system with a long-term, multi-tiered file caching and storage facility. It utilizes both robotic tape drives and an online disk. It uses an IBM T3584 robotic tape library with over 10PB capacity. This work enjoyed a limitless quota on Fortress.

ii. Purdue Research Computing – Data Depot: A reliable, fast, high-capacity, and secure data storage service. It was purposefully designed, configured, and operated for the Purdue researchers' needs. It is usable in any field of research and is shareable with collaborators on-campus and off-campus. It provides 100GB space free of charge and could also be purchased in increments of 1 TB.

iii. Purdue Scratch Parallel Filesystem: Scratch storage consists of several redundant and high-availability disk spaces filesystem. Scratch filesystem for Gilbreth was used. It consists of 2.3PB of redundant, high-availability disk space. It has a quota of 200TB and 2,000,000 files.

3.6 Data Preprocessing and Analysis

In this phase, I carried out several activities to fit the data into formats that are most appropriate for each level of analysis. Some activities that were carried out and tools used are described in the following section.

3.6.1 Data Wrangling

This part of the research took a whole lot of time compared to other phases of this research. It requires a very careful approach because a mistake at this point will negatively affect the whole result of this research. The original network traffic trace from the CAIDA dataset were all in pcap format. The dataset with a total file size of 640GBytes was zipped in small chunks for easy download and transmission. First, I extracted all the data with Bash scripting into Fortress storage. The needed direction A dataset was then separated from the whole chunk of

the dataset. The data were further converted into CSV format to enable the easy manipulation of the dataset by different software tools. I cleaned the data by removing all the not available (NA) and null parameters from the dataset.

3.6.2 Variables for Anomaly Detection

For the first part of this study I used 23 parameters/variables for model building in the LSTM layer for anomaly detection, this include units, activation, recurrent_initializer, reccurent_activation, use_bias, bias_initializer, kernel_initializer, unit_forget_bias, kernel_regularizer, bias_regularizer, recurrent_regularizer, activity_regularizer, kernel_constraint, recurrent_constraint, bias_constraint, dropout, recurrent_dropout, implementation, return_sequences, go_backwards, return_state, , stateful and unroll. The model would be trained with time-steps, learning rate, batch size, threshold cutoff, epochs of the neural network, and hidden layer.

For the second part, I used some features of the network traffic traces as variables. These are outlined in Table 3.3.

Table 3.3: Network Traffic Variables to be used in Anomaly Detection

Features	**Description**
Source_IP	Packet Source IP Address (Node)
Destination_IP	Packet Destination IP Address (Node)
Packet_Length	Length of the captured packet
Time	Time duration of the captured frame
Time-To-Live (TTL)	Time-to-Live of each packet
Protocol	L3 Protocols: Internet Protocol, (UDP), andTCP

3.6.3 Variables for Community Detection

The variables that I used in this study for the community detection part are displayed in Table 3.4.

Table 3.4: Variables used in the study for community detection

Variable	Description
N_anom	The total number of unique anomaly nodes/vertices available in the dataset
L_anom	The total number of unique anomaly links/edges available in the dataset
N_link	The total number of links (edges) between the nodes in each community
N_nodes	The total number of unique nodes/vertices available in each community
$\boldsymbol{Q}$	The value of modularity for the community structure of the anomaly graph according to Newman (2006).

Explanatory/Independent variables are N_anom, L_anom, N_link, and N_nodes while the dependent variable is, $\boldsymbol{Q}$.

3.6.4 Software Packages, Libraries, and Tools

An anaconda 5.1.0 environment was loaded on the Purdue Gilbreth cluster for this research. The following software packages, libraries, and tools were installed in the environment. Python 3.8.3 was chosen to be used for this work because it is the most recent version that is available on the cluster, it is lightweight and easy to use and debug. Scikit-learn 0.23.2 (sklearn) which is a machine learning library was used with Python because it contains almost all the algorithm needed for this work and has an extensive background. Pandas 1.0.5 powerful library on Python was used to perform different operations such as filtering, bulk deletion, replacement, and addition. Matplotlib 3.2.2 was used for visualization such as graph creation. Numpy 1.18.5 was used for mathematical and logical operations. Seaborn 0.10.1 and Tensorflow 2.3.0 were also used in collaborations with many other packages such as Keras 2.4.3 in this environment.

3.7 Anomaly Detection Modeling

LSTM sequential model was implemented with Keras. This model has an LSTM layer and a dense (fully connected) layer. To get the final output between 0 and 1, I applied sigmoid and tanh activation functions to the dense layer.

Sigmoid activation function

$$f(z)=\sigma(z)=\frac{1}{1+e^{-z}} \quad \text{(Eq 3.1)}$$

tanh activation function

$$f(z)=\tanh(z) \quad \text{(Eq 3.2)}$$

For the loss function, I used the Adam optimizer and the mean squared error. Some parameters like threshold cutoff, time-steps number, epochs of the neural network, hidden layer, and batch size were varied to analyze different results of the model in this study. The study employed the distributed algorithm approach for the identification and analysis of anomalies in the network traffic associated with the time series in the dataset. The following procedures were used in this modeling:

i. To select the dataset, data was loaded from the dataFilePath as a CSV file using Pandas data frame.

ii. To describe the value column, describe( ) command was run on the dataset to understand the data more.

iii. To normalize the data, a seaborn KDE plot was used to plot the dataset. This reveals the minimum and maximum data points of the dataset; scaling was used for the normalization. Scaling = (x-Min) / (Max-Min).

iv. The anomaly detection model was formed with a sequential model with Keras. The formed model has the LSTM layer as the hidden layer, while its dense layer forms the output layer. The LSTM layer's output was used as the input of the dense layer. Sigmoid activation was applied to make the final output range between 0 and 1.

v. The model was trained for 100 epochs, the training set was used as the validation data.

vi. The loss and mean absolute error graphs were plotted during the training process.

vii. After training the model, data for testing was predicted, and the root means square error (RMSE) was computed.

viii. The predicted dataset and the test dataset were used to arrive at the value of the difference, this value was passed through vector norms.

ix. The difference values were sorted, and a cut-off value was used to select the threshold for the anomaly.

x. Any value outside the range of the threshold was considered an anomaly.

TTL was used as the pivotal variable to identify anomalies in the dataset (Patel, Srinivasan, Chang, Gupta & Kataria, 2020). The threshold picked after model training were in three different ranges, they are ($1 < TTL_A \leq 30$), ($64 < TTL_A \leq 98$), and ($128 < TTL_A \leq 255$) where TTL_A is abnormal TTL value. This agrees with previous research such as Scheitle, Gasser, Emmerich & Carle (2016); Yamada & Goto (2012). All the detected anomalies were extracted and aggregated in preparation for community detection.

3.8 Community Detection of Anomalies

As previously discussed, one of the main aims of this research is to investigate the relationship that may exist between anomalies in large-scale networks. It is believed that the ability to establish communication between anomalies may provide a good insight into possible threats to cybersecurity specialists. This will help them to narrow down their investigation to specific areas of concern, as a result, associated resources, cost, and time spent on solving such risks would not be a waste.

Community detection involves the use of graph-based solutions to address real-world network problems. This research followed used Louvain algorithm and PageRank for the community detection part of this study.

3.8.1 Community Detection with Louvain

Louvain algorithm is a greedy optimization method with complexity O(NlogN) that identifies disjoint communities in a network. It seeks to maximize the value of modularity (Q) for each community. The values of modularity range from -1 to 1. Modularity value closer to 1 signifies quality community.

The two stages of the Louvain algorithm were implemented. At the first stage, all the nodes were assigned a community of their own, then for each node ***i***, the gain in modularity is computed by moving the node to its neighbor ***j*** with the highest modularity gain, if there is no modularity gain the node remains in its initial community. This process was repeated for all the nodes in the

anomaly dataset. The second stage of the algorithm groups all the nodes in the same community to form a new single node. Intra-edges are collapsed into a single self-loop edge and the weight is the sum of the weight of all the intra-edges. Multiple inter-edges between two communities are collapsed into a single edge and the weight is the sum of the edges between them, a new network was formed when the second stage was completed. The algorithm then iteratively called the first stage again and the cycle is repeated until there was no more modularity gain.

3.8.2 Community Detection with PageRank

The principle used here is an approach that grouped labeled nodes that are connected by edges with the main aim of minimizing the total description length of the labeled large network. To reduce the problem size and speed up the iterations, this part was achieved with the implementation of a variation of the PageRank algorithm. PageRank is good because it is naturally parallelizable and will scale easily to a massive dataset. The following rules were followed in implementing PageRank:

3.9 Assessment Instrument

The quality of the results of anomaly detection with LSTM deep learning can be evaluated using different metrics. The metrics used for evaluation are determined by the structure of the data and the goal of the experiment. The quality of the LSTM model can be evaluated based on a confusion matrix using true positive (TP), true negative (TN), false positive (FP), and false-negative (FN) to estimate the accuracy and precision of the model.

Another method for calculating the accuracy of the deep learning LSTM model involves the use of mean absolute error (MAE) and root mean square error (RMSE). This method was used to determine the accuracy of the deep learning LSTM model that was used for anomaly detection in this research. The method was chosen because of the nature of the dataset; the dataset was not pre-classified. The presence of anomalies in the dataset was assumed before the experiment was carried out. The anomalies in the dataset were being sought out of the wild.

Modularity, a measure of goodness of partitioned network was used to evaluate the quality of the identified community of anomalies. Gephi and Tableau were used for data visualization of detected communities. An example of Gephi visualization of identified communities is shown in

Figure 3.3. Different communities of anomalies detected can be represented with distinct colors as shown in the diagram.

3.10 Conclusion

This chapter provided insight into the execution workflow of this study. It presented a detailed description of the variables that were considered for the study. The location of the study and the available resources for the study were also outlined in this chapter.

This study explains the efforts made on the implementation of AI to speed up the identification of anomalies in large network traffic. It described how AI can be used to analyze the existing relationships among identifiable communities of anomalies in a given network. Lastly, it described how AI can be used to gain insight into the pattern of anomalous connections in the network.

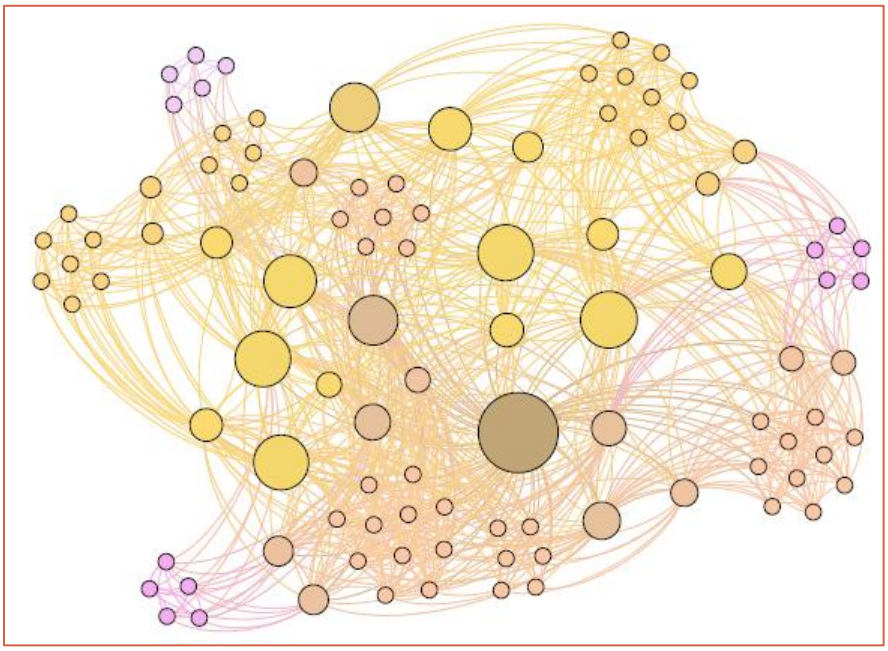


Figure 3.3: Gephi visualization of different communities.
(Source: Bolaji, 2018)

CHAPTER 4. RESULTS AND DISCUSSIONS

This chapter provides the result from various implementations I carried out in this study. As mentioned in previous chapters, the major aim of this research was to identify the presence of anomalies in large network traffic datasets and to study the relationship between the discovered anomalies. These were achieved by using deep learning LSTM and community detection algorithms. The results of this study are presented and discussed in this chapter.

4.1 Data Presentation and Analysis

As mentioned in chapter 3, an anonymized passive network traffic traces dataset from the CAIDA Equinix-nyc monitor was used in the implementation of this framework because of its robustness. The robustness of the CAIDA dataset has been helpful in many studies such as "cache snooping rare domains at large public domain name service (DNS) resolvers" by Randall, Liu, Akiwate, Padmanabhan, Voelker, Savage & Schulman, (2020); finding outbound addresses in traceroute (Marder, Luckie, Huffaker & Claffy, 2020). The data were zipped in about 65 pcap files. Each of the pcap files was first unzipped and later converted to CSV files for easy data manipulation and evaluation. The size of each of the pcap files is about 4.6GB on average. Each has an average of 29 million rows and 7 columns. The sample of the specific columns and their data are as shown in Table 4.1.

The source, destination, and info columns contain the most important parameters that were used in this study. The TTL values were extracted from the info column.

Table 4.1: Embedded information in each of the pcap files used in the study

	No.	Time	Source	Destination	Protocol	Length	Info
0	1	0.000000	26.120.99.176	175.84.134.113	TCP	1504	443 > 56199 [ACK] Seq=1 Ack=1 Win=32768 Len=...
1	2	0.000001	181.47.185.47	30.241.116.225	TCP	56	49860 > 443 [ACK] Seq=1 Ack=1 Win=40368 Len=...
2	3	0.000001	169.85.223.179	136.127.65.205	TCP	1484	20115 > 2928 [ACK] Seq=1 Ack=1 Win=256 Len=1440
3	4	0.000002	169.85.223.179	136.127.65.205	TCP	1484	20115 > 2928 [ACK] Seq=1441 Ack=1 Win=256 Le...
4	5	0.000003	42.27.128.236	180.211.115.195	UDP	1470	52557 > 1026 Len=1438[Packet size limited du...

4.2 Implementation Requirements

The whole study was implemented in an environment on the Purdue Gilbreth cluster. The first research question of this study was based on anomaly detection while the second research question focused on community detection of discovered anomalies. The following were carried out as research experiments:

i. Anomaly detection with LSTM Deep Learning

ii. Community Detection with Louvain and PageRank Algorithm

4.3 Anomaly detection with LSTM Deep Learning

LSTM is known to be very good with time-series datasets; the dataset in use has a timestamp that can easily be plotted for all the data points. LSTM has shown the highest performance for time-series classification in the studies of Xu, Zhao, Liu & Sun (2020), and Hashida & Tamura (2019). Since the dataset used in this research is time series as can be seen in the plots of TTL against date time as shown in Figure 4.1, 4.2, and 4.3. The plots were shown for the first 1,000 rows, the first 10,000 rows, and a big chunk of the whole dataset containing millions of rows. The plots show discrete changes in the values of TTL with time for each of the received packets of the dataset.

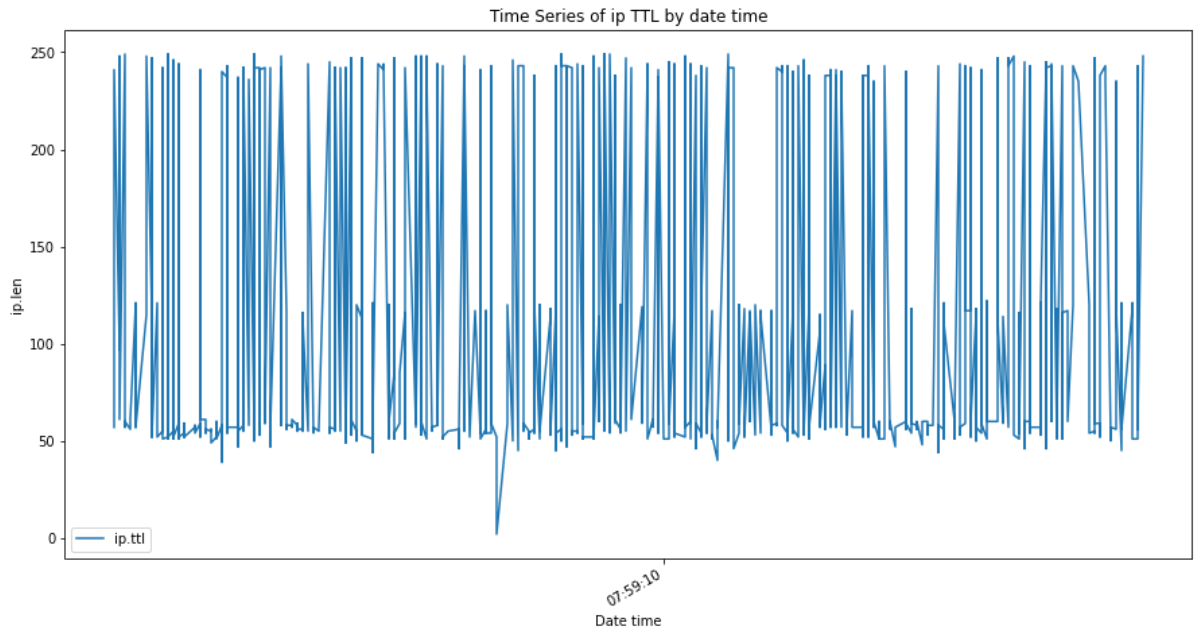


Figure 4.1: Plot of TTL against date time for the first 1000 rows

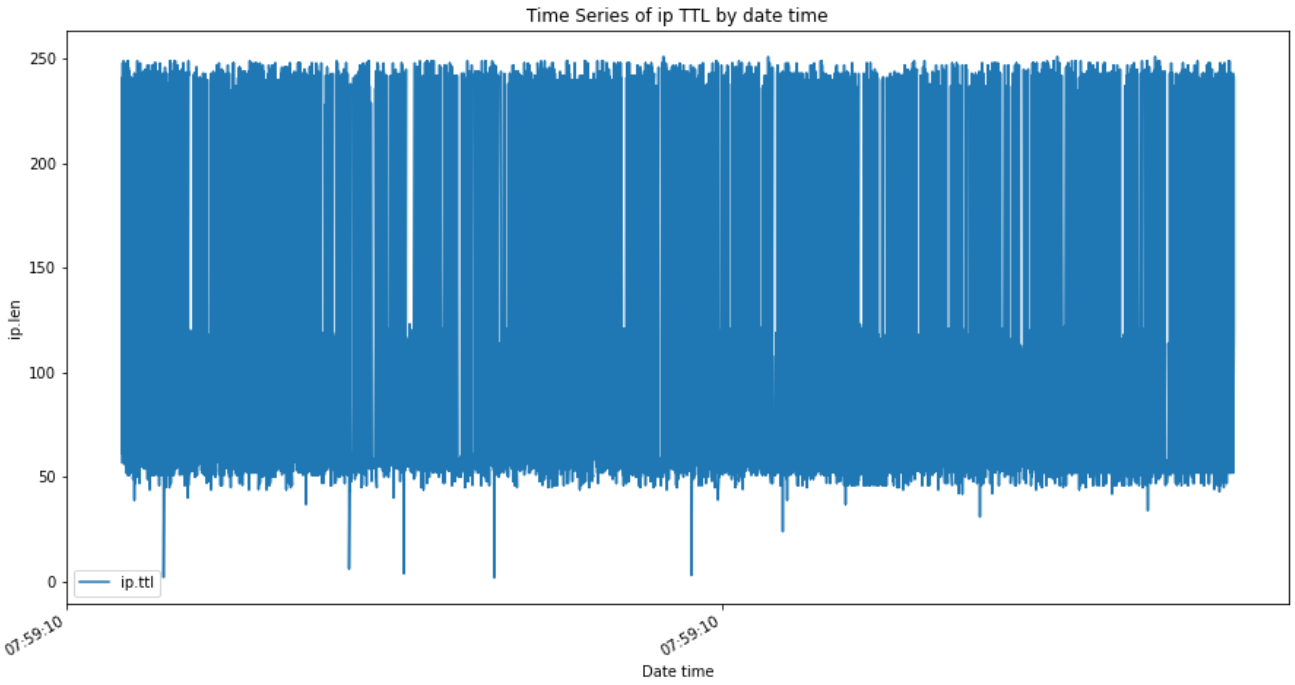


Figure 4.2: Plot of TTL against date time for the first 10000 rows

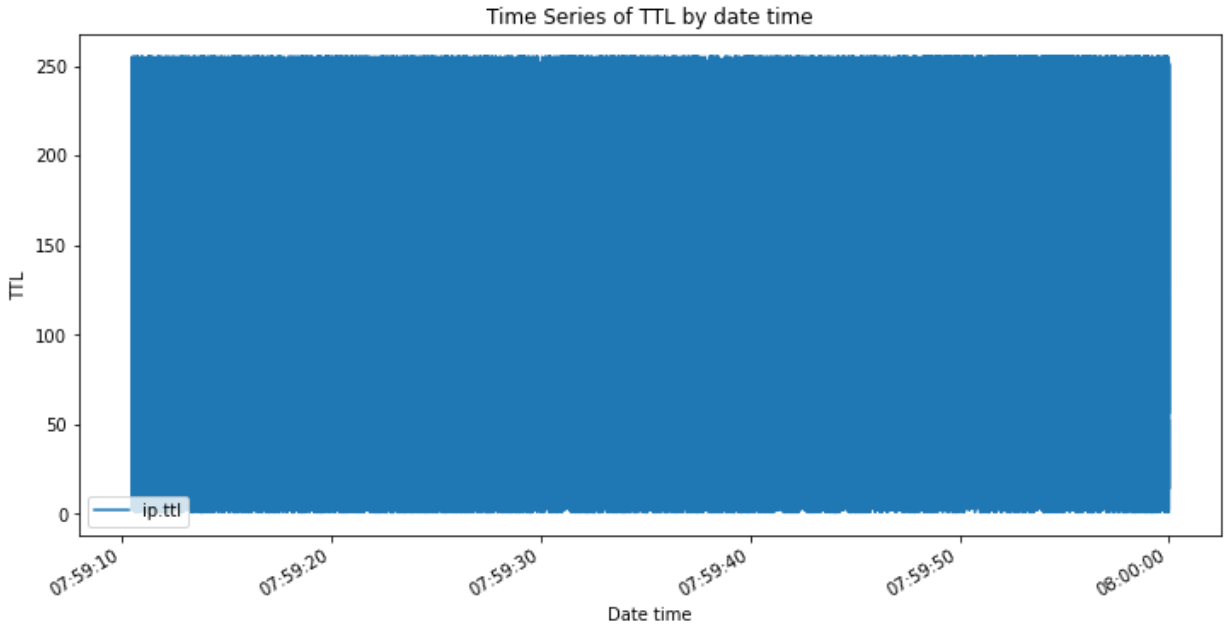


Figure 4.3: Plot of TTL against date time for the millions of rows

The anomaly detection process was implemented using Python as formulated by Alla & Adari (2019). Different packages such as Keras, sklearn, seaborn, matplotlib, sys, pandas, NumPy, and TensorFlow were imported into the environment. The data was loaded from the dataFilePath

(/depot/datalab/bolaji/Anoms/AnomaliesAll.csv) as a CSV using Pandas. The describe () command was used to look at the structure of the data for a better understanding of its packet length and TTL values. It yielded the value column of the data as shown in Table 4.2. Data understanding in cybersecurity is very essential because it can expose hidden trends of breaches.

The value column for both packet length and TTL values before scaling and after scaling were plotted with KDE as shown in Figures 4.4 and 4.5 respectively. The minimum value for the packet length is 28 while that of TTL is 2. The maximum values for both are 1,500 while that of TTL is 225. The formula used for calculating scaling is (x-Min) / (Max-Min) (Alla & Adari, 2019; Domingos & Hulten, 2001). The scaling process includes the derivation of upper and lower bound for the learning loss as a function. The function is for the number of examples that were used in each of the steps of the algorithm. The KDE plotted graph shows that the shape of the dataset remains the same before and after scaling. Big data scaling reduces the number of resources that are required for analysis in data analytics. Scaling also speeds up the data interpretation process, this is huge in security provisioning in cyber networks.

4.3.1 Calculating the MAE and RMSE of the model

The goal of the whole process is to find anomalous data points, that is, data points that are out of order among the dataset. In detecting the anomalies, an LSTM deep learning model that was built for this purpose was implemented.

Table 4.2: Description of the value column based on packet length and TTL values

	ip.len	**ip.ttl**
count	1.358340e+07	1.358340e+07
mean	6.491317e+02	9.145724e+01
std	6.350038e+02	4.470046e+01
min	2.800000e+01	2.000000e+00
25%	5.200000e+01	8.300000e+01
50%	2.190000e+02	8.700000e+01
75%	1.450000e+03	8.900000e+01

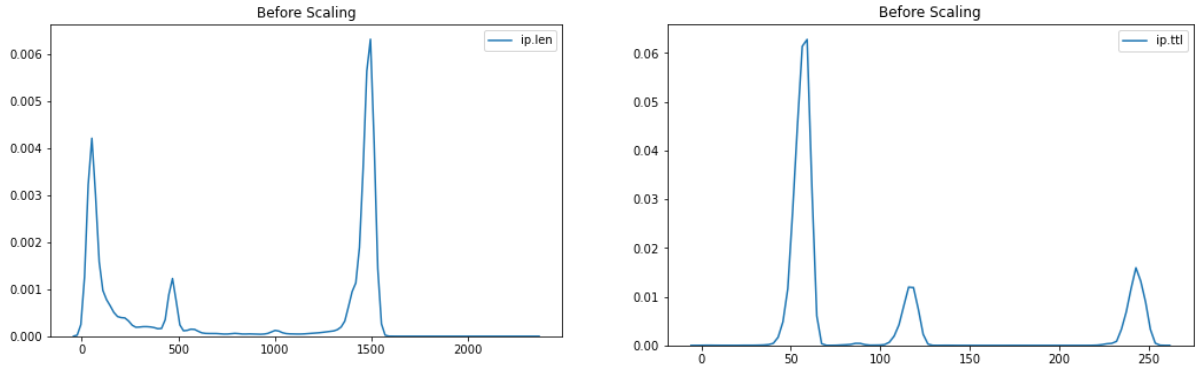


Figure 4.4: The plot of value column of packet length and TTL values before scaling

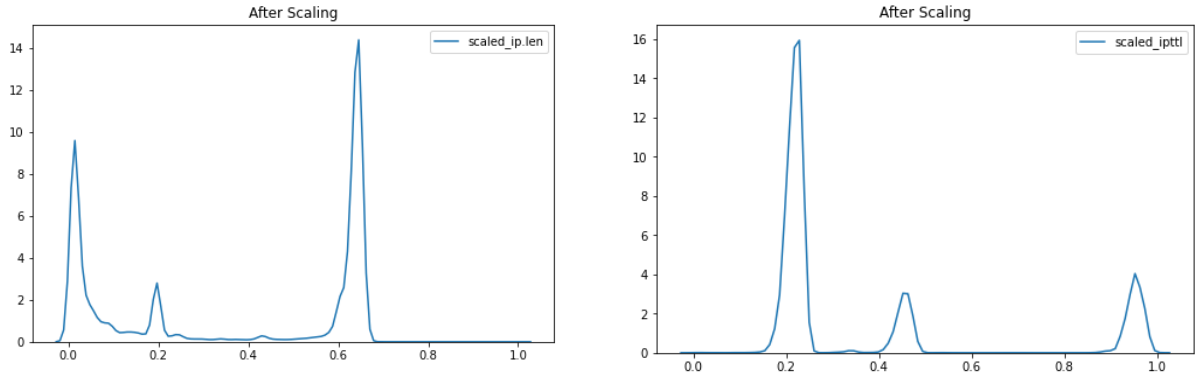


Figure 4.5: The plot of scaled value column of packet length and TTL values

The model was used to classify and separate anomalies from the dataset. A very important question to ask about the model is, “how accurate is the built model and can the model’s accuracy be measured?” This is where the use of mean absolute error and calculated loss serve as metrics for testing deep learning models. Root mean square error (RMSE) and Mean absolute error (MAE) are useful in measuring the accuracy of LSTM models as can be seen in the studies of Ali & Hassanein (2020), Wang, Guo & Chen (2019). An inaccurate model, for example, can lead to massive problems in data security. The integrity of data cannot be guaranteed, high-level confidentiality and privacy can also be jeopardized as a result of an inaccurate model. Spending a huge amount of money to purchase a security system that cannot be tested for accuracy poses a very high risk to cyber networks.

The mean absolute error of any test dataset is the same as the average of the absolute values of error of prediction on all data points of the test dataset. Mean absolute error was used to measure the accuracy of the anomaly detection model. The attempt was to predict a data point at a time (**t**)

based on the history of the existing data until the time (**t**-1). This helped to compare an expected value to an actual value; to determine if the data is within the expected range of values for time (**t**). The difference between predicted and actual values produced a sequence of errors as a distribution. The mean absolute error (MAE) is achieved by using the following approach:

Actual Value - Predicted Value = Prediction Error (Eq. 4.1)

The prediction error is recorded for each of the predictions after all errors have been converted to positive using the absolute value for each of the errors. That is,

Absolute Error = |Prediction Error| (Eq. 4.2)

To arrive at MAE, the mean for all the recorded absolute errors was calculated with the formula in equation 4.3.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \quad \text{(Eq. 4.3)}$$

The process was achieved by building a sequential model (anomaly detection model) using Keras. LSTM layer was used as the hidden layer of the model while a dense layer (connected layer) was used as the output layer. The time series dataset was fed into the LSTM layer, the layer learned the values of the dataset with respect to time. The output of the LSTM layer was used as the input for the dense layer, the dense layer transformed the input values into a fully connected one. Sigmoid activation was applied on the dense layer to get the final output between 0 and 1.

For loss function, **adam** optimizer and mean squared error were used. When output that is produced from the model is different from the input, the loss function penalizes the network that creates them. The loss metric was used to distinguish between the anomalies and the normal datapoint because anomalies do have higher reconstruction error as can be seen in Figure 4.6. Reconstruction error reflects the characteristic of anomalies (Chang, Du & Zhang, 2019). The reconstruction error plot of the dataset revealed that data points that are above the threshold (anomalies) had higher errors point compared to the normal points.

The anomaly detection model was trained until the best accuracy was achieved; this was at 100 epochs. The training dataset was used as the validation data. The loss during the training and validation process was plotted as shown in Figure 4.7. It could be deduced from the plot that there is a healthy correlation between training loss and validation loss. They are both reducing around a constant value. This serves as an indication that the model is well trained and that the model is good on both the hidden and training data.

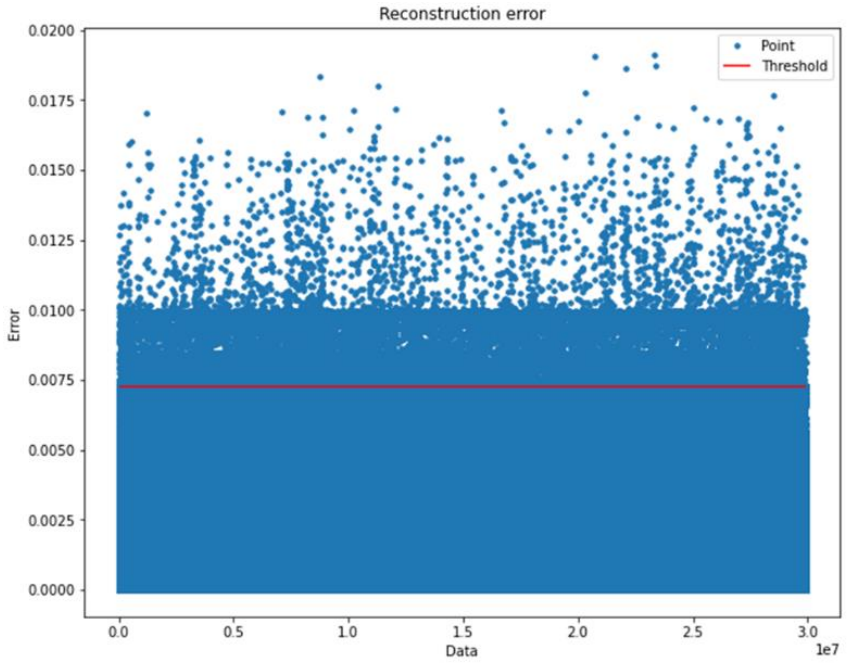


Figure 4.6: Reconstruction error plot

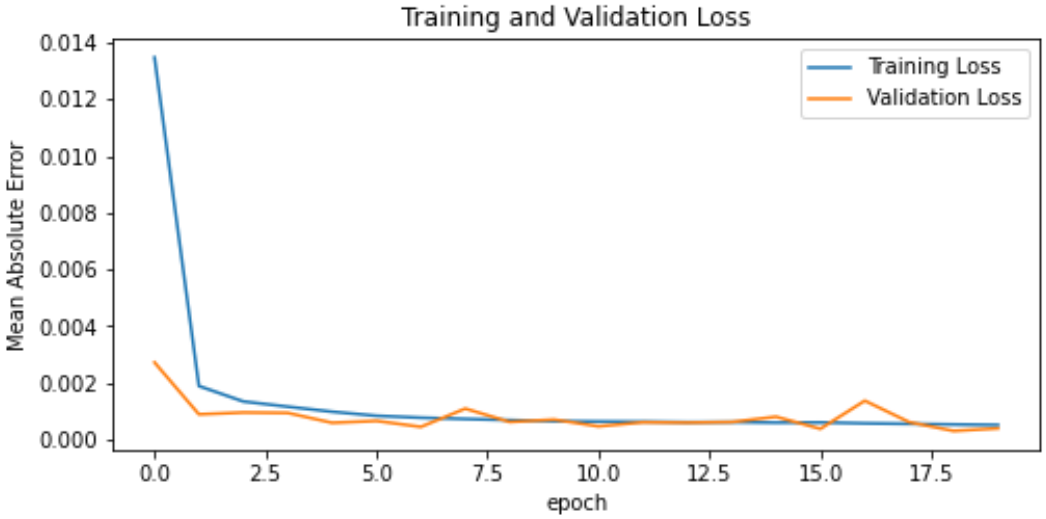


Figure 4.7: Training and validation loss graph

4.3.1 Anomaly Detection Model Validation

To validate this model, regression loss functions were used instead of the confusion matrix. The structure of the dataset is not compatible with using a confusion matrix because the anomalies in the dataset are not pre-labeled. Therefore, MAE and RMSE were computed to estimate the accuracy of the model. After the model was trained, a test dataset was used. The test dataset was split into subsequences of the same length according to the time steps, and this was used as the training dataset. Root mean square error (RMSE) was computed after this and it has an output of 0.001.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}} \qquad \text{(Eq. 4.4)}$$

The RMSE value was quite low, this agrees with the findings in the studies of Ibrahim & Hossain (2020), and Ali & Hassanein (2020). The low error loss from the training phase after 100 epochs confirmed the low RMSE at loss: 5.1812e-04, mean absolute error: 5.1813e-04, validation loss: 3.9858e-04, validation mean absolute error: 3.9858e-04. The low validation values achieved in the study are supported by the values gotten with the LSTM model by Roy, Roy, Gupta & Sharma (2020), Tandon, Tripathi, Saraswat & Dabas (2019), and Liu, Jiang & Wang (2020).

4.3.2 Calculating the cutoff value/threshold

To identify an anomaly in the dataset, the distribution of the calculated loss in the training dataset was used to determine a threshold value. As a regularization method in deep learning, it is usually a requirement to calculate the magnitude (length) of vectors. After sorting the diffs, a cut of value was used to pick the threshold. The threshold was set above the noise level to prevent the triggering of false positives. The chosen threshold was 0.0072. Any data point above this value was considered an anomaly. The plot of the dataset with respect to the chosen threshold is as shown in Figure 4.8.

The data points colored red are regarded as the anomalies while the ones in green color are the normal data points according to the chosen threshold. The anomalies were chosen based on TTL values of each of the packets as explained in chapter 3. This agreed with the position of Patel

et.al., (2020), Gasser et.al., (2016), and Yamada & Goto (2012). The position indicated that malicious IP packets can be identified by abnormal TTL values. Using LSTM deep learning methods for anomaly detection with TTL values shows that this is a worthy cause.

4.4 Conclusion for Research Question Number 1

The accuracy of the model performance, as evaluated by measurements of the two loss functions, MAE and RMSE is very high as can be proven by other studies such as Sunny, Maswood & Alharbi (2020); Ibrahim & Hossain (2020); and Ali & Hassanein (2020). The values of mean absolute error at 5.1813e-04 and that of root mean square error at 1.0000e-03 shows high accuracy in the difference between the actual values observed and the anomalies values predicted by the deep learning LSTM model.

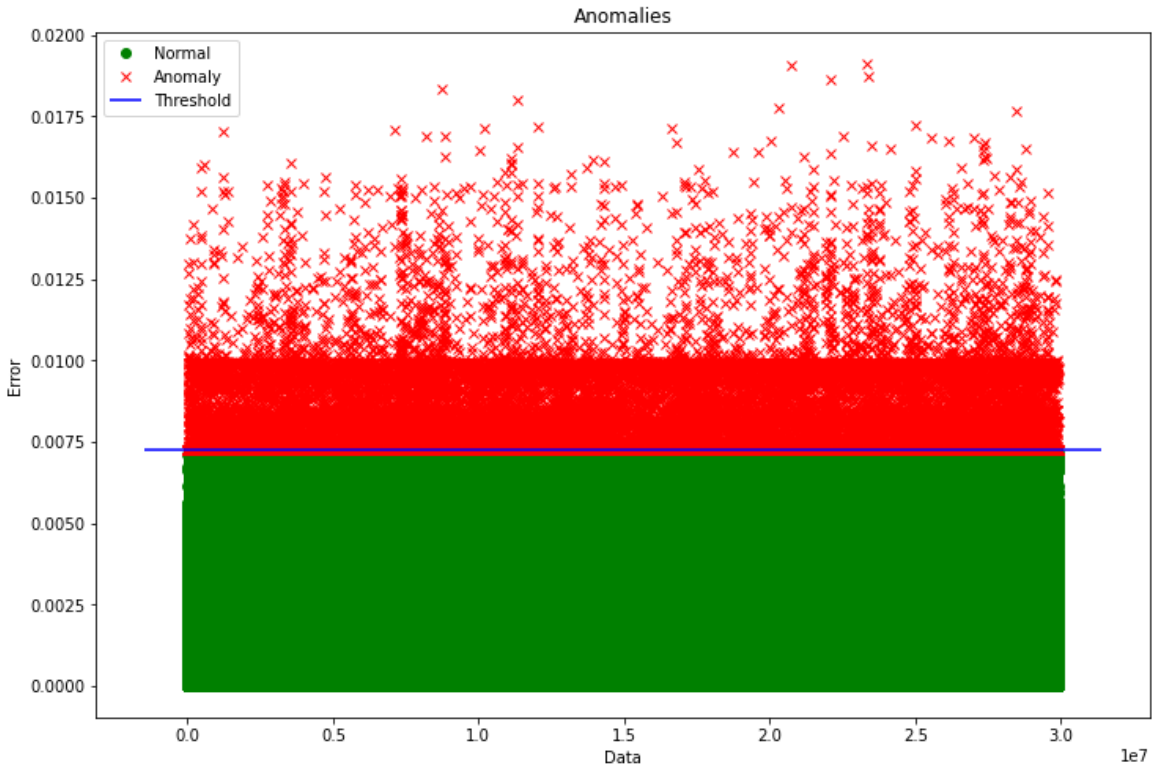


Figure 4.8: Plot of the data points with respect to the chosen threshold

This result shows that a high level of accuracy can be achieved with the CDoA model when deep learning long short-term memory (LSTM) is used on a large-scale network with millions or billions of nodes. The deep learning LSTM model will be significant and useful in determining

anomalous packets in a large-scale network using the Internet protocol Time-To-Live value as the basis of the determination. Separating anomalous packets in big data will significantly help cybersecurity specialists in narrowing down their search beacon in big data; as a result, investigation time is saved for more purposeful use, and the cost is reduced.

4.5 Community Detection with Louvain and PageRank Algorithm

After the anomalous packets were separated in the experiment as previously described. An attempt was made to discover communities among the identified anomalous packets. This was to investigate the existence of a relationship among the components of the identified anomalous packets. To achieve this, a new dataset was formed using the identified anomalies in the pcap files. The combined size of the anomaly dataset is 1.02GB. It contains few columns that may be used in community detection and visualization of the communities. The columns are ip.src (relabeled as **Source**), ip.dst (relabeled as **Target**), ip.len (relabeled as **PacketLen**), and ip.ttl (relabeled as **TTL**) as shown in Table 4.1

Table 4.3: Overview of newly formed Anomaly Dataset

PacketLen	Source	Target	TTL
40	55.36.90.123	171.223.205.255	225
314	52.30.36.94	131.96.28.128	225
40	55.36.90.123	171.223.205.255	225
40	55.36.90.123	171.223.205.255	225
40	55.36.90.123	171.223.205.255	225

4.5.1 Community Detection with Louvain Algorithm

Louvain algorithm optimizes modularity, that is, it seeks to maximize the value of modularity (Q) for each community. The values of modularity range from -1 to 1. Modularity value close to 1 signifies quality community. To achieve this, unique IP addresses were identified from both the source and target columns of the anomaly dataset. These IP addresses were used as nodes. The edge list was made using the rows that indicated communications between a source

and destination(s) or target(s) that were extracted from the dataset. There were **149,001** nodes and **1,048,576** edges.

All the nodes and edges were used to generate a graph with Python-NetworkX. The graph generated is as shown in Figure 4.9. It represented the connectivity between various nodes and edges. Gephi was also used to generate the anomaly graph with resolution setting at 1.0 and 2.0. The generated graph at resolutions 1.0 and 2.0 are as shown in Figure 4.10a and b respectively. Different colors were used to represent identified communities of anomalies in the graph.

It could be observed that all the graphs maintained the same shape. However, the size of the dataset makes it difficult to analyze the identified communities in the graph. Different colors were used to represent each of the identified communities. Further analysis was carried out with Gephi and Tableau to better visualize the graphs and the sub-communities that were formed in the graph.

About 1,920 sub-communities were identified in the graph. The measured modularity value of the graph is 0.91, this is considered very high. This agrees with various studies on modularity optimization such as Newman (2006) and Wang, Sun, Sun & Chen (2020), and Huang, Wang & Chao (2018).

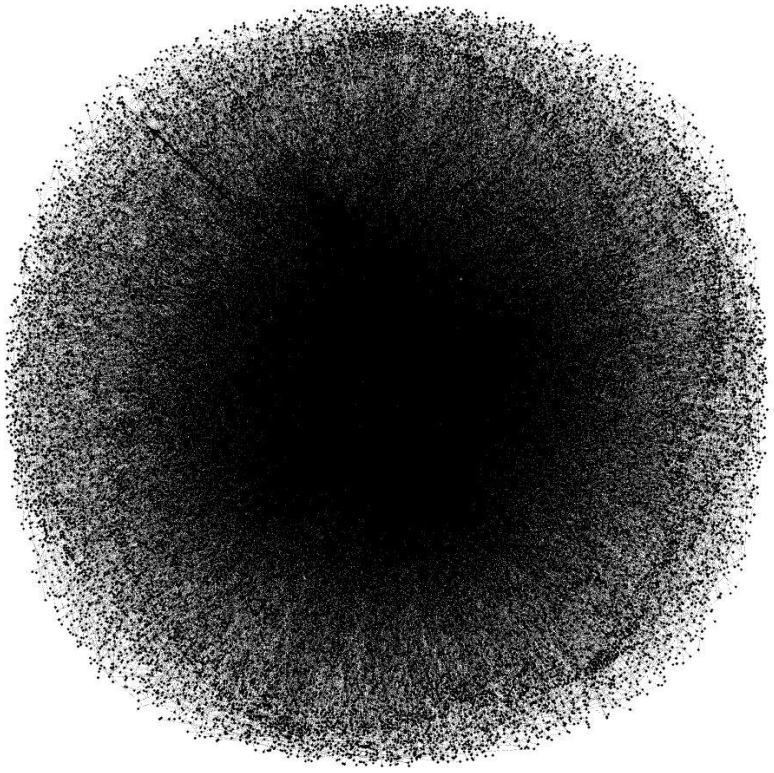
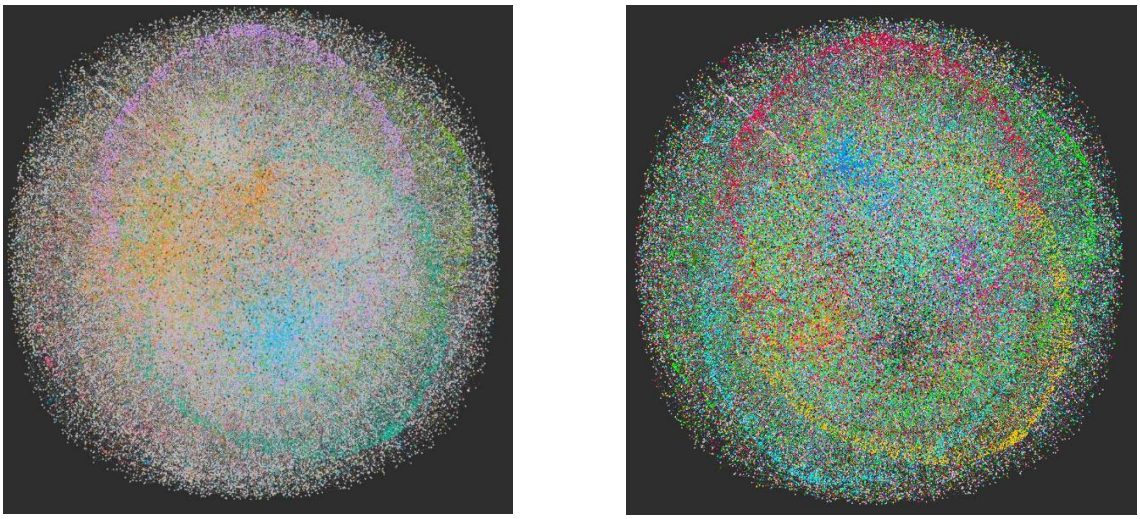


Figure 4.9: The Graph of the Anomaly dataset (Q = 0.914)



(a) (b)

Figure 4.10a and b: The generated graph at resolution (a)1.0 and (b) 2.0

Modularity maximization devises approximate and heuristic schemes because the search space is exponential with respect to the number of nodes (Papadopoulos *et al.*, 2011). As a result, the greedy solution of Louvain merges the majority of the identified sub-communities that are negligible based on the number of nodes that are involved. For example, the largest identified sub-community in the graph has 10.41% of all the nodes that made up the graph. It is as shown in Figure 4.11. The communities with similar features were merged to optimize the modularity value of the graph. Figure 4.12 represents this with a bubble chart (circle packing diagram). A bubble chart uses circles to represent categories, colors to represent differences, and the size of the circle to represent the proportion of quantities. The diagram shows that 39 clusters were generated as a result of the merge. The largest cluster contains all the sub-communities with 0.01% of all the nodes. Different colors represent different classes, while the sizes represent modularity class count in each cluster.

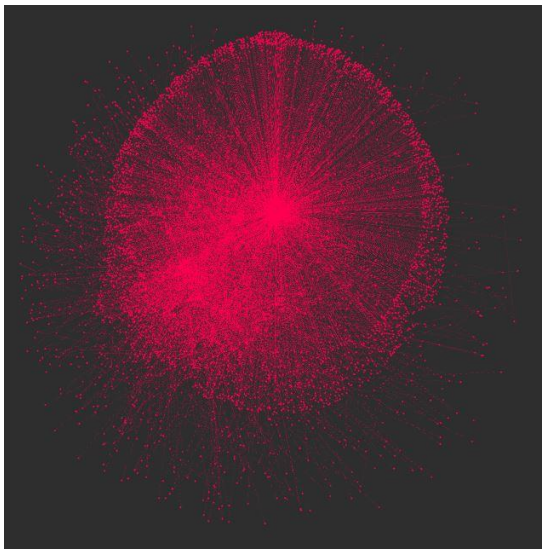


Figure 4.11: The largest sub-community in the graph with 10.41% (15, 600) of all the nodes

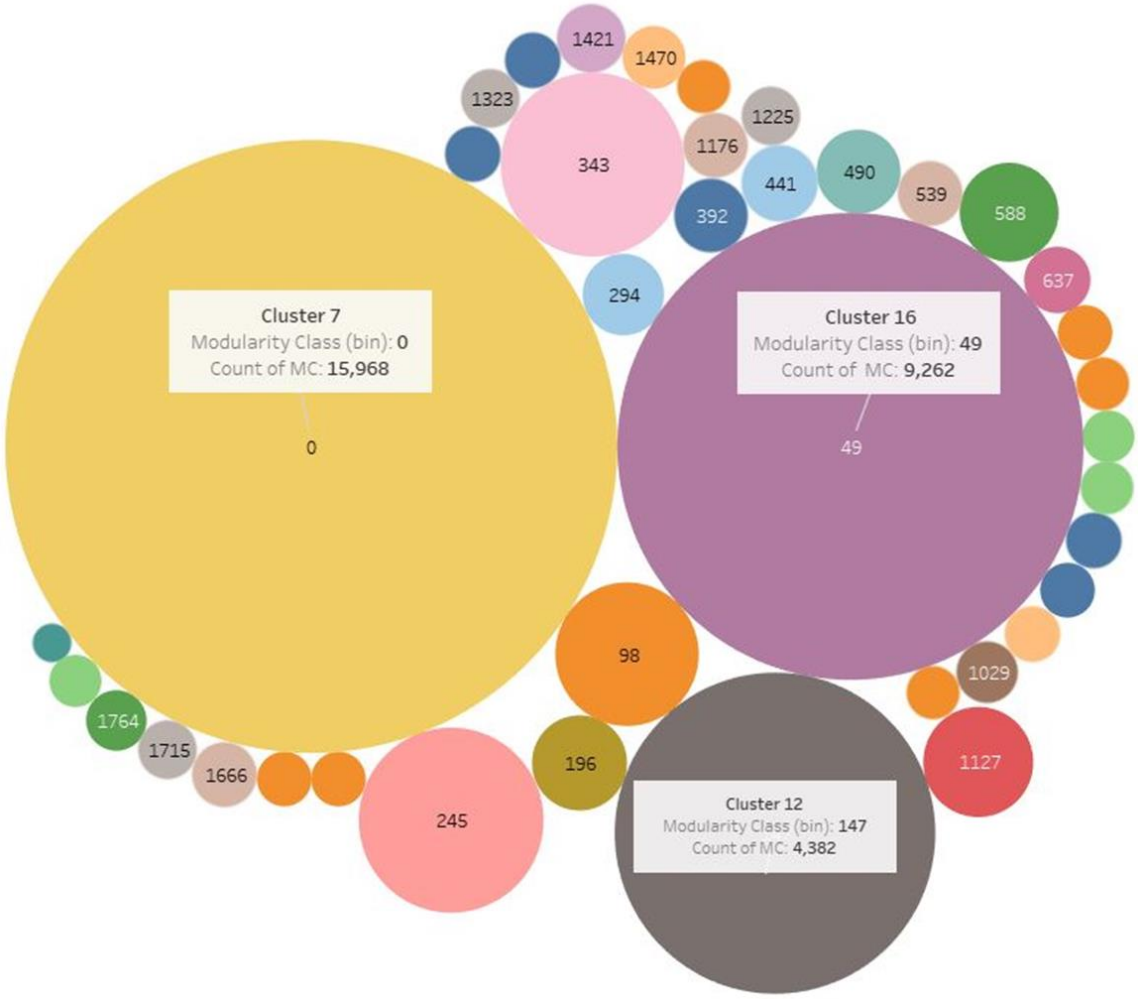


Figure 4.12: The sub-communities in the graph after merging the ones with related features.

4.5.2 Community Detection with PageRank Algorithm

The importance of the PageRank algorithm started with its use to count the number and quality of links to a page, and to generate a ranking score for the pages. PageRank was used in this research to generate a ranking score (r) for each of the nodes in the dataset. It specifically iterates over every node to check for its neighbors and out edges. As mentioned earlier Python NetworkX was used to create the graph. The probability for damping factor (D) was chosen to be 0.85. The graph was iterated over 95 times.

The abridged version of the PageRank score generated for each of the nodes is listed in Appendix A of this report. The value of the PageRank score is the probability between 0 and 1, just like the modularity value used in the previous section. Individual node PageRank's value was based on the number of nodes that are connected to it (Zhang, Xia, Xu, Yu, Wu, Yu & Wei (2020) and Page, Brin, Motwani & Winograd (1999).

The highest PageRank score observed is 0.03502 for a node with IP address 146.206.121.50 while the one with the lowest score was given as 0.00002 with an IP address of 175.84.136.42. The implication of these scores shows that the device with IP address 146.206.121.50 has the highest influence on the network; it is the device with the highest number of connectivity. The highest-ranking devices are the ones with the most connections as expected. The visualization of the PageRank is as shown in Figure 4.13 the list of the first 1,000 PageRank score of the 149,001 nodes is listed in Appendix A.

The dynamic nature of present-day network traffics due to the growing numbers of connected devices makes it difficult to study community structure and find hierarchical overlapping community structure in large-scale networks. The implementation and result of this section show that PageRank can be used to tackle the community detection problems in the large-scale network (Zhang et.al., 2020).

4.5.3 Community Detection Result Validation

Many other quantitative measures have been used in previous research for validating the output of the community detection algorithm (Li, 2016). A detailed study showed that using metadata as a test for community detection algorithm has a lot of shortcomings (Peel, Larremore & Clauset) but, modularity has been reported to be the most widely used accepted metric (Newman

& Girvan, 2004). The modularity achieved in this study is very high at 0.91, this validates the existence of strong communities in the graph of discovered anomalies. The validation result agrees with previous studies on community detection, such as validation of community robustness (Carissimo, 2016), Community detection: effective evaluation on large social networks (Lee &Cunningham, 2014), and Fagnan, Abnar, Rabbany & Zaiane (2018).

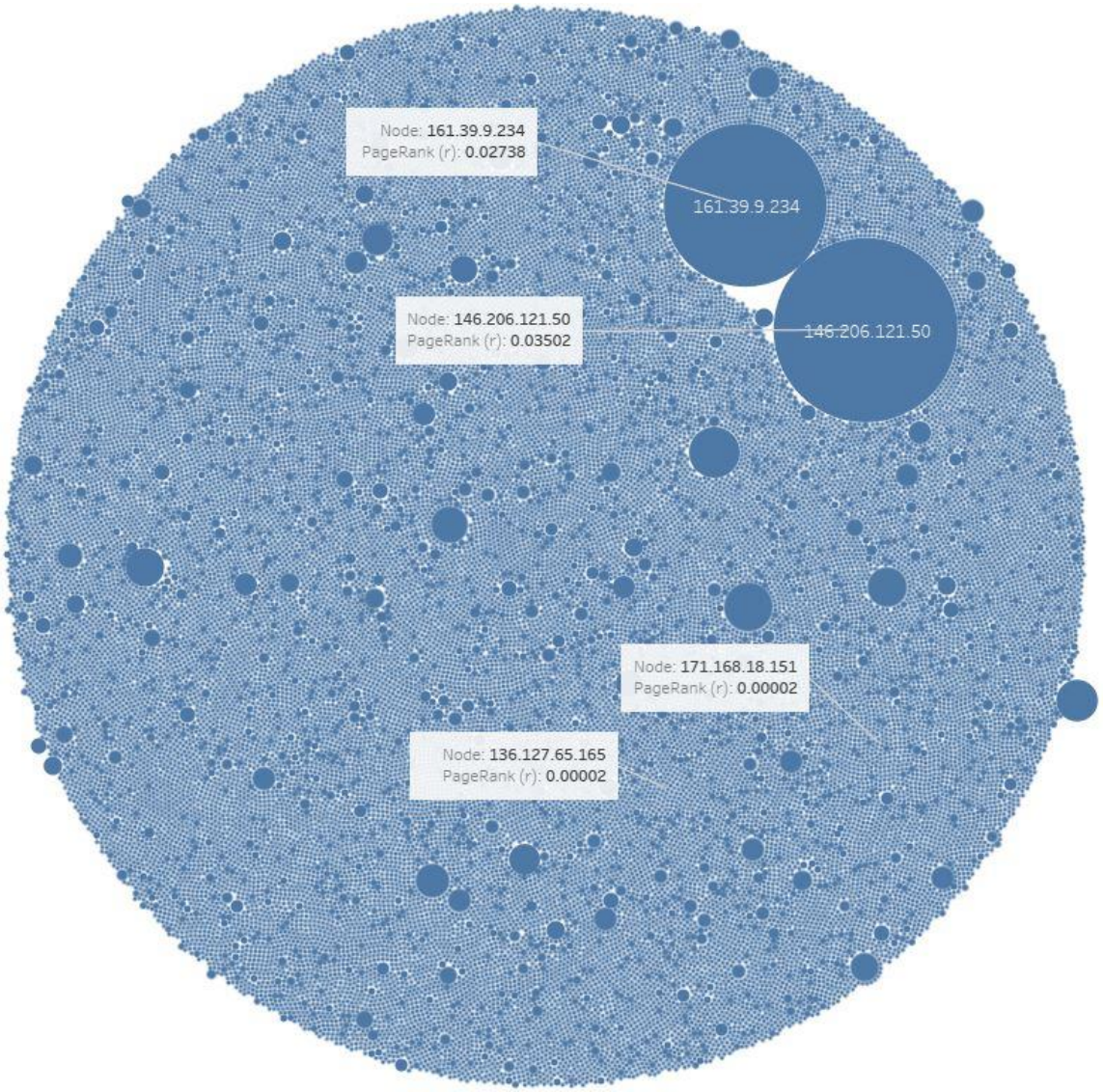


Figure 4.13: Visualization of the PageRank Score for the Graph of the Anomalies.

4.6 Conclusion for Research Question Number 2

As previously mentioned in chapter 2, it was stated in the literature that the Louvain algorithm is one of the fastest community detection algorithms. It can handle large networks while preserving the quality of the communities detected. The modularity measure was used as the quality function to assess the community detection implementation in this study. The result of the modularity value which is equal to 0.91 is very high. This is an indication that the graph of the anomaly dataset has a very strong community. The Modularity value shows that there is strong compartmentalization of the communities discovered in the dataset. This agrees with the position of previous studies by Newman (2004), Newman (2006), Newman & Girvan, 2003, and Wang, Sun, Sun & Chen (2020).

4.7 Conclusion

This chapter discussed the experimental results of various implementations that were carried out to answer the research questions of this study. The results showed high accuracy for the LSTM deep learning anomaly detection model that was used in this study. The communities discovered in the study have very high modularity value which indicates a strong network. The result of the implementation of the PageRank algorithm yielded good ranking scores for all the nodes in the graph.

CHAPTER 5. CONCLUSIONS, SUMMARIES AND RECOMMENDATIONS

This chapter provides the concluding thoughts and summary of the attempts made to fulfill the goal of this study. It discusses further the results of this research and provides answers to each of the research questions of this study. This chapter emphasizes the importance of CDoA to the field of cybersecurity by discussing its applicability to present-day networks. It explains the possible future direction of this research and ties the whole process together. This chapter discusses anomaly detection with deep learning, community detection of anomalies, contribution, and future research direction.

5.1 Anomaly Detection with Deep Learning in Large-scale Network

The original goal of this study was to use artificial intelligence to design, develop and implement a multilayered enterprise cybersecurity solution. However, after much studies, discussions with the research advisor, and data gathering, it was discovered that the goal was not research-driven. The initial goal was eventually redirected into formulating research questions and hypothesis which can be addressed within the specific timeframe on available resources.

Despite many fruitful academic kinds of research on anomaly detection in large networks, cyber-attacks have continued to plague most of the well know big technologies. The detection of anomalies in real-world networks has applications in various domains of human endeavor; the application includes, but is not limited to, credit card fraud detection, malware identification and classification, cancer detection from diagnostic reports, abnormal traffic detection, identification of fake media posts, and the like. Many ongoing and current researches are providing tools for analyzing labeled and unlabeled data; however, the challenges of finding anomalies and patterns in large-scale datasets still exists because of rapid changes in the threat landscape(Arisoy, Nasrabadi & Kayabol, 2021; Araujo, 2017 and Yao, Shu, Cheng & Stolfo, 2017).

As a result, we have a host of related challenges. How can we speedily identify sources of compromise or infections in large networks with millions or billions of nodes? Is it possible to predict the next employee that will fall for a phishing attack? What pattern of intrusion can be identified with artificial intelligence in a power distribution grid system? Can we stop the

fraudulent transfer of a large amount of money in and out of a country? How do we know if a machine learning algorithm is infected? This study was carried out to model large network traffic using appropriate matrix, parameters, and tensor representation to gain insights into these questions.

The use of the CAIDA dataset for the study provided a good representation of real-world network traffic scenarios. Big Data analytics is one of the main challenges facing the domain of cybersecurity in recent times. This is because of the rapid growth in the amount of data that are generated daily and the required expertise for data analytics. The increasing adoption of artificial intelligence in solving complex problems in different domains has made it possible for machine learning and deep learning implementation to gain wider adoption in cybersecurity.

For this study, the raw dataset from CAIDA was used. The raw nature of the data posed a lot of refactoring challenges for the researcher. Turning the unstructured dataset into usable sets of data for deep learning algorithms is the most appropriate thing for this study. The data wrangling part covered about 60% of the whole time that was spent on this research. The initial research hypotheses included a section that plans to use a confusion matrix in measuring the performance of the existing deep learning model that was adopted for this study. However, after getting preliminary results from the experiments of this study, it was discovered that the confusion matrix cannot be used because of the raw and unclassified nature of the dataset that was used. There was no way to calculate true positive and false positive using the available data because anomalies have not been previously discovered or classified. The hypothesis eventually changed to using mean absolute error and root mean square error to estimate the accuracy of the model.

As discussed in the previous chapter, the accuracy of this implementation was high. This is a significant milestone in addressing the problem of big data analytics in cybersecurity. This proves that many of the attacks that are currently being experienced in large networks can be tackled by using artificial intelligence-based algorithms.

5.2 Community Detection of Anomalies

One of the major interests of this study is to understand the relationship that may exist between threat actors in large networks. Providing a general overview of linkages between compromised devices on the network can lead to the provision of fast solutions for safeguarding the whole network. This is synonymous with what is called “contact tracing” activity which

became a very significant method in stemming the tide of the recent COVID-19 infection (Tanaka, Ramachandran & Krishnamachari, 2020 and Wang, Lin, Obaidat, Yu, Wei & Zhang 2021). A node that is observed to be in proximity with another infected node is assumed to be likely infected or compromised also. These can be separated for further vulnerability assessment.

This study was able to provide a CDoA model which combined the deep learning model with a community detection approach that can be used to provide a first-hand overview of the relationship among different sources of threats to a security responder. The use of Louvain and Pagerank algorithms demonstrated that attacks in large networks can be ranked according to their levels of severity based on the number of connections their host has.

The result of the experiments indicates that CDoA can be used to provide quick guidance on priority areas or areas of concentration to cybersecurity specialists in a cyber-attack scenario. For example, Figure 5.1 shows the modularity class of communities that exist in the anomaly dataset that was used in this study. The class with ID 239 indicates the community covers 10.41% of nodes that made up the network. The severity of attacks on a larger community is high compared to the severity of attacks on the class with ID 97 which shows that the class only covers 0.33% of the nodes present in the large network. Each of the classes is represented with different colors as previously displayed in Figure 4.10a & b and Figure 4.11. The classes are arranged according to their sizes, that is, the number of node connectivity each has in the overall network.

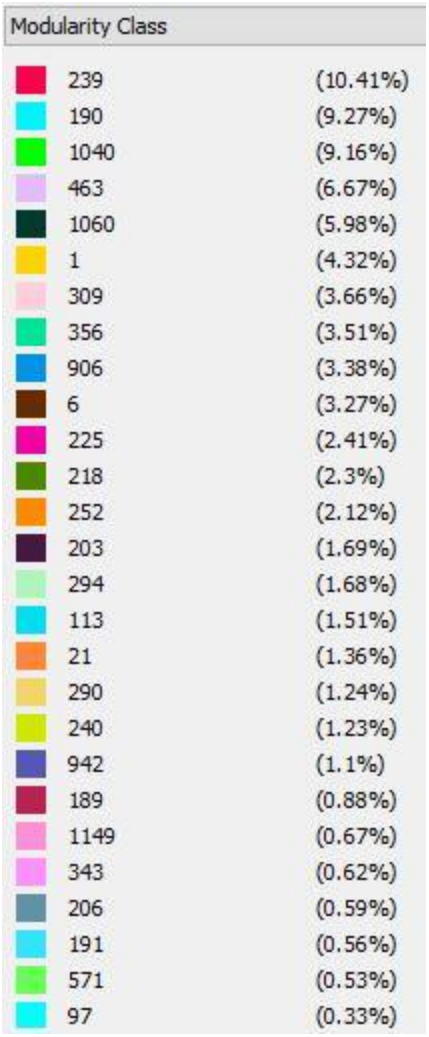


Figure 5.1: Ranking of communities discovered in the anomaly dataset

5.3 Contribution

Previous research studies have shown that variants of machine learning and deep learning algorithms can be combined to provide robust anomaly detection models for network traffic. Anomaly detection models with deep learning have evolved from semi-supervised models, unsupervised models to other combinations like hybrid models and one-class neural networks (Ruff, Kauffmann, Vandermeulen, Montavon, Samek, Kloft, Dietterich & Müller (2021); Chalapaty & Chawla (2019); Singh, Hand & Alexis (2020); Kabir & Luo (2020); Dawoud, Shahristani & Raun (2018)).

However, to the best of the researcher's knowledge, very few or no research studies have combined community detection algorithms with deep learning anomaly detection to study

anomalies in unstructured data of large-scale networks. Most studies have used existing data that have been classified into benign or malignant data for their studies. Others have collected both normal data and injected data in a controlled environment for their studies. Very few or no researchers are known to have used uncontrolled 'wild' datasets for anomaly detection study.

The main contribution of this research is to fill the existing gap of combining proven anomaly and community detection algorithms on an unstructured dataset to study the relationship between identified anomalies in a large-scale network. For example, most anomaly detection models have been used to classify anomaly detection, few or no one was known to have moved beyond model training, identification, and classification of the anomalies to the investigation of the relationship between the anomalies.

The high and quick overview of packets sent or received by each node in a large-scale network as shown in Figure 5.2 is one of the main contributions of this work. It shows the load distribution on the network per node. For example, the IP address 16.136.231.132 has sent out more than 38 million packets within few hours of captured traces, this within a network calls for great concern.

Visualization of such a source and others like it will provide quick direction of threat investigation to a cybersecurity specialist. It will make it easier for a cybersecurity specialist to investigate large networks in time and plan adequate remediation. The nodes with high packet transfer rates can represent a case of data infiltration or distributed denial of service attacks if such devices with high packet rate transmission are not known to be load balancers or servers on the given network. It can also represent the presence of botnet activities based on the number of devices it is connected to. The size of each circle in the diagram represents the total length of packets being sent out during the observed hours. The number represents the IP address of the devices on the network. I used the same color for all the nodes to show that they have similar attributes on the network.

Depending on the characteristics of the data points of the anomaly dataset, each modularity class of the dataset can also be used to represent different threats that may be existing in the large network as shown in Figure 5.3. It illustrates an example of how the CDoA model could be used to present a quick overview of known, discovered, or confirmed threats in a typical large-scale network traffic dataset.

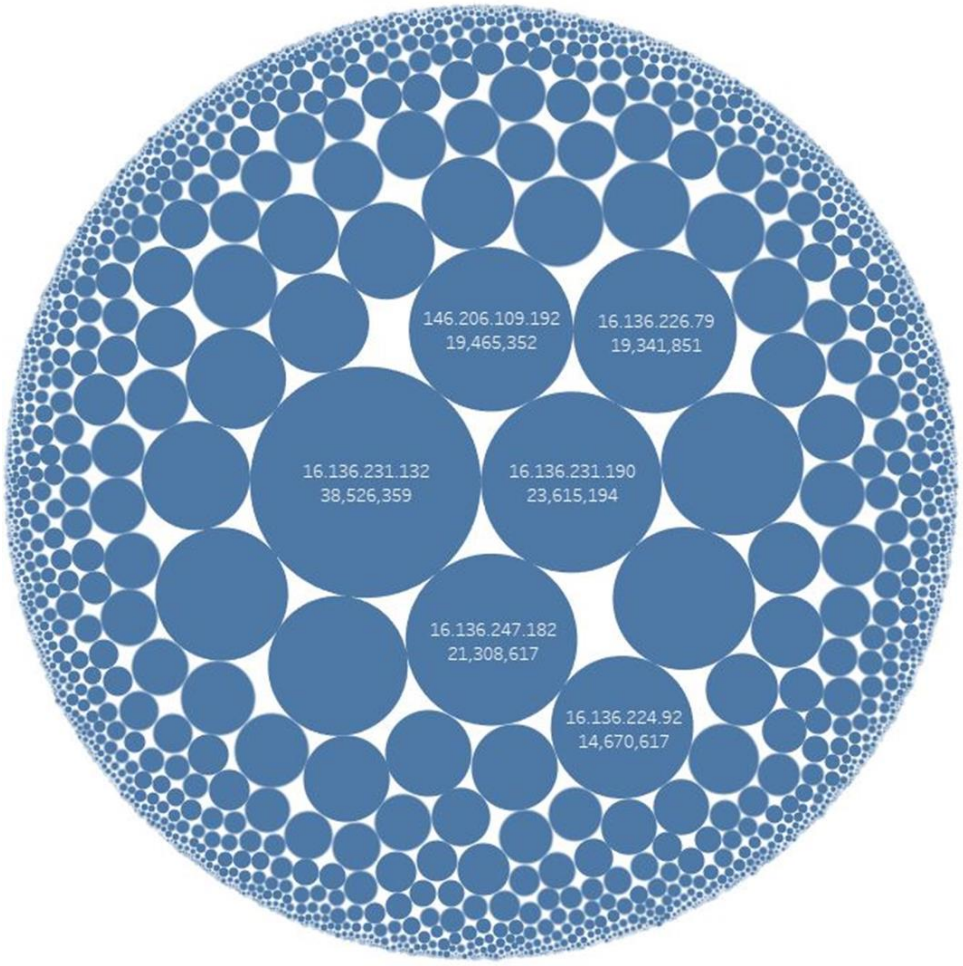


Figure 5.2: Visualization of packets distribution by nodes

The broader aim and impact of this study is to provide sophisticated, AI-assisted countermeasures to cyber-threats in large-scale networks. To close the existing gaps created by the shortage of skilled and experienced cybersecurity specialists and analysts in the cybersecurity field, solutions based on out-of-the-box thinking are inevitable; this research aimed at yielding one of such solutions. It was built to detect specific and collaborating threat actors in large networks and to help speed up how the activities of anomalies in any given large-scale network can be curtailed in time.

5.4 Future Research Direction

Future research studies should focus on using other parameters in the network for community detection of anomaly study. Other parameters such as protocols, the pattern of data

distribution, IP length, data frame time, and IP versions can be used as a basis for anomaly detection and community detection in large-scale networks in future studies. Moreover, it would be interesting to use the CDoA model on a pre-classified anomaly dataset. This will make it possible to use the confusion matrix as one of the metrics of evaluation of this model.

Future research studies should also consider using the hybrid deep learning model in conjunction with a community detection algorithm to build a CDoA model to check if this would improve the performance of the model. One other approach is to vary the community detection algorithm to test for the effectiveness of the CDoA model.

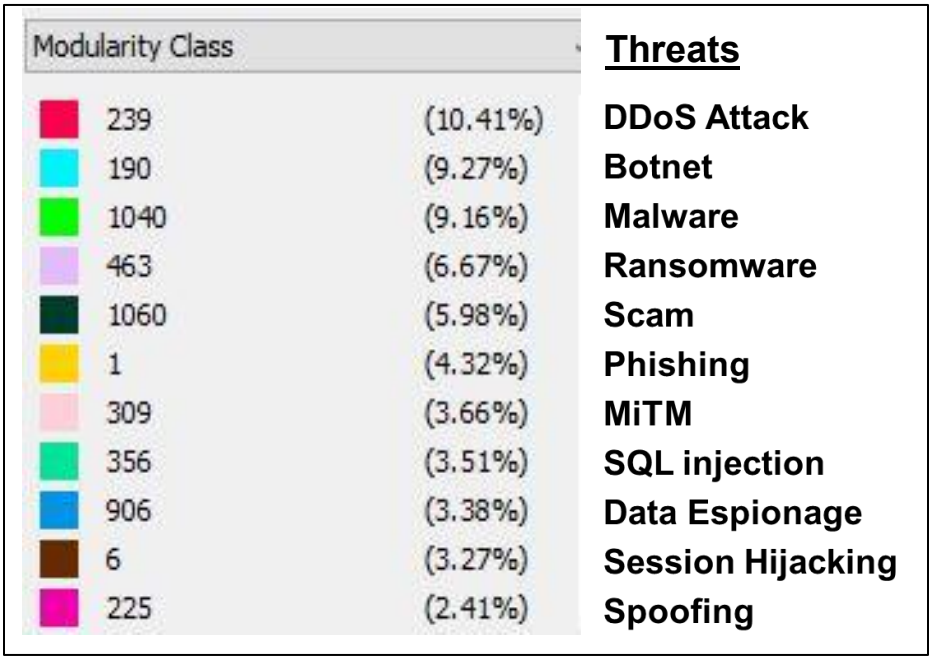


Figure 5.3: Representation of the Real-life Application of the CDOA Model Output

REFERENCES

Abbe E. & Sandon C. (2015). Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. *In IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS'15)*. 670–688. Retrieved from DOI:http://dx.doi.org/10.1109/FOCS.2015.47

Aditya, V., Dhuli, S., Sashrika, P. L., Shivani, K. K., & Jayanth, T. (2020). Closeness Centrality Based Cluster Head Selection Algorithm for Large-scale WSNs. *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*. doi:10.1109/cicn49253.2020.9242639

Alla, S., & Adari, S. K. (2019). Beginning Anomaly Detection Using Python-based Deep Learning. doi:10.1007/978-1-4842-5177-5

Araujo, M. (2017). Communities and Anomaly Detection in Large Edge-labeled Graphs. Ph.D. Thesis. Retrieved from https://repository.lib.ncsu.edu/.../1840.../Community-basedAnomalyDetection.pdf?

Babichev, S., Taif, M. A., & Lytvynenko, V. (2016). Inductive model of data clustering based on the agglomerative hierarchical algorithm. *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)*. doi:10.1109/dsmp.2016.7583499

Barber, M. J. (2007). Modularity and community detection in bipartite networks. Physical Batagelj V, Zaversnik M (2003) An O(m) algorithm for cores decomposition of networks. Eprint arXiv:cs/0310049

Bedi, P., & Sharma, C. (2016b). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 6*(3), 115–135. Retrieved from https://doi.org/10.1002/widm.1178

Bendre M. R., Thool R. C. & Thool V. R. (2015). Big data in precision agriculture: Weather forecasting for future farming. 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, pp. 744-750. DOI: 10.1109/NGCT.2015.7375220

Benson A. R., Gleich D. F., & Leskovec J. (2016). Higher-order organization of complex networks. *Science 353, 6295* (2016), 163–166. Retrieved from DOI:http://dx.doi.org/10.1126/science.aad9029.

Berman D., Buczak A., Chavis J., & Corbett C. (2019). A Survey of Deep Learning Methods for Cyber Security. Information, vol. 10, no. 4, p. 122.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008. Retrieved from https://doi.org/10.1088/1742-5468/2008/10/P10008

Bolaji A. (2018). Gephi visualization of communities in a graph. Unpublished activities in CGT 581, Spring Semester, Purdue University, Indiana.

Bostani H. & Sheikhan M. (2017). Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. Comput. Commun., 98, pp. 52-71

Chaabene, N. E., Bouzeghoub, A., Guetari, R., & Ghezala, H. H. (2021). Deep learning methods for anomalies detection in social networks using multidimensional networks and multimodal data: A survey. *Multimedia Systems*. doi:10.1007/s00530-020-00731-z

Chakraborty T., Dalmia A., Mukherjee A. & Niloy G. (2017). Metrics for Community Analysis: A Survey. ACM Comput. Surv. 50 (4), 54:1–54:37.

Chalapathy R. & Chawla S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

Chandola, Banerjee & Kumar (2007). Anomaly Detection: A Survey *ACM Computing Surveys (CSUR),41*(3), p.1-58,

Chen Z., Hendrix W., & Samatova N. F. (2011). Community-based anomaly detection in evolutionary networks. Journal of Intelligent Information Systems, vol. 39, no. 1, pp. 59–85.

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. Retrieved from https://arxiv.org/pdf/cond-mat/0408187.pdf.

Commons.wikimedia.org. Retrieved from: https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png

Constantin L. (2016). Armies of hacked IoT devices launch unprecedented DDoS attacks. Retrieved from https://www.networkworld.com/article/3123817/security/armies-of-hacked-iot-devices-launch-unprecedented-ddos-attacks.html

CrowdStrike Inc. (2020). 2020 CrowdStrike Global Threat Report. Retrieved from https://www.crowdstrike.com/resources/reports/2020-crowdstrike-global-threat-report/

Deng G, Ma Z, & Li X. (2018). Overlapping community detection algorithm based on Markov chain clustering. *IEEE 3rd International Conference on Big Data Analysis (ICBDA), Shanghai*, 458-462.

Deshmukh, H. V. (2018). Community Detection in Cyber Networks. Theses and Dissertations. Available from ProQuest. Retrieved from https://docs.lib.purdue.edu/dissertations/AAI10808294

Do, E. & Gadepally, V. (2020). Classifying Anomalies for Network Security. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). doi:10.1109/icassp40776.2020.9053419

Domingos P. & Hulten G. (2001). A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. In Proceedings of the Eighteenth International Conference on Machine Learning. Pg. 106-113.

Donetti L. & Munoz M. (2004). Detecting network communities: a new systematic and efficient algorithm. J Stat Mech P10012. doi:10.1088/1742-5468/2004/10/P10012

Fagnan J., Abnar A., Rabbany R. & Zaiane O.R. (2018) Modular Networks for Validating Community Detection Algorithms. Retrieved from arxiv:1801.01229 [physics]. 1801.01229.

Faltings B., Leyton-Brown K., & Ipeirotis P. (Eds.) (2012). Conference Proceeding, *ACM Conference on Electronic Commerce (EC'12). ACM*. Retrieved from http://dl.acm.org/citation.cfm?id=2229012

Fan, H., Zhang, F., & Li, Z. (2020). Anomalydae: Dual Autoencoder for Anomaly Detection on Attributed Networks. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. doi:10.1109/icassp40776.2020.9053387

Farias, G., Fabregas, E., Dormido-Canto, S., Vega, J., & Vergara, S. (2020). Automatic recognition of anomalous patterns in discharges by recurrent neural networks. Fusion Engineering and Design, 154, 111495. doi:10.1016/j.fusengdes.2020.111495.

Fortunato S. (2010). Community detection in graphs. *Physics Reports 486, (3–5*), 75–174.

Gan G., Ma C., & Wu J.(2007). Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability), SIAM.

Gendreau,A. A. & Moorman, M. (2016) Survey of intrusion detection systems towards an end to end secure internet of things. In 2016 IEEE 4th international conference on future internet of things and cloud (FiCloud) 2016 (pp. 84–90).

Girvan M. & Newman M. (2002). Community structure in social and biological networks. Proc Natl Acad Sci USA 99(12):7821–7826.

Gkantsidis C., Mihail M., & Zegura E. (2003). Spectral analysis of internet topologies. In Proceedings of the 22nd joint Conference of the Computer and Communications Societies (INFOCOM), pp. 364–374, San Francisco, United States. IEEE.

Grau, A. (2015). "Can you trust your fridge?" *IEEE Spectrum, 52*(3), 50-56.

Greene T. (2016). Largest DDoS attack ever delivered by botnet of hijacked IoT devices. Retrieved from https://www.networkworld.com/article/3123672/security/largest-ddos-attack-ever-delivered-by-botnet-of-hijacked-iot-devices.html

Gregory S. (2009). Finding overlapping communities in networks by label propagation. Eprint arXiv: 0910.5516.

Günnemann S., Färber I, Boden B., & Seidl T. (2014). Gamer: a synthesis of subspace clustering and dense subgraph mining. Knowledge and Information Systems (KAIS), 40(2):243–278.

Gupta B., Tewari A, Jain A. & Agrawal D. (2016). Fighting against phishing attacks: state of the art and future challenges, *Neural Computing and Applications 28*(12), p.3629-3654. Retrieved from doi>10.1007/s00521-016-2275-y

Harish V.K. (2016). An Anomaly-Based Intrusion Detection System Based on Artificial Immune System (AIS) Techniques. Theses and Dissertations. Available from ProQuest. Retrieved from https://docs.lib.purdue.edu/dissertations/AAI10808294

Hoque, N., Bhuyan, M. H., Baishya, R. C., Bhattacharyya, D., & Kalita, J. K. (2014). Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications, 40* , 307-324.

Huang, L., Wang, C., & Chao, H. (2018). Overlapping Community Detection in Multi-view Brain Network. 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). doi:10.1109/bibm.2018.8621075.

Hussain, F., Abbas, S. G., Fayyaz, U. U., Shah, G. A., Toqeer, A., & Ali, A. (2020). Towards a Universal Features Set for IoT Botnet Attacks Detection. doi:10.21203/rs.3.rs-114467/v1

Jie, F., Wang, C., Chen, F., Li, L., & Wu, X. (2020). A Framework for Subgraph Detection in Interdependent Networks via Graph Block-Structured Optimization. *IEEE Access, 8*, 157800-157818. doi:10.1109/access.2020.3018497

Kai, K.S.B., Singtel E. C., & Balachandran V. (2020) Anomaly Detection on DNS Traffic using Big Data and Machine Learning. Retrieved from http://ceur-ws.org/Vol-2622/paper14.pdf

Katz J. & Aakhus M. (2001). Perpetual Contact: mobile communication, private talk, public performance. Cambridge University Press, New York.

Kimberly G., (2010). Official Certified Ethical Hacker Review Guide. Second Edition. Wiley Publishing, Inc., Indianapolis, Indiana.

Li M. & Liu J. (2018). A link clustering based memetic algorithm for overlapping community detection. *Physica A 503* (2018) 410–423.

Liao, W., Deng, K., & Wang, S. (2019). Community detection based on Graph Coloring. *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. doi:10.1109/ssci44817.2019.9002759

Mahoney M. W., Dasgupta A., Lang K. J., & Leskovec J. (2009). Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics 6*(1), 29–123.

Maimo L., Gomez A., Clemente F., Perez M., & Perez G.M (2018). A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks IEEE Access, 6, pp. 7700-7712.

Malhotra, P., Vig L., Gautam S. & Agarwal P. (2015). Long short term memory networks for anomaly detection in time series. In ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

Marcus A., (2013). Internet Security Overview. Workshop Notes, Africans Network Operators Group. Lusaka, Zambia. Retrieved from https://nsrc.org/afnog

Marder, A., Luckie, M., Huffaker, B., & Claffy, K. (2020). Vrfinder. *ACM SIGMETRICS Performance Evaluation Review*, *48*(1), 55-56. doi:10.1145/3410048.3410080

Marteau, P. (2021). Random Partitioning Forest for Point-Wise and Collective Anomaly Detection—Application to Network Intrusion Detection. *IEEE Transactions on Information Forensics and Security*, *16*, 2157-2172. doi:10.1109/tifs.2021.3050605

Meidan Y., Bohadana M., Mathov Y., Mirsky Y., Breitenbacher D., Shabtai A., & Elovici Y., (2018). "N-baiot: Network-based detection of iot botnet attacks using deep autoencoders," CoRR, vol. abs/1805.03409, [Online]. Available at: http://arxiv.org/abs/1805.03409

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: an ensemble of autoencoders for online network intrusion detection. In: Network and Distributed System Security (NDSS) Symposium, San Diego, CA, USA

Moradi, F. (2014). Improving Community Detection Methods for Network Data Analysis. http://www.cse.chalmers.se/~tsigas/papers/Farnaz-Thesis.pdf.

Newman M. E. J. (2004). Detecting community structure in networks. *EPJB 38* (2), 321–330.

Newman M. E. J. (2006). Modularity and community structure in networks. PNAS 103(23), 8577–8582. Retrieved from https://www.pnas.org/content/103/23/8577

Newman, M., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E, 69, 1–16*. Retrieved from https://doi.org/10.1103/PhysRevE.69.026113.

Noble C. & Cook D. (2003). Graph-based anomaly detection. *In KDD. ACM*, 631–636. Retrieved from https://dl.acm.org/citation.cfm?id=956831

Nugraha, A., Perdana, M. A., Santoso, H. A., Zeniarja, J., Luthfiarta, A., & Pertiwi, A. (2018). Determining The Senior High School Major Using Agglomerative Hierarchial Clustering Algorithm. *2018 International Seminar on Application for Technology of Information and Communication*. doi:10.1109/isemantic.2018.8549834

Orgaz B., Salcedo-Sanz G. & Camacho D. (2018). A Multi-Objective Genetic Algorithm for overlapping community detection based on edge encoding. Information Sciences. 462. 290-314. 10.1016/j.ins.2018.06.015.

Page Lawrence, Brin Sergey, Motwani Rajeev, & Winograd Terry (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Papadopoulos, S., Kompatsiaris, Y., Vakali, A., & Spyridonos, P. (2011). Community detection in Social Media. Data Mining and Knowledge Discovery, 24(3), 515-554. doi:10.1007/s10618-011-0224-z.

Patterson, J. & Gibson, A. (2017). Deep Learning. A Practitioner's Approach; O'Reilly Media, Inc.: Sebastopol, CA, USA, pp. 6.

Peddabachigari, S., Abraham, A., Grosan, C., & Thomas, J. (2007). Modeling intrusion detection system using hybrid intelligent systems. *Journal of network and computer applications*, *30*(1), 114-132.

Pons, P., & Latapy, M. (2006). Computing Communities in Large Networks Using Random Walks. *Journal of Graph Algorithms and Applications, 10*(2), 191–218. Retrieved from http://www.liafa.jussieu.fr/

Porter M.A., Onnela J.-P., & Mucha P.J. (2009). Communities in networks. Notices of the American Mathematical Society, 56, pp. 1082-1166

Raghavan, U., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E, 76*(3), 36106. Retrieved from https://doi.org/10.1103/PhysRevE.76.036106

Raj Abhi (2016). Major cyber-attack disrupts internet service across Europe and US – Dyn DDoS. Retrieved from https://securityzap.com/major-cyber-attack-disrupts-internet/.

Randall, A., Liu, E., Akiwate, G., Padmanabhan, R., Voelker, G. M., Savage, S., & Schulman, A. (2020). Trufflehunter. *Proceedings of the ACM Internet Measurement Conference*. doi:10.1145/3419394.3423640

Reddy, G. & Reddy G. (2014). A study of cyber security challenges and its emerging trends on latest technologies. Retrieved from arXiv preprint arXiv: 1402.1842.

Reichardt J. & Bornholdt S. (2006). Statistical mechanics of community detection. Phys Rev E 74:016110. Review E - Statistical, Nonlinear, and Soft Matter Physics, 76(6). Retrieved from https://journals.aps.org/pre/abstract/10.1103/PhysRevE.74.016110

Rhodes-Ousley M., (2013). Information Security: The Complete Reference. Second Edition. The McGraw-Hill Companies.

Rosvall M. & Bergstrom C. T., (2007). *Proc. Natl. Acad. Sci. USA 104*, 7327

Roy, M. S., Roy, B., Gupta, R., & Sharma, K. D. (2020). On-Device Reliability Assessment and Prediction of Missing Photoplethysmographic Data Using Deep Neural Networks. IEEE Transactions on Biomedical Circuits and Systems, 14(6), 1323-1332. doi:10.1109/tbcas.2020.3028935

Sarswat A. & Guddeti R. M. R. (2018). A novel overlapping community detection using parallel CFM and sequential Nash equilibrium. *10th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, pp. 649-654.

Scheitle Q., Gasser O., Emmerich P., & Carle Georg (2016). Carrier-Grade Anomaly Detection Using Time-to-Live Header Information. Retrieved from arXiv:1606.07613v1 [cs.NI].

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Netw., 61, 85–117.

Sen, T., Chaudhary, D. K., & Choudhury, T. (2017). Modified Page Rank Algorithm: Efficient Version of Simple Page Rank with Time, Navigation and Synonym Factor. *2017 3rd International Conference on Computational Intelligence and Networks (CINE)*. doi:10.1109/cine.2017.24

Shao, M., Li, J., Chen, F., & Chen, X. (2018). An Efficient Framework for Detecting Evolving Anomalous Subgraphs in Dynamic Networks. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. doi:10.1109/infocom.2018.8485830

Shi J. & Malik J. (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell. 22(8):888–905

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., & Edwards, S. T. (2017). Time Series Anomaly Detection: Detection of Anomalous Drops with Limited Features and Sparse Examples in Noisy

Sikorski, M., & Honig, A. (2012). Practical Malware Analysis (1st Ed., Vol. 1). San Francisco, CA: No Starch Press.

Singh, D. K., Haraty, R. A., Debnath, N. C., & Choudhury, P. (2020). An Analysis of the Dynamic Community Detection Algorithms in Complex Networks. *2020 IEEE International Conference on Industrial Technology (ICIT)*. doi:10.1109/icit45562.2020.9067224

Sorrell Steffen (2018). Juniper Research "IOT ~ The Internet of Transformation 2018". Retrieved from https://www.juniperresearch.com/document-library/white-papers/iot-the-internet-of-transformation-2018.

Sun, H., He, F., Huang, J., Sun, Y., Li, Y., Wang, C., . . . Jia, X. (2020). Network Embedding for Community Detection in Attributed Networks. *ACM Transactions on Knowledge Discovery from Data, 14*(3), 1-25. doi:10.1145/3385415

Tan Y., Gu X, & Wang H. (2010). Adaptive system anomaly prediction for large-scale hosting infrastructures. Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing. pp. 173-182.

Tanaka N., Ramachandran G. Krishnamachari & B. (2020). Poster: Centralized vs. Decentralized Contact Tracing: Do GDP and Democracy Index Influence Privacy Choices? IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), Crystal City, VA, USA, 2020, pp. 14-1.

The CAIDA UCSD Statistical information for the CAIDA Anonymized Internet Traces. Retrieved from https://www.caida.org/data/passive/passive_trace_statistics.xml. [Accessed 04/12/2019].

Tong, H., Faloutsos, C. & Pan, J. (2008). Knowl Inf Syst 14: 327. Retrieved from https://doi.org/10.1007/s10115-007-0094-2

Usha, M., & Nagadeepa, N. (2018). Combined two phase page ranking algorithm for sequencing the web pages. *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. doi:10.1109/icisc.2018.8398925

Von Luxburg U. (2006). A tutorial on spectral clustering. Technical report 149. Max Planck Institute for Biological Cybernetics, August 2006.

Wang J. & Paschalidis I. (2017). Botnet Detection Based on Anomaly and Community Detection. in IEEE Transactions on Control of Network Systems, 4(2), pp. 392-404.

Wang J. P., Lin C., Obaidat M. S., Yu Z., Wei Z. & Zhang Q. (2020). Contact Tracing Incentive for COVID-19 and other Pandemic Diseases from a Crowdsourcing Perspective," in IEEE Internet of Things Journal. Retrieve from doi: 10.1109/JIOT.2020.3049024.

Wang, F., Li, T., Wang, X., Zhu, S., & Ding, C. (2011). Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery, 22*(3), 493-521. doi:10.1007/s10618-010-0181-y

Wasserman S. & Faust K. (1994). Social network analysis: methods and applications. Cambridge University Press, Cambridge.

Wegner D. (1995). A Computer Network Model of Human Transactive Memory. *Social Cognition: 13*(3), pp. 319-339. https://doi.org/10.1521/soco.1995.13.3.319

Gao, J., Song, Wen Q., Wang X., Sun, L., Xu, H., & Zhu, S. (2020). RobustSTL: A Robust Seasonal-Trend Decomposition Algorithm for Long Time Series. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 5409-5416. doi:10.1609/aaai.v33i01.33015409

Wu, Y., Wei, D., & Feng, J. (2020). Network Attacks Detection Methods Based on Deep Learning Techniques: A Survey. *Security and Communication Networks*, *2020*, 1-17. doi:10.1155/2020/8872923

Yamada, R., & Goto, S. (2013). Using abnormal TTL values to detect malicious IP packets. *Proceedings of the Asia-Pacific Advanced Network*, *34*(0), 27. doi:10.7125/apan.34.14

Ye, F., Chen, C., Zheng, Z., Li, R., & Yu, J. (2019). Discrete Overlapping Community Detection with Pseudo Supervision. *2019 IEEE International Conference on Data Mining (ICDM)*. doi:10.1109/icdm.2019.00081

Ye, F., Li, S., Lin, Z., Chen, C., & Zheng, Z. (2018). Adaptive Affinity Learning for Accurate Community Detection. *2018 IEEE International Conference on Data Mining (ICDM)*. doi:10.1109/icdm.2018.00188

Zhou Xu, Liu Yancheng, Wang Jian & Li Chun (2017). A density-based link clustering algorithm for overlapping community detection in networks. *Physica A 486*, 65–78.

Zhou, Y., & Li, J. (2019). Research of Network Traffic Anomaly Detection Model Based on Multilevel Autoregression. *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. doi:10.1109/iccsnt47585.2019.89625

Zurier S. (2021) Payment processor used by government hit by ‘Cuba’ ransomware gang. Retrieved from https://www.scmagazine.com/home/security-news/payment-processor-used-by-state-municipal-agencies-hit-by-cuba-ransomware-gang/

APPENDIX A. CODES

A.1 Unzipping Multiple PCAP Files From CAIDA

The following source code was used to unzip the pcap files.

```
from platform import python_version


print(python_version())
# app.py

import zipFile
import os
import gzip
import shutil

os.chdir("/scratch/gilbreth/abolaji/passive-2019/equinix-nyc/20190117-
130000.UTC/" )

search_path = os.getcwd()

file_type = ".gz"

for fname in os.listdir(path=search_path):

    if fname.endswith(file_type):

        with gzip.open(fname,'rb') as f_in:

            with open(fname+'.pcap','wb') as f_out:

                shutil.copyfileobj(f_in,f_out)
```

A.2 Extracting needed Data from PCAP Files into CSV

The following source code was used to extract the files to csv on the cluster.

```
#!/bin/sh -l

#############################################################################

#

# Number of cores and gpus

# Sub-Cluster A: 20, 2

# Sub-Cluster B: 24, 2
```

```
# Sub-Cluster C: V100 GPUs 20, 4
# Sub-Cluster D: 16, 2
# Sub-Cluster E: V100 GPUs 16, 2
# F: 40, ?
##############################################################################
#
#SBATCH --job-name=tshark-convert
##SBATCH --mail-type=ALL
#SBATCH --mail-type=END
#SBATCH --mail-user=jaspring@purdue.edu
#SBATCH --account=partner
#SBATCH --time=23:59:00
#SBATCH --nodes=1
#SBATCH --gpus-per-node=1
#SBATCH --output=%x.%j.out
#SBATCH --mem-per-cpu=64GB
tshark -r $FILENAME -T fields -e ip.len -e ip.len -e ip.id -e ip.ttl -e ip.proto -e ip.src -e ip.dst -e
tcp.srcport -e tcp.dstport -e tcp.seq -e tcp.len -e tcp.stream -e tcp.time_relative -e tcp.time_delta -E
header=y -E separator=, -E quote=d -E occurrence=f > $FILENAME.csv
```

Script:

```
#!/bin/sh
for filename in *.pcap
do
sbatch --export=ALL,FILENAME=$filename tshark.command.sub
done
```

A.3 Merging the CSV Files into a Single File

```
import os
import glob
import pandas as pd
os.chdir(r"/depot/datalab/bolaji/Anoms/")
extension = 'csv'
```

```
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
Final = (pd.concat([pd.read_csv(f) for f in all_filenames]))
Final.to_csv("AnomaliesAll2.csv", index=False, sep=',',encoding='utf-8-sig
')
```

A.4 Code for LSTM Deep Learning Anomaly Detection Model

/**

* This code is an adaptation of the code from a book

*Title: Beginning Anomaly Detection Using Python-Based Deep Learning

*Author: Alla, S., & Adari, S. K.

*Date:2019

*Code version:

*Availability: Link

**/

```
import sys
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
print("pandas: ", pd.__version__)
print("seaborn: ", sns.__version__)
print("matplotlib.pyplot: ", plt.__version__)
filename= r"/depot/datalab/bolaji/A1/equinix-nyc.dirA.20190117-125910.UTC.
anon.pcap.csv"
df =pd.read_csv(filename,error_bad_lines=False, engine="python")
data_s = df[['frame.time','ip.len','ip.src','ip.dst','ip.ttl']].copy()
print('Shape:' , data_s.shape[0])
data_s = data_s.dropna()
```

```
print('Shape:' , data_s.shape[0])

data_s["frame.time"].str[:-4].astype("datetime64[ns]")

fig, (ax1) = plt.subplots(ncols=1, figsize=(8, 5))

ax1.set_title('Before Scaling')

sns.kdeplot(data_s['ip.ttl'], ax=ax1)
class Visualization:

    labels = ["Normal", "Anomaly"]

    def draw_anomaly(self, y, error, threshold):

        groupsDF = pd.DataFrame({'error': error,

                                 'true': y}).groupby('true')


        figure, axes = plt.subplots(figsize=(12, 8))


        for name, group in groupsDF:

            axes.plot(group.index, group.error, marker='x' if name == 1 el
se 'o', linestyle='',

                      color='r' if name == 1 else 'g', label="Anomaly" if na
me == 1 else "Normal")

        axes.hlines(threshold, axes.get_xlim()[0], axes.get_xlim()[1], col
ors="b", zorder=100, label='Threshold')

        axes.legend()

        plt.title("Anomalies")

        plt.ylabel("Error")

        plt.xlabel("Data")

        plt.show()


    def draw_error(self, error, threshold):
```

```
        plt.figure(figsize=(10, 8))
        plt.plot(error, marker='o', ms=3.5, linestyle='',
                 label='Point')
        plt.hlines(threshold, xmin=0, xmax=len(error)-1, colors="r", zorde
r=100, label='Threshold')
        plt.legend()
        plt.title("Reconstruction error")
        plt.ylabel("Error")
        plt.xlabel("Data")
        plt.show()
i = 0


tensorlog = tensorlogs[i]
dataFilePath = dataFilePaths[i]
print("tensorlog: ", tensorlog)
print("dataFilePath: ", dataFilePath)
df = pd.read_csv(filepath_or_buffer=dataFilePath, header=0, sep=',')
print('Shape:' , df.shape[0])
print('Head:')
print(df.head(5))
df['Datetime'] = pd.to_datetime(df['frame.time'])
#print(df.head(3))
#df.shape
#df.plot(x='Datetime', y='ip.len', figsize=(12,6))
#plt.xlabel('Date time')
#plt.ylabel('ip.len')
#plt.title('Time Series of ip.len by date time')
```

```
fig, (ax1) = plt.subplots(ncols=1, figsize=(8, 5))
ax1.set_title('Before Scaling')
sns.kdeplot(df['ip.len'], ax=ax1)
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0, 1))
df['scaled_ip.len'] = pd.DataFrame(scaler.fit_transform(pd.DataFrame(df['i
p.len'])),columns=['ip.len'])
print('Shape:' , df.shape[0])
df.head(5)
fig, (ax1) = plt.subplots(ncols=1, figsize=(8, 5))
ax1.set_title('After Scaling')
sns.kdeplot(df['scaled_ip.len'], ax=ax1)
data_s = df[['Datetime','ip.len','ip.src','ip.dst','ip.ttl','scaled_ip.len
']].copy()
print('Shape:' , data_s.shape[0])
data_s.head(5)
time_steps = 48
metric = 'mean_absolute_error'


model = Sequential()
model.add(LSTM(units=32, activation='tanh', input_shape=(time_steps, 1), r
eturn_sequences=True))


model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='mean_absolute_error', metrics=[metri
c])
```

```
print(model.summary())
sequence = np.array(data_s['scaled_ip.len'])
print(sequence)
time_steps = 48
samples = len(sequence)
trim = samples % time_steps
subsequences = int(samples/time_steps)
sequence_trimmed = sequence[:samples - trim]


print(samples, subsequences)
sequence_trimmed.shape = (subsequences, time_steps, 1)
print(sequence_trimmed.shape)
training_dataset = sequence_trimmed
print("training_dataset: ", training_dataset.shape)


batch_size=32
epochs=100


history = model.fit(x=training_dataset, y=training_dataset,batch_size=batc
h_size,epochs=epochs,
          verbose=1,validation_data=(training_dataset,training_dataset))
```

```
acc = history.history['mean_absolute_error']
val_acc = history.history['val_mean_absolute_error']
loss = history.history['loss']
val_loss = history.history['val_loss']
plt.figure(figsize=(8, 8))
```

```
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Mean Absolute Error')
plt.plot(val_acc, label='Validation Mean Absolute Error')
plt.legend(loc='upper right')
plt.ylabel('Mean Absolute Error')
plt.title('Training and Validation Mean Absolute Error')
plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Mean Absolute Error')
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
import math
from sklearn.metrics import mean_squared_error


sequence = np.array(data_s['scaled_ip.len'])
print(sequence)
time_steps = 48
samples = len(sequence)
trim = samples % time_steps
subsequences = int(samples/time_steps)
sequence_trimmed = sequence[:samples - trim]


print(samples, subsequences)
sequence_trimmed.shape = (subsequences, time_steps, 1)
```

```
print(sequence_trimmed.shape)


testing_dataset = sequence_trimmed
print("testing_dataset: ", testing_dataset.shape)


testing_pred = model.predict(x=testing_dataset)
print("testing_pred: ", testing_pred.shape)


testing_dataset = testing_dataset.reshape((testing_dataset.shape[0]*testin
g_dataset.shape[1]), testing_dataset.shape[2])
print("testing_dataset: ", testing_dataset.shape)


testing_pred = testing_pred.reshape((testing_pred.shape[0]*testing_pred.sh
ape[1]), testing_pred.shape[2])
print("testing_pred: ", testing_pred.shape)
errorsDF = testing_dataset - testing_pred
print(errorsDF.shape)
rmse = math.sqrt(mean_squared_error(testing_dataset, testing_pred))
print('Test RMSE: %.3f' % rmse)


#based on cutoff after sorting errors
dist = np.linalg.norm(testing_dataset - testing_pred, axis=-1)


scores =dist.copy()
print(scores.shape)
scores.sort()
cutoff = int(0.999 * len(scores))
```

```
print(cutoff)

#print(scores[cutoff:])

threshold= scores[cutoff]

print(threshold)

plt.plot(testing_dataset, color='green')

plt.plot(testing_pred, color='red')

z = zip(dist >= threshold, dist)


y_label=[]

error = []

for idx, (is_anomaly, dist) in enumerate(z):

    if is_anomaly:

        y_label.append(1)

    else:

        y_label.append(0)

    error.append(dist)
```

A.5 Code for Louvain Algorithm

/**

* This code is an adaptation of the code from

*Title: Exploring and Analyzing Network Data with Python

*Author: John R. Ladd, Jessica Otis, Christopher N. Warren, and Scott Weingart

*Date:2020

*Code version:

*Availability: Link

**/

```
import networkx as nx
import pandas as pd
import csv
from operator import itemgetter
from networkx.algorithms import community

with open('/depot/datalab/bolaji/Anoms/AnoUni/Nodes.csv', 'r') as nodecsv:
 # Open the file
    nodereader = csv.reader(nodecsv) # Read the csv
    # Retrieve the data (using Python list comprehension and list slicing
to remove the header row, see footnote 3)
    nodes = [n for n in nodereader][1:]
node_names = [n[0] for n in nodes] # Get a list of only the node names
```

```
with open('/depot/datalab/bolaji/Anoms/AnoUni/Edges2.csv', 'r') as edgecsv
: # Open the file
    edgereader = csv.reader(edgecsv) # Read the csv
    edges = [tuple(e) for e in edgereader][1:] # Retrieve the data
G.add_nodes_from(node_names)
G.add_edges_from(edges)
print(nx.info(G))


import community as community_louvain
import matplotlib.cm as cm
import matplotlib.pyplot as plt
partition = community_louvain.best_partition(G)
%%time
# draw the graph
pos = nx.spring_layout(G)
# color the nodes according to their partition
cmap = cm.get_cmap('viridis', max(partition.values()) + 1)
nx.draw_networkx_nodes(G, pos, partition.keys(), node_size=40,
                       cmap=cmap, node_color=list(partition.values()))
nx.draw_networkx_edges(G, pos, alpha=0.5)
plt.show()
#Graphtype = nx.Graph()
#G = nx.from_pandas_edgelist(df, 'Source', 'Target', ['TTL'])
```

A.6 Code for PageRank Algorithm Implementation

/**

* This code is an adaptation of the code from

*Title: PageRank algorithm

*Author: Unknown.

*Date:2021

*Code version:

*Availability: Link

**/

```
from scipy.sparse import coo_matrix

import numpy as np

import csv

nodesID = {}

n = 0

line_count = 0

%%time

with open('/depot/datalab/bolaji/Anoms/AnoUni/Edges2.txt', 'r') as edges:

# Open the file

    for line in edges:

        line_count += 1

        if line.startswith('#') : continue

        tokens = line.strip().split('\t')

        #print tokens

        if tokens[0] not in nodesID:

            nodesID[tokens[0]] = n

            n += 1
```

```
        if tokens[1] not in nodesID:
            nodesID[tokens[1]] = n
            n += 1
col = []
row = []
value = []
line_count = 0
with open('/depot/datalab/bolaji/Anoms/AnoUni/Edges2.txt','r') as edges:
    for line in edges:
        line_count += 1
        if line.startswith('#') : continue
        tokens = line.strip().split('\t')
        url1 = nodesID[tokens[0]]
        url2 = nodesID[tokens[1]]

        col.append(url1)
        row.append(url2)
        value.append(1.0)

print (M)
M.shape
inLink = M.sum(1)
inLink
outLink = M.sum(0).T
outLink
outLink.shape
value = [1.0 / outLink[col[i], 0] for i, v in enumerate(value)]
```

```
M = coo_matrix((value, (row,col)), shape=(n, n))
print (M)
print (M.shape)
beta= 0.8
epsilon = 1./(10**11)
r = np.ones([n,1])
r = r/n
print (np.sum(r))
r
a_file = open(r"C:\Users\FOLA-BUNMI\Downloads\test.txt", "w")
for row in r:
    np.savetxt(a_file, row)
a_file.close()
for _ in range(250):
    old_r = r
    r  = beta * M * r
    for j in range(n):
        if inLink[j,0] == 0 :
            r[j] = 0
    S = r.sum()
    r = r +  (1 - S)/n

    if np.sum(np.abs(old_r - r)) < epsilon:
        print ("{} iterations".format(_))
        old_r = r
        break
    else:
```

```
        old_r = r


#print((r), [nodesID])
print(my_dict)
with open('test3.csv', 'w') as f:
    for key in my_dict.keys():
        f.write("%s,%s\n"%(key,my_dict[key]))
```

A.7 PageRank score for the first 1,000 nodes

SN	PageRank (r)	Source	Node No.
1	0.035017014	146.206.121.50	232
2	0.027379503	161.39.9.234	11
3	0.002681855	52.30.35.134	66
4	0.002519125	61.94.210.119	47
5	0.00179733	203.253.164.213	1392
6	0.001670975	161.72.147.65	3169
7	0.001534544	162.198.28.36	3958
8	0.001440519	16.136.249.221	695
9	0.001222921	161.90.117.125	3895
10	0.00106154	175.241.100.26	265
11	0.001007772	161.89.82.225	4385
12	0.000997523	154.158.26.111	3951
13	0.000862354	182.22.78.12	3201
14	0.000794486	136.9.100.66	935
15	0.000756392	45.174.112.221	402
16	0.000711271	162.198.30.92	758
17	0.000687681	175.84.137.216	1868
18	0.000596885	161.185.16.201	1815
19	0.000590405	161.159.172.201	41
20	0.000571148	203.253.164.194	1368
21	0.000568642	161.158.26.29	3851
22	0.000565097	161.89.80.40	137
23	0.000553186	113.201.161.82	1873
24	0.000552511	180.211.115.96	5567
25	0.000536379	161.190.162.49	96
26	0.000536379	161.40.201.135	841

27	0.000536379	161.195.232.249	1680
28	0.000529342	164.172.28.39	1799
29	0.000512181	146.206.109.220	578
30	0.000498565	16.136.243.221	492
31	0.000496054	52.76.82.239	12104
32	0.000455721	52.31.77.43	1701
33	0.000448053	177.125.196.20	12142
34	0.000439589	182.245.239.8	3526
35	0.000439589	173.251.247.153	5082
36	0.000431111	182.22.75.34	570
37	0.000428756	203.66.190.180	1317
38	0.00042678	122.252.42.103	1811
39	0.000419912	25.111.51.13	1509
40	0.000419262	171.222.165.65	2494
41	0.000399336	170.237.67.207	1499
42	0.000391194	173.136.104.188	734
43	0.000391194	161.5.164.130	1355
44	0.000384838	55.87.248.82	13299
45	0.000383128	170.230.43.141	1641
46	0.000375062	113.201.213.247	2043
47	0.000375062	162.160.48.202	5260
48	0.000366418	16.136.247.134	184
49	0.00035893	203.21.182.107	2246
50	0.00035893	175.84.140.91	4984
51	0.00035893	173.219.98.240	10293
52	0.000350864	170.199.92.207	8559
53	0.000341848	85.171.48.156	1917
54	0.00034175	167.104.87.169	993
55	0.000338054	180.229.155.199	21669
56	0.000327882	104.150.50.154	3446
57	0.000327389	175.68.135.97	443
58	0.000326667	161.5.163.239	1920
59	0.000326667	170.238.213.189	6704
60	0.000326667	167.6.25.195	10485
61	0.000325418	55.49.253.216	2115
62	0.000310535	179.163.208.82	4665
63	0.000294404	52.55.168.212	2129
64	0.000294404	175.240.30.234	30819
65	0.000289661	202.118.75.59	2266
66	0.000288426	171.168.6.173	3831
67	0.000285187	180.222.92.75	4968
68	0.000278477	173.178.117.220	174
69	0.000278272	180.211.112.179	423

70	0.000278272	161.185.236.13	3032
71	0.000278272	170.199.231.214	3739
72	0.000275602	161.146.182.57	5527
73	0.000264456	169.77.36.139	3001
74	0.000263042	203.93.247.140	1476
75	0.000262387	180.120.208.121	5011
76	0.00026214	13.71.229.205	2567
77	0.00026214	143.143.105.224	5532
78	0.000255822	221.228.62.130	1273
79	0.000250041	16.136.226.91	8298
80	0.000248631	146.206.134.241	4212
81	0.000246008	52.18.155.118	1833
82	0.000246008	13.73.120.195	2881
83	0.000246008	145.122.186.46	3891
84	0.000246008	175.84.132.251	4528
85	0.000246008	136.28.170.224	10316
86	0.000245928	170.238.248.102	2021
87	0.000243466	173.214.39.173	1351
88	0.00022997	34.245.72.251	3073
89	0.000229877	16.136.226.67	197
90	0.000229877	171.129.182.189	651
91	0.000229877	164.172.28.72	1591
92	0.000229877	175.84.149.163	2589
93	0.000229877	180.120.246.58	6700
94	0.000228969	113.201.206.108	2428
95	0.000226316	171.199.171.70	1383
96	0.000221612	182.10.60.120	1574
97	0.000218571	131.14.142.66	5484
98	0.000217637	13.240.165.102	2507
99	0.000215583	161.130.10.187	858
100	0.000213947	131.251.199.162	1726
101	0.000213745	66.35.161.128	9
102	0.000213745	203.34.60.4	1553
103	0.000213745	173.141.34.43	4948
104	0.000213745	135.56.168.159	5297
105	0.000213745	202.127.153.85	5409
106	0.000213745	34.205.112.197	10178
107	0.000210191	30.239.0.80	1478
108	0.000208456	175.84.162.191	1805
109	0.000205679	131.118.66.106	5929
110	0.000203663	175.84.134.121	8852
111	0.000197815	203.97.55.35	10971
112	0.000197613	182.250.32.50	6232

113	0.000197613	13.69.50.179	1571
114	0.000197613	175.84.141.231	7887
115	0.000197613	173.136.99.20	11099
116	0.000196486	2.229.3.110	1560
117	0.000193118	52.86.120.79	1612
118	0.000192255	175.84.136.230	4831
119	0.000190827	132.252.197.238	183
120	0.000189651	77.5.227.184	5432
121	0.000186528	171.234.246.224	951
122	0.000186206	161.80.108.160	7611
123	0.000184296	182.245.236.238	635
124	0.000181482	175.71.242.171	6044
125	0.000181482	161.53.112.202	18271
126	0.000179333	180.237.7.211	33692
127	0.000179054	175.84.161.233	15144
128	0.000176825	113.201.196.117	2239
129	0.000169371	203.253.164.1	1512
130	0.000168051	25.111.247.34	2081
131	0.000166083	69.27.140.65	10640
132	0.00016535	42.27.128.130	1124
133	0.00016535	162.246.10.100	4862
134	0.00016535	175.84.135.158	5095
135	0.00016535	175.68.195.147	6247
136	0.00016535	175.84.137.60	6733
137	0.00016535	175.84.165.100	7015
138	0.00016535	175.84.148.118	7558
139	0.00016535	169.26.85.233	9279
140	0.00016535	113.201.152.182	12986
141	0.000162322	202.140.70.56	4379
142	0.000159973	131.251.72.25	313
143	0.000157435	182.250.24.143	1135
144	0.000156686	45.235.3.175	873
145	0.000154997	173.136.111.55	1353
146	0.000150947	118.163.134.75	12206
147	0.000149878	175.84.142.153	3416
148	0.00014942	23.12.195.252	1377
149	0.00014942	113.201.225.140	19870
150	0.000149366	136.199.115.221	12204
151	0.000149218	69.35.152.139	6727
152	0.000149218	175.84.153.197	6941
153	0.000149218	47.36.84.65	8813
154	0.000148965	136.127.65.201	655
155	0.000148527	177.125.214.33	13177

156	0.000145991	161.88.188.196	561
157	0.000145646	153.62.125.73	33618
158	0.000144999	182.19.217.203	1577
159	0.000141669	161.80.67.136	10136
160	0.000140703	180.120.246.99	1896
161	0.000140115	52.18.143.139	1959
162	0.00013993	160.219.74.44	8001
163	0.000139351	173.31.235.58	919
164	0.000137816	16.136.205.126	399
165	0.000137707	158.48.141.194	1033
166	0.000136825	161.40.202.25	845
167	0.000135177	162.102.142.231	7445
168	0.000135035	203.253.164.220	1401
169	0.00013477	52.30.19.146	2395
170	0.000133993	161.145.237.152	28171
171	0.000133115	161.185.16.250	1887
172	0.000133087	175.84.139.106	2010
173	0.000133087	175.84.162.247	4905
174	0.000133087	161.146.139.21	9377
175	0.000133087	131.14.83.225	11812
176	0.000133087	72.106.33.179	12793
177	0.000133087	175.241.82.27	18244
178	0.000133022	161.215.87.178	2636
179	0.000132586	77.38.63.129	4901
180	0.000132243	55.44.142.203	2484
181	0.000131358	175.84.138.196	1666
182	0.000130255	136.227.171.156	1678
183	0.000129241	171.198.95.58	8598
184	0.000129233	56.55.110.215	2213
185	0.000129193	161.195.235.72	829
186	0.000128821	180.212.185.134	1662
187	0.000127709	126.74.191.161	932
188	0.000126757	177.125.214.2	2806
189	0.000125722	162.102.134.75	472
190	0.000125675	161.190.11.221	16006
191	0.000125406	175.70.102.10	2181
192	0.000125089	195.192.187.139	838
193	0.000125021	162.160.50.34	6401
194	0.000124732	34.221.96.236	3160
195	0.000123742	161.185.226.108	1845
196	0.000122919	173.141.54.29	7032
197	0.000122332	135.56.237.29	5699
198	0.000118971	162.160.61.5	7170

199	0.000118893	66.84.56.228	4343
200	0.000118725	171.191.225.93	5788
201	0.000118221	175.84.142.234	4535
202	0.000117061	16.136.252.22	3751
203	0.000117	182.17.120.74	10611
204	0.000116955	72.106.32.148	105
205	0.000116955	177.125.214.123	2673
206	0.000116955	113.201.208.21	3139
207	0.000116955	70.251.239.212	7316
208	0.000116955	126.255.163.24	8874
209	0.000116955	161.195.225.62	14477
210	0.000116955	162.67.187.31	15942
211	0.000116955	180.120.208.81	16823
212	0.000116955	161.55.27.20	21149
213	0.00011695	99.11.157.14	9536
214	0.000114827	175.84.160.184	4171
215	0.000113794	77.226.254.216	34681
216	0.000113711	184.255.249.69	1670
217	0.000113055	170.199.90.7	393
218	0.000113039	171.198.7.158	663
219	0.000113005	61.18.60.155	12439
220	0.000112432	131.106.82.17	1790
221	0.000112295	175.84.135.67	1566
222	0.000112079	118.49.145.71	33627
223	0.00011168	113.201.183.224	13051
224	0.000111559	13.219.169.185	2786
225	0.000111158	34.210.255.150	1911
226	0.000110998	161.61.122.33	1102
227	0.000110156	175.70.12.172	13676
228	0.000109573	52.214.80.87	2513
229	0.000108573	69.35.151.255	7580
230	0.000108318	173.196.54.7	1071
231	0.000108292	182.232.27.224	62
232	0.000107326	171.198.20.109	2079
233	0.000107055	114.131.79.222	9292
234	0.000106382	175.84.138.68	3255
235	0.000106274	37.0.229.184	1251
236	0.0001062	45.224.106.3	6761
237	0.000106021	180.211.115.0	4646
238	0.000104003	161.30.51.62	35034
239	0.000102592	161.185.234.112	3011
240	0.00010236	175.71.184.115	2533
241	0.000101759	175.73.181.56	1221

242	0.000101695	52.30.20.243	20051
243	0.000101098	177.125.233.236	3064
244	0.000100823	42.27.130.233	1192
245	0.000100823	197.180.162.95	2501
246	0.000100823	66.134.187.165	3085
247	0.000100823	169.68.59.80	6239
248	0.000100823	173.141.0.93	6558
249	0.000100823	45.86.171.127	11402
250	0.000100823	161.40.201.214	14471
251	0.000100823	175.241.33.144	18756
252	0.000100823	113.201.214.206	20148
253	0.000100823	162.246.67.103	24134
254	0.00010055	161.5.166.179	3366
255	0.000100176	177.125.215.223	2553
256	0.000100159	175.84.140.107	755
257	9.97E-05	34.226.171.106	3081
258	9.93E-05	45.62.240.219	2116
259	9.88E-05	34.210.164.39	1860
260	9.86E-05	175.84.159.23	3284
261	9.85E-05	136.87.91.121	34822
262	9.75E-05	55.60.251.122	2886
263	9.74E-05	175.84.145.222	7678
264	9.73E-05	196.43.146.120	2370
265	9.72E-05	177.125.198.119	13117
266	9.58E-05	42.27.128.228	990
267	9.56E-05	175.84.147.224	4482
268	9.54E-05	178.57.62.237	4569
269	9.53E-05	161.185.239.157	1742
270	9.46E-05	153.62.125.77	6482
271	9.44E-05	175.69.146.15	9056
272	9.42E-05	223.231.151.122	590
273	9.30E-05	162.67.179.85	9005
274	9.22E-05	55.68.32.228	12638
275	9.21E-05	158.48.141.195	1089
276	9.16E-05	175.241.98.34	5101
277	9.14E-05	71.217.76.215	1035
278	9.11E-05	161.34.214.231	6009
279	9.09E-05	175.84.167.251	677
280	9.07E-05	161.80.15.194	7650
281	9.04E-05	161.80.21.16	29
282	8.99E-05	171.129.179.50	1091
283	8.95E-05	180.1.127.102	3457
284	8.92E-05	37.24.12.122	3978

285	8.92E-05	16.136.231.152	4144
286	8.88E-05	136.9.147.165	18398
287	8.87E-05	170.227.6.22	31763
288	8.81E-05	184.253.194.170	6433
289	8.79E-05	170.249.198.42	13048
290	8.73E-05	175.84.162.0	6389
291	8.65E-05	109.87.242.88	10398
292	8.56E-05	161.73.110.167	11782
293	8.55E-05	175.73.163.133	1300
294	8.54E-05	2.193.157.121	8880
295	8.53E-05	161.145.254.28	33058
296	8.52E-05	171.198.16.245	4685
297	8.50E-05	175.71.19.30	5128
298	8.50E-05	199.124.196.2	4846
299	8.49E-05	202.73.26.224	7613
300	8.48E-05	117.154.56.83	21089
301	8.48E-05	175.84.142.175	15285
302	8.48E-05	170.237.66.134	1439
303	8.47E-05	175.241.66.34	1720
304	8.47E-05	161.30.44.114	1718
305	8.47E-05	113.201.216.252	2861
306	8.47E-05	182.250.165.156	5811
307	8.47E-05	202.136.44.161	5876
308	8.47E-05	45.224.163.238	6419
309	8.47E-05	161.40.201.48	6449
310	8.47E-05	143.143.186.124	6479
311	8.47E-05	161.5.163.201	6831
312	8.47E-05	173.168.29.235	6855
313	8.47E-05	180.232.18.139	7407
314	8.47E-05	173.55.233.240	7808
315	8.47E-05	75.201.73.153	7929
316	8.47E-05	161.40.202.112	7962
317	8.47E-05	161.127.77.23	11616
318	8.47E-05	55.86.232.34	11808
319	8.47E-05	171.198.22.195	13847
320	8.47E-05	171.235.126.88	16884
321	8.47E-05	175.84.135.162	17599
322	8.47E-05	161.75.137.245	19038
323	8.47E-05	175.84.161.196	20821
324	8.47E-05	175.73.227.102	1063
325	8.45E-05	161.89.80.249	3296
326	8.44E-05	161.34.213.125	793
327	8.43E-05	158.48.141.222	1219

328	8.42E-05	161.80.15.114	4705
329	8.40E-05	180.237.55.118	10114
330	8.39E-05	13.49.59.120	2232
331	8.34E-05	112.98.126.124	1058
332	8.33E-05	161.80.46.254	10401
333	8.33E-05	182.250.166.246	653
334	8.24E-05	45.22.118.155	8217
335	8.23E-05	175.68.119.52	1054
336	8.18E-05	7.23.50.233	1441
337	8.16E-05	161.145.143.124	3068
338	8.15E-05	175.84.164.20	8774
339	8.14E-05	62.47.94.54	212
340	8.14E-05	129.197.65.92	93
341	8.09E-05	171.129.178.204	428
342	8.02E-05	161.131.83.182	4271
343	7.98E-05	113.201.203.53	12838
344	7.98E-05	113.201.229.73	2144
345	7.96E-05	175.71.106.137	12520
346	7.91E-05	175.84.154.106	17649
347	7.90E-05	171.176.21.97	20553
348	7.84E-05	175.68.140.189	13763
349	7.83E-05	171.222.163.13	23
350	7.83E-05	173.168.17.208	4784
351	7.80E-05	109.234.166.154	3504
352	7.79E-05	16.136.252.33	717
353	7.76E-05	200.7.28.69	4739
354	7.74E-05	136.10.171.94	3699
355	7.68E-05	175.71.43.0	28173
356	7.66E-05	132.252.133.80	2216
357	7.66E-05	175.84.133.171	8802
358	7.66E-05	146.206.122.3	14401
359	7.66E-05	175.84.137.128	17676
360	7.63E-05	113.201.153.63	2251
361	7.63E-05	42.27.129.231	3157
362	7.61E-05	52.103.137.30	10877
363	7.58E-05	47.52.147.0	3615
364	7.57E-05	16.136.249.248	5018
365	7.55E-05	161.5.248.122	37568
366	7.53E-05	55.83.255.223	12152
367	7.52E-05	161.89.82.3	2487
368	7.47E-05	23.23.119.230	3374
369	7.45E-05	69.104.35.55	974
370	7.41E-05	164.172.28.86	1321

371	7.40E-05	143.48.213.220	36574
372	7.40E-05	136.255.161.199	1643
373	7.38E-05	146.206.103.177	3368
374	7.37E-05	146.206.111.237	17333
375	7.35E-05	34.243.94.245	2693
376	7.26E-05	55.11.19.172	12133
377	7.24E-05	161.130.5.120	19
378	7.23E-05	203.80.10.188	693
379	7.22E-05	182.16.8.210	6382
380	7.20E-05	175.84.147.45	7183
381	7.19E-05	180.212.218.0	1109
382	7.16E-05	161.40.201.247	4804
383	7.14E-05	99.116.52.166	14252
384	7.13E-05	161.89.82.193	4455
385	7.11E-05	161.34.211.200	13843
386	7.11E-05	203.80.4.202	27115
387	7.07E-05	182.22.77.190	5150
388	7.07E-05	45.80.165.99	414
389	7.06E-05	175.84.138.66	2957
390	7.05E-05	161.34.216.199	11024
391	7.04E-05	13.57.233.243	3042
392	7.03E-05	175.73.177.63	6993
393	6.98E-05	175.84.150.15	4993
394	6.98E-05	203.253.164.234	1480
395	6.92E-05	175.240.12.168	5522
396	6.90E-05	201.222.50.139	1781
397	6.90E-05	161.104.68.0	12465
398	6.89E-05	146.206.111.221	8376
399	6.86E-05	161.146.133.214	18606
400	6.86E-05	173.136.103.220	3392
401	6.86E-05	202.209.124.14	23226
402	6.86E-05	175.84.157.180	14070
403	6.86E-05	146.206.166.122	9724
404	6.86E-05	203.244.242.151	1031
405	6.86E-05	63.47.55.31	2689
406	6.86E-05	34.235.250.164	2698
407	6.86E-05	72.106.58.237	2873
408	6.86E-05	170.238.213.157	6556
409	6.86E-05	171.191.230.89	8515
410	6.86E-05	175.69.188.176	8954
411	6.86E-05	61.158.234.149	9149
412	6.86E-05	69.35.149.97	9708
413	6.86E-05	203.67.62.219	10215

414	6.86E-05	161.89.80.210	10615
415	6.86E-05	131.175.72.28	10835
416	6.86E-05	191.151.24.211	10917
417	6.86E-05	167.104.87.176	10983
418	6.86E-05	203.38.37.209	11840
419	6.86E-05	136.185.4.113	12012
420	6.86E-05	113.201.169.87	12975
421	6.86E-05	143.143.105.247	17077
422	6.86E-05	49.36.203.110	17263
423	6.86E-05	182.22.238.22	18578
424	6.86E-05	131.14.105.30	18730
425	6.86E-05	203.140.147.225	18932
426	6.86E-05	113.201.220.69	20577
427	6.86E-05	180.212.217.82	26351
428	6.86E-05	117.154.143.252	26611
429	6.86E-05	113.201.139.154	26928
430	6.86E-05	13.240.42.98	27435
431	6.86E-05	175.84.142.53	28259
432	6.86E-05	173.250.209.222	28698
433	6.86E-05	180.90.99.80	30693
434	6.86E-05	161.185.239.148	2049
435	6.86E-05	52.117.186.251	13417
436	6.83E-05	170.251.76.66	1634
437	6.76E-05	42.27.129.217	3612
438	6.75E-05	161.34.167.147	71
439	6.70E-05	117.155.121.170	11035
440	6.69E-05	45.169.199.230	598
441	6.68E-05	162.190.51.56	7602
442	6.65E-05	164.37.46.149	2586
443	6.65E-05	173.168.29.4	3362
444	6.65E-05	175.69.101.89	12438
445	6.57E-05	175.68.171.136	5679
446	6.56E-05	182.250.35.130	6257
447	6.55E-05	161.31.199.1	213
448	6.54E-05	175.240.22.218	27
449	6.54E-05	175.68.242.114	9822
450	6.54E-05	175.240.29.77	10733
451	6.53E-05	164.88.118.203	10593
452	6.52E-05	195.192.101.48	14384
453	6.51E-05	175.73.97.129	5705
454	6.47E-05	146.206.106.93	272
455	6.45E-05	218.183.179.115	14481
456	6.44E-05	113.201.197.30	85

457	6.44E-05	113.201.142.213	12416
458	6.43E-05	173.136.103.61	270
459	6.41E-05	175.240.19.130	25
460	6.41E-05	113.201.152.212	2610
461	6.40E-05	182.127.153.108	30988
462	6.40E-05	113.201.199.100	12696
463	6.39E-05	25.111.44.39	1314
464	6.38E-05	203.24.248.14	5161
465	6.37E-05	203.38.38.172	21504
466	6.33E-05	146.206.112.136	374
467	6.32E-05	175.84.132.186	4836
468	6.32E-05	149.14.184.87	2870
469	6.31E-05	96.126.217.23	5902
470	6.31E-05	161.191.71.155	2482
471	6.25E-05	180.229.153.50	43
472	6.23E-05	203.253.166.231	1576
473	6.23E-05	161.146.157.139	10803
474	6.22E-05	171.232.87.24	30476
475	6.20E-05	13.243.152.14	1673
476	6.19E-05	171.198.4.59	6201
477	6.17E-05	203.29.249.106	45
478	6.16E-05	169.68.59.181	4008
479	6.14E-05	171.235.127.4	3481
480	6.10E-05	175.84.131.250	4713
481	6.09E-05	161.90.153.12	23140
482	6.06E-05	160.219.79.42	13996
483	6.05E-05	175.70.12.189	463
484	6.05E-05	201.67.248.4	6617
485	6.04E-05	113.201.220.239	2917
486	6.03E-05	175.84.137.29	5362
487	6.01E-05	45.125.27.124	11230
488	5.99E-05	175.84.161.145	12853
489	5.95E-05	96.163.14.177	1698
490	5.95E-05	162.188.201.69	10807
491	5.94E-05	203.253.166.182	1501
492	5.94E-05	174.110.86.7	4812
493	5.93E-05	161.75.142.20	20555
494	5.93E-05	113.201.155.91	19899
495	5.92E-05	161.30.44.209	2850
496	5.92E-05	136.28.84.105	36781
497	5.91E-05	175.73.60.97	18272
498	5.89E-05	45.235.120.227	1523
499	5.89E-05	175.68.62.142	9966

500	5.89E-05	69.35.151.186	7345
501	5.89E-05	171.128.144.2	2868
502	5.89E-05	175.240.2.221	22389
503	5.88E-05	190.26.240.64	11038
504	5.85E-05	170.199.82.70	4397
505	5.84E-05	52.205.189.139	19983
506	5.82E-05	161.195.225.237	32899
507	5.81E-05	16.136.248.125	4405
508	5.78E-05	171.191.238.130	768
509	5.77E-05	135.56.249.65	7721
510	5.77E-05	66.84.63.11	5934
511	5.76E-05	175.71.220.53	4875
512	5.74E-05	161.185.230.184	1518
513	5.74E-05	61.119.179.184	1077
514	5.73E-05	161.40.201.61	254
515	5.72E-05	87.88.34.35	10756
516	5.71E-05	175.241.104.182	21829
517	5.70E-05	161.31.223.65	19732
518	5.69E-05	99.116.52.241	9454
519	5.69E-05	175.84.153.206	3330
520	5.68E-05	180.222.92.161	30625
521	5.66E-05	55.46.187.192	12873
522	5.65E-05	187.92.12.45	12408
523	5.65E-05	182.250.165.252	4632
524	5.63E-05	161.185.234.208	27413
525	5.62E-05	175.84.128.58	17845
526	5.62E-05	7.19.133.217	2001
527	5.62E-05	161.34.217.226	4470
528	5.62E-05	161.80.26.237	22392
529	5.61E-05	203.253.165.9	1555
530	5.60E-05	118.187.245.138	4038
531	5.58E-05	175.84.160.58	1636
532	5.57E-05	175.84.135.20	22950
533	5.53E-05	175.240.28.219	19390
534	5.53E-05	149.46.170.58	33828
535	5.52E-05	175.241.123.112	820
536	5.49E-05	161.146.132.170	3535
537	5.48E-05	218.183.191.102	9317
538	5.47E-05	35.124.60.27	2593
539	5.47E-05	45.224.196.59	661
540	5.46E-05	45.81.254.25	10957
541	5.46E-05	113.201.210.185	1304
542	5.46E-05	161.54.117.88	5233

543	5.45E-05	128.42.138.47	10769
544	5.45E-05	171.168.34.114	4359
545	5.44E-05	202.209.119.35	299
546	5.44E-05	161.83.185.252	11143
547	5.41E-05	166.146.141.2	10471
548	5.40E-05	161.90.142.126	18266
549	5.40E-05	52.205.249.22	31966
550	5.39E-05	180.229.196.142	28652
551	5.38E-05	135.56.113.198	7805
552	5.37E-05	113.201.202.36	2332
553	5.37E-05	16.136.205.115	4307
554	5.36E-05	161.158.16.235	1185
555	5.36E-05	171.176.23.228	25459
556	5.35E-05	173.215.35.94	25885
557	5.35E-05	55.163.84.27	12274
558	5.33E-05	49.208.195.160	5021
559	5.32E-05	164.172.28.87	1517
560	5.31E-05	161.130.2.160	52
561	5.31E-05	171.168.39.71	19248
562	5.30E-05	55.64.116.216	12941
563	5.29E-05	162.160.50.76	4617
564	5.28E-05	175.84.137.21	1828
565	5.28E-05	170.251.76.67	1632
566	5.28E-05	170.250.216.79	5786
567	5.27E-05	201.244.25.195	24849
568	5.27E-05	113.201.128.184	2447
569	5.26E-05	161.61.74.224	10059
570	5.26E-05	161.72.151.220	25552
571	5.26E-05	36.90.130.103	15308
572	5.26E-05	182.244.58.120	22827
573	5.26E-05	161.191.210.66	3126
574	5.25E-05	182.250.165.179	168
575	5.25E-05	161.75.179.83	30897
576	5.25E-05	161.5.248.106	1179
577	5.25E-05	161.89.81.139	7846
578	5.25E-05	161.5.164.126	14729
579	5.24E-05	175.73.200.12	14037
580	5.24E-05	34.245.13.205	1563
581	5.24E-05	161.40.204.7	413
582	5.24E-05	216.2.118.156	1586
583	5.24E-05	113.201.191.21	1599
584	5.24E-05	72.106.40.247	1763
585	5.24E-05	107.180.44.227	2411

586	5.24E-05	177.125.235.165	2441
587	5.24E-05	175.84.167.200	2659
588	5.24E-05	52.0.228.197	2755
589	5.24E-05	161.124.191.125	2830
590	5.24E-05	13.210.11.245	2983
591	5.24E-05	55.44.135.96	3059
592	5.24E-05	175.71.232.115	3095
593	5.24E-05	161.40.202.196	3681
594	5.24E-05	175.241.57.253	4158
595	5.24E-05	173.214.39.23	4254
596	5.24E-05	187.92.13.82	5405
597	5.24E-05	161.5.163.96	5793
598	5.24E-05	146.206.172.94	6175
599	5.24E-05	180.232.112.188	7369
600	5.24E-05	203.99.189.60	7908
601	5.24E-05	175.84.159.19	8133
602	5.24E-05	90.35.152.217	8622
603	5.24E-05	173.164.225.39	8627
604	5.24E-05	131.251.72.3	8663
605	5.24E-05	42.27.130.95	8900
606	5.24E-05	161.145.142.21	9309
607	5.24E-05	175.84.167.69	9625
608	5.24E-05	175.84.145.215	9635
609	5.24E-05	46.215.222.34	10271
610	5.24E-05	175.73.36.30	10360
611	5.24E-05	161.55.85.18	10409
612	5.24E-05	171.234.248.31	10613
613	5.24E-05	91.36.232.224	10821
614	5.24E-05	45.23.21.158	10853
615	5.24E-05	162.103.190.77	10863
616	5.24E-05	136.9.223.183	11072
617	5.24E-05	161.40.187.175	11162
618	5.24E-05	219.109.93.205	11297
619	5.24E-05	175.71.245.238	11386
620	5.24E-05	170.196.26.83	11605
621	5.24E-05	175.240.17.57	11864
622	5.24E-05	175.84.163.110	12442
623	5.24E-05	34.196.221.19	12467
624	5.24E-05	161.190.11.41	12492
625	5.24E-05	113.201.205.125	12503
626	5.24E-05	173.215.42.11	12536
627	5.24E-05	214.148.89.178	12655
628	5.24E-05	175.84.133.146	12787

629	5.24E-05	7.25.109.140	13222
630	5.24E-05	52.214.145.151	13291
631	5.24E-05	118.65.53.55	13370
632	5.24E-05	113.201.188.222	13398
633	5.24E-05	45.224.196.81	15391
634	5.24E-05	171.168.47.197	15647
635	5.24E-05	143.143.146.27	15710
636	5.24E-05	171.168.20.1	16051
637	5.24E-05	175.84.164.43	16110
638	5.24E-05	201.83.205.135	17123
639	5.24E-05	175.84.131.219	17231
640	5.24E-05	180.222.199.101	17442
641	5.24E-05	126.17.208.101	17540
642	5.24E-05	143.143.60.21	18286
643	5.24E-05	117.182.166.135	18520
644	5.24E-05	170.250.24.1	18543
645	5.24E-05	184.255.249.29	18549
646	5.24E-05	162.67.227.37	18598
647	5.24E-05	171.176.30.63	19156
648	5.24E-05	37.62.229.216	19405
649	5.24E-05	175.73.39.27	19449
650	5.24E-05	131.14.107.10	19773
651	5.24E-05	182.19.217.199	20077
652	5.24E-05	13.56.54.165	20394
653	5.24E-05	175.84.151.129	21552
654	5.24E-05	184.253.82.66	23177
655	5.24E-05	161.145.137.185	23328
656	5.24E-05	175.73.103.108	23989
657	5.24E-05	162.190.71.241	23998
658	5.24E-05	135.56.93.41	24092
659	5.24E-05	173.141.36.211	25045
660	5.24E-05	136.127.111.6	25297
661	5.24E-05	136.186.102.248	25433
662	5.24E-05	161.190.163.201	25741
663	5.24E-05	170.199.127.169	25745
664	5.24E-05	161.146.184.245	26101
665	5.24E-05	218.247.229.8	26262
666	5.24E-05	171.176.28.126	26372
667	5.24E-05	161.158.54.205	26550
668	5.24E-05	161.31.202.230	26585
669	5.24E-05	177.125.235.199	26720
670	5.24E-05	146.7.148.109	26807
671	5.24E-05	203.254.27.22	27032

672	5.24E-05	113.201.157.177	27366
673	5.24E-05	136.34.92.147	27560
674	5.24E-05	169.26.85.236	27631
675	5.24E-05	195.192.215.89	27999
676	5.24E-05	135.56.148.187	28007
677	5.24E-05	171.198.93.159	29731
678	5.24E-05	174.236.10.2	30500
679	5.24E-05	173.129.186.137	30536
680	5.24E-05	46.22.0.92	30668
681	5.24E-05	171.220.0.205	30775
682	5.24E-05	175.73.89.182	31055
683	5.24E-05	103.96.207.254	31075
684	5.24E-05	45.21.227.167	31316
685	5.24E-05	173.136.99.93	31380
686	5.24E-05	25.64.12.24	31884
687	5.24E-05	55.160.190.4	32521
688	5.24E-05	175.84.144.230	35052
689	5.23E-05	60.77.138.58	20750
690	5.21E-05	175.73.223.31	10299
691	5.20E-05	171.198.21.254	7464
692	5.19E-05	158.48.218.137	10192
693	5.19E-05	161.40.201.161	22115
694	5.19E-05	57.57.222.246	377
695	5.17E-05	161.146.161.23	3244
696	5.17E-05	34.192.136.182	2749
697	5.15E-05	211.176.102.94	4090
698	5.14E-05	171.222.160.19	91
699	5.13E-05	202.209.114.107	3402
700	5.12E-05	52.115.245.102	12158
701	5.11E-05	170.238.221.203	5781
702	5.11E-05	49.36.197.222	8617
703	5.11E-05	175.84.148.239	7053
704	5.10E-05	143.110.26.70	25389
705	5.07E-05	175.68.147.234	14135
706	5.07E-05	161.131.81.57	7227
707	5.06E-05	201.240.78.94	351
708	5.05E-05	180.211.115.111	2911
709	5.04E-05	203.38.44.111	1423
710	5.04E-05	175.73.29.76	30862
711	5.04E-05	203.253.164.217	1363
712	5.03E-05	171.198.3.99	3253
713	5.01E-05	192.175.233.74	34275
714	5.01E-05	177.125.198.100	2306

715	5.01E-05	190.202.169.114	5694
716	5.00E-05	223.231.48.108	21638
717	5.00E-05	180.1.127.184	143
718	4.99E-05	175.84.151.3	6091
719	4.98E-05	182.237.78.108	59
720	4.98E-05	128.222.2.57	3988
721	4.97E-05	175.84.159.205	7153
722	4.97E-05	170.238.219.205	13784
723	4.95E-05	171.168.1.86	9524
724	4.95E-05	98.110.231.226	18987
725	4.92E-05	175.70.123.230	18135
726	4.92E-05	171.222.166.82	18985
727	4.90E-05	161.5.163.58	4526
728	4.90E-05	223.249.45.189	1525
729	4.89E-05	180.1.127.205	35469
730	4.88E-05	175.70.199.213	23424
731	4.88E-05	16.136.226.93	4057
732	4.87E-05	157.62.163.30	5797
733	4.86E-05	173.31.236.23	21839
734	4.84E-05	173.196.45.98	11160
735	4.84E-05	203.94.159.252	13737
736	4.83E-05	182.247.131.151	14454
737	4.83E-05	52.118.34.253	25374
738	4.81E-05	175.241.82.247	10197
739	4.81E-05	61.119.179.174	10495
740	4.79E-05	117.164.136.187	18260
741	4.79E-05	140.152.217.245	1960
742	4.78E-05	175.240.27.138	25977
743	4.78E-05	175.84.146.248	21572
744	4.78E-05	136.185.84.254	18264
745	4.76E-05	175.84.162.224	6517
746	4.76E-05	113.201.170.5	2109
747	4.75E-05	161.190.175.22	799
748	4.73E-05	161.185.239.150	2783
749	4.73E-05	182.250.167.52	3716
750	4.72E-05	210.200.42.157	9460
751	4.71E-05	173.141.3.146	1333
752	4.71E-05	161.145.137.5	7398
753	4.70E-05	175.84.141.219	10089
754	4.69E-05	146.206.114.34	698
755	4.69E-05	161.53.63.135	4683
756	4.68E-05	204.227.68.11	10130
757	4.67E-05	162.173.237.136	2613

758	4.67E-05	175.84.160.18	21332
759	4.65E-05	113.201.224.197	13272
760	4.65E-05	171.168.7.220	5268
761	4.65E-05	161.89.82.111	14262
762	4.64E-05	175.68.237.97	25019
763	4.64E-05	126.40.66.145	1256
764	4.64E-05	170.231.102.97	5828
765	4.63E-05	170.238.217.194	583
766	4.63E-05	175.84.147.240	6210
767	4.63E-05	175.84.156.138	2341
768	4.61E-05	182.237.79.201	31
769	4.60E-05	105.18.187.26	2421
770	4.60E-05	45.226.119.98	21494
771	4.60E-05	108.41.214.156	15783
772	4.59E-05	161.55.84.198	2660
773	4.57E-05	69.35.149.123	5647
774	4.57E-05	175.84.161.160	12595
775	4.57E-05	45.172.88.122	8232
776	4.56E-05	175.84.156.67	5091
777	4.54E-05	171.176.19.188	3668
778	4.54E-05	175.241.13.208	23511
779	4.53E-05	45.226.7.39	6933
780	4.53E-05	161.159.173.4	1257
781	4.53E-05	175.68.209.27	4290
782	4.53E-05	175.84.162.165	20018
783	4.52E-05	61.22.69.19	17362
784	4.51E-05	175.84.141.24	9510
785	4.51E-05	161.89.86.8	8471
786	4.51E-05	223.231.106.161	7027
787	4.51E-05	170.199.73.62	953
788	4.50E-05	161.30.54.219	35
789	4.50E-05	162.190.19.20	4285
790	4.50E-05	45.23.226.189	36761
791	4.50E-05	211.143.185.136	3192
792	4.49E-05	113.201.142.240	26645
793	4.49E-05	171.176.24.46	11239
794	4.48E-05	45.175.66.2	14138
795	4.48E-05	161.30.45.207	77
796	4.48E-05	182.17.80.76	4923
797	4.47E-05	136.199.114.44	9763
798	4.47E-05	80.8.30.102	33987
799	4.47E-05	161.40.203.205	3945
800	4.47E-05	161.31.214.144	19997

801	4.45E-05	162.98.225.154	10940
802	4.44E-05	161.130.10.14	12818
803	4.44E-05	170.238.254.203	10536
804	4.44E-05	113.201.149.231	12092
805	4.44E-05	161.53.131.127	22504
806	4.44E-05	161.145.143.3	25728
807	4.44E-05	41.149.85.159	26497
808	4.44E-05	171.235.116.20	27324
809	4.44E-05	161.53.10.229	28493
810	4.44E-05	177.125.214.167	32370
811	4.42E-05	212.195.73.104	16603
812	4.42E-05	162.161.21.25	1157
813	4.41E-05	113.201.226.181	12563
814	4.40E-05	175.84.164.76	6603
815	4.39E-05	161.145.233.52	15814
816	4.39E-05	175.84.141.246	6819
817	4.39E-05	175.73.215.240	18783
818	4.39E-05	113.201.207.39	37570
819	4.38E-05	175.241.75.89	275
820	4.38E-05	175.84.147.222	8718
821	4.38E-05	161.144.97.216	31419
822	4.38E-05	175.68.92.223	21416
823	4.38E-05	161.34.214.24	114
824	4.38E-05	45.62.144.11	11716
825	4.36E-05	211.176.97.222	8389
826	4.36E-05	42.27.131.111	267
827	4.36E-05	166.194.196.29	10438
828	4.36E-05	16.110.190.135	28990
829	4.35E-05	173.196.126.143	4436
830	4.35E-05	128.72.211.35	30736
831	4.32E-05	202.209.114.244	18087
832	4.32E-05	161.146.191.131	34047
833	4.32E-05	161.80.58.84	4510
834	4.32E-05	55.44.128.232	1409
835	4.32E-05	116.100.240.235	10266
836	4.32E-05	202.148.107.213	25815
837	4.32E-05	223.231.21.33	6156
838	4.31E-05	187.92.14.27	1928
839	4.31E-05	69.35.147.85	8229
840	4.31E-05	201.92.22.86	11885
841	4.31E-05	180.211.1.240	13839
842	4.31E-05	170.238.72.21	3554
843	4.31E-05	173.141.2.155	1820

844	4.30E-05	25.74.174.165	12602
845	4.30E-05	162.67.181.61	14552
846	4.30E-05	37.232.225.142	12017
847	4.30E-05	161.40.202.34	33963
848	4.28E-05	175.84.140.113	2293
849	4.27E-05	175.71.1.163	8938
850	4.26E-05	55.63.179.121	13000
851	4.25E-05	175.241.100.25	297
852	4.25E-05	52.201.47.29	100
853	4.25E-05	71.217.221.229	1080
854	4.24E-05	35.79.178.51	20041
855	4.24E-05	69.35.155.124	7179
856	4.23E-05	170.227.13.31	795
857	4.23E-05	52.218.22.236	12365
858	4.23E-05	175.84.154.86	33722
859	4.23E-05	175.84.132.87	17657
860	4.23E-05	36.59.215.145	22669
861	4.23E-05	161.39.15.102	10784
862	4.22E-05	202.209.116.72	8932
863	4.22E-05	223.166.150.244	11212
864	4.21E-05	146.7.147.205	1386
865	4.21E-05	49.36.159.243	8398
866	4.21E-05	175.84.128.99	3343
867	4.20E-05	202.209.125.142	815
868	4.19E-05	45.170.193.77	37258
869	4.19E-05	161.72.1.141	24210
870	4.19E-05	201.244.17.53	6350
871	4.18E-05	170.238.72.108	36394
872	4.17E-05	202.118.35.6	20977
873	4.17E-05	42.27.128.77	20987
874	4.17E-05	162.103.88.227	25248
875	4.16E-05	118.66.185.104	13346
876	4.16E-05	45.172.78.83	11223
877	4.15E-05	173.215.39.205	2611
878	4.14E-05	175.84.132.245	1604
879	4.14E-05	175.84.155.156	13874
880	4.14E-05	175.240.10.56	6399
881	4.13E-05	37.190.21.119	5242
882	4.12E-05	201.194.10.94	8673
883	4.12E-05	136.127.65.149	36822
884	4.11E-05	162.160.153.28	14688
885	4.11E-05	180.229.155.131	13813
886	4.11E-05	171.235.120.182	21310

887	4.11E-05	177.125.214.27	2145
888	4.10E-05	175.84.153.151	880
889	4.10E-05	161.5.163.232	4136
890	4.10E-05	175.70.105.76	23896
891	4.10E-05	161.146.134.191	32584
892	4.09E-05	161.83.182.175	12518
893	4.09E-05	113.201.215.235	31620
894	4.09E-05	175.84.140.10	4762
895	4.09E-05	177.160.239.227	10274
896	4.09E-05	161.75.133.153	554
897	4.08E-05	179.163.203.99	331
898	4.08E-05	143.45.162.123	19029
899	4.07E-05	35.5.69.9	18246
900	4.06E-05	161.185.231.254	10909
901	4.06E-05	220.178.3.149	18486
902	4.06E-05	187.92.213.3	6064
903	4.06E-05	161.146.140.150	18524
904	4.06E-05	175.69.130.10	33442
905	4.05E-05	161.40.202.205	3394
906	4.05E-05	16.136.247.143	538
907	4.05E-05	161.80.16.160	3302
908	4.04E-05	175.71.211.9	9945
909	4.04E-05	113.201.219.212	1786
910	4.03E-05	136.68.156.29	1319
911	4.03E-05	113.201.201.246	1722
912	4.03E-05	161.40.202.191	9152
913	4.03E-05	161.83.253.244	20780
914	4.03E-05	161.40.204.24	24794
915	4.03E-05	175.84.155.225	30752
916	4.03E-05	175.84.140.88	7756
917	4.03E-05	135.56.61.236	21841
918	4.03E-05	162.173.232.53	24418
919	4.02E-05	175.73.57.187	11792
920	4.01E-05	182.236.93.238	4252
921	4.01E-05	131.14.10.51	5953
922	4.01E-05	80.9.254.161	5178
923	4.01E-05	161.53.10.162	30074
924	4.00E-05	13.51.248.22	12325
925	4.00E-05	170.227.4.52	9732
926	4.00E-05	37.27.86.234	8808
927	4.00E-05	175.71.182.14	23849
928	4.00E-05	171.128.156.245	7143
929	4.00E-05	167.104.87.168	1068

930	3.99E-05	175.84.134.115	28467
931	3.98E-05	55.36.78.80	31695
932	3.98E-05	170.238.203.236	5863
933	3.98E-05	195.89.24.101	12399
934	3.97E-05	162.102.118.153	22624
935	3.97E-05	175.241.116.116	3464
936	3.96E-05	113.201.216.125	2321
937	3.96E-05	161.145.232.158	10950
938	3.95E-05	118.65.59.110	20234
939	3.95E-05	136.7.228.97	9025
940	3.95E-05	180.211.115.29	14771
941	3.95E-05	162.198.28.34	670
942	3.95E-05	161.5.150.26	8221
943	3.94E-05	175.84.132.203	3640
944	3.94E-05	171.199.181.86	37196
945	3.94E-05	200.28.218.248	29165
946	3.93E-05	117.154.160.44	25963
947	3.93E-05	175.84.136.10	13854
948	3.93E-05	113.201.196.152	20390
949	3.93E-05	161.80.4.220	8330
950	3.93E-05	175.84.152.254	8681
951	3.92E-05	161.40.202.119	8846
952	3.92E-05	173.196.77.144	4628
953	3.92E-05	170.230.42.53	34975
954	3.92E-05	143.48.177.27	26051
955	3.91E-05	113.201.151.195	1305
956	3.91E-05	161.61.127.96	3439
957	3.91E-05	13.219.210.141	13145
958	3.90E-05	173.215.40.67	1826
959	3.90E-05	179.160.58.204	10279
960	3.90E-05	171.168.22.161	16136
961	3.90E-05	158.48.141.156	1056
962	3.90E-05	173.250.215.201	2804
963	3.90E-05	75.201.73.155	7858
964	3.90E-05	182.237.76.144	21813
965	3.90E-05	113.201.192.116	31645
966	3.89E-05	175.84.156.201	18000
967	3.89E-05	161.5.161.34	5737
968	3.88E-05	105.229.64.61	11029
969	3.88E-05	203.253.164.255	1378
970	3.88E-05	131.96.193.7	54
971	3.88E-05	171.198.20.108	30884
972	3.88E-05	170.227.3.236	32645

973	3.88E-05	161.130.5.34	4546
974	3.88E-05	173.219.141.10	5504
975	3.88E-05	113.201.191.7	12238
976	3.88E-05	194.191.86.134	21603
977	3.87E-05	45.62.83.102	2766
978	3.86E-05	171.235.120.255	5541
979	3.86E-05	175.70.191.45	10837
980	3.86E-05	175.251.49.114	16993
981	3.86E-05	175.84.155.232	611
982	3.86E-05	135.56.51.68	5385
983	3.86E-05	131.96.29.66	23086
984	3.86E-05	170.230.40.14	3735
985	3.85E-05	161.80.34.23	7797
986	3.85E-05	175.84.151.186	5363
987	3.85E-05	171.220.86.170	28879
988	3.85E-05	42.27.130.163	21549
989	3.84E-05	143.45.227.111	10501
990	3.84E-05	162.191.104.189	30887
991	3.84E-05	173.215.42.10	32006
992	3.84E-05	170.199.101.4	475
993	3.84E-05	99.11.157.100	28244
994	3.84E-05	211.176.102.228	6864
995	3.84E-05	113.201.208.82	20211
996	3.84E-05	180.211.115.183	13811
997	3.83E-05	202.118.33.252	18640
998	3.83E-05	175.84.163.201	545
999	3.83E-05	175.84.145.217	23339
1000	3.83E-05	161.144.232.64	1664