Addis Ababa Institute of Technology
School of Electrical and Computer Engineering
Telecommunication Engineering Graduate Program

Multi-Class DNS Attacks Classification using Deep Learning

By: Teshome Assefa

Supervised by: Fistum Assamnew (Ph.D.)

A thesis submitted to Addis Ababa University, Addis Ababa Institute of Technology School of Electrical and Computer Engineering, in partial fulfillment of the requirements for the degree of Master of Science in Telecommunications Engineering (Telecommunication Information system truck).

January 2021

Addis Ababa, Ethiopia

Declaration

I, Teshome Assefa, approve that the work offered in this thesis is my work and has not been offered for a degree in any other university previously. I have fully acknowledged all the sources of information, which have been used in the thesis.

Teshome Assefa

January 2021

Addis Ababa Institute of Technology School of Electrical and Computer Engineering Telecommunication Engineering Graduate Program

This is to certify that, the thesis prepared by Teshome Assefa entitled, Multi-class DNS Attack classification using Deep Learning and submitted in partial fulfillment of the requirements for the degree of Master of Science Telecommunication Engineering complies with the regulations of the University and meets the accepted standards concerning originality and quality.

Signed by the Examining Committee:

Internal Examiner

_____Signature_____Date_____

External Examiner

_____Signature_____Date_____

Adviser

Fitsum Assamnew (Ph. D) Signature_____Date_____

Co-Adviser

_____Signature_____Date_____

_____

Dean, School of Electrical and Computer Engineering

Abstract

Nowadays, the number of Internet customers and data usage are increasing rapidly in Ethiopia and worldwide. One of the main components for providing internet services for customers is Domain Name System. It translates a domain name to IP address. It improves by introducing innovative architecture, interfaces, and protocols. It aids customers to simplify their services using these platforms. However, these have also opened new vulnerabilities on DNS. Thus it becomes the target of attackers. The attacker takes the advantage of openness to attack DNS such as DOS/DDOS, cache poisoning, Tunneling, Domain generating algorithms attack, and others.

There are many implementations to detect DNS attacks. Recently, deep learning has emerged as an effective method for multi-class DNS attack detection and classification. We found that RNN classifiers improve DNS attack classification. They are good at dealing with sequential information. These classifiers' performances were evaluated by calculating mean test accuracy, AUC-ROC, f1-score, and confusion matrix. We compared the performance of four different supervised deep learning classifiers, LSTM, GRU, BiGRU, and BiLSTM, on five-class DNS attacks. We selected the BIGRU model. The value of test accuracy and AUC of it are 97.18% and 0.9970 respectively.

The performance of per class classifications by Ensemble model which we reviewed has been built less than BIGRU to classify ramnit and qakbot classes. F1-score of BIGRU to classify qakbot is greater than Ensemble model by 0.1666. And the same way, the F1-score of BIGRU to classify ramnit classes is also greater than the Ensemble model by 0.0798.

Keywords—Domain Name System, deep learning, RNN, DOS/DDOS, cache poisoning, DNS Tunneling, DNS Amplification, ramnit, qakbot

Acknowledgment

Primarily, I would like to thank the Almighty God for guiding me in every aspect while doing this study. Next, I would like to express my appreciation to our adviser Dr. Fistum Assamnew for his appreciated support and wise comments.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Abbreviation	Description
ACC	Accuracy
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under Curve
BILSTM	Bidirectional Long Short-Term Memory
C&C server	Command and Control Server
CNN	Convolutional Neural Network
DAAD	DNS Amplification Attacks Detector
DDoS	Distributed Denial-of-Service
DGA	Domain Generating Algorithms
DNN	Deep Neural Network
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DoS	Denial-of-service
FN	False Negative
FP	False Positive
GRU	Gated recurrent units
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPS	Intrusion Prevention Systems
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NXDOMAIN	Non-eXistent Domain
PR	Precision
PRNG	pseudo-random generator
RE	Recall
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks

Abbreviation	Description
ROC	Receiver Operating Characteristic
SD	Standard Deviation
SQL	Structured Query Language
SVM	Support Vector Machine
TLD	Top-Level Domain
TN	True Negative
TP	True Positive
TSIG	Transaction Signature
UDP	User Datagram Protocol
VPN	Virtual Private Network
XSS	Cross-Site Scripting

CHAPTER ONE

1. Introduction

DNS fundamentally contains three layers in its structure; namely, access layer, service layer, and resource layer. The organization of those layers supports simplifying management on each layer, detecting problems, and making a solution. The access layer is a layer that contains DNS Management, Domain Name Management, and WHOIS Web end. It provides Web Fronted for users to purchase a domain name, manage domain name, and subdomain name. The service layer is the core business processing layer that supports Shared Registry System which provides domain name registration service to the registrar, including creating, modifying, and deleting the domain name, contact person, and host. The resource layer is used to store the domain registry information such as domain name info, contact info, and other registry information.

The Domain Name System is a vital part of the internet since it supports several critical applications such as the Web, FTP, Email, and other applications that require remote access to servers using the Internet Protocol [12]. The main purpose of DNS is to provide forward and backward resolution of host/domain names into IP addresses [1, 12]. The quality of internet services depends on both responsiveness and correctness of translation [1].

Nowadays, increasing mobile phones that are accessing the Internet worldwide are the main contributors to the expansion of the internet. Without DNS most people cannot connect to

their favorite website. Attackers have targeted DNS to attack. These attacks are likely because of exploiting the DNS protocol. Some of the attacks are Denial of Service (DOS) attack, cache poisoning attack, tunneling attack, DNS Amplification, and others.

Furthermore, different types of attacks are usually processed in different ways. How to detect various types of DNS attacks become the main challenge in the field of DNS security. The conventional approaches used by many for DNS attack detection include the DNSSEC provides more productivity to solve the DNS cache poisoning problem [4], perform payload and traffic analysis to identify DNS Tunneling [5], and others. In this work, we recommend a method to detect and classify multi-class DNS attacks using deep learning.

1.1 Statement of the Problem

Currently, DNS service requests are increasing rapidly. At the same time, numbers and types of DNS attacks are also increasing exponentially. This brings challenges to telecom operators, like ethiotelecom, that attackers have focused on this platform to exploit for different purposes.

Now, ethiotelecom detects DNS attacks based on ruled base access control mechanisms and others. Thus, these preventing mechanisms are not dynamic, high false-positive rate, and challenge to detect new attacks. It also takes time to set policies, challenges to manage patches and updates.

In recent times, deep learning has emerged as an effective approach to detecting DNS attacks as a replacement for of traditional way because it has more ability to detect new attacks. It also reduces the false-positive rate, detecting mechanism becomes dynamic and effective to secure the DNS by increasing the classification accuracy.

Many kinds of research have been conducted aiming at choosing the effective deep learning classifier that accomplishes the accepted performances to identify different types of attacks. Their results, method, and nature of the dataset used for research differ from each other. These show that there is no single deep learning algorithm that can handle efficiently all the kinds of attacks [3].

In this research, we will emphasize developing four supervised deep learning models to classify Multi-class DNS attacks. Finally, we compare them and find the best classifier using mean accuracy, F1-score, and AUC metrics.

1.2 Objectives

1.2.1 General Objectives

- The main objective of this thesis is to build four deep learning models that classify multi-class DNS attacks. Compare and select the best model based on their performance. Show our model has better accuracy than other models which we reviewed have been built.

1.2.2 Specific objectives

- To show that DNS attacks could also be initiated from attackers that might cause serious problems on the performance of DNS resolution.
- Study the common and distinct features of different classes.
- Harmonizing the different datasets to be used in classification.
- Building the multi-class deep learning DNS attack detection models.
- Evaluating performance of build models and select the model that has the highest performance
- Compare our best model with other models which we reviewed have been built

- And then make a recommendation and conclusion.

1.3 Methodology

While conducting this research, the following methodology was used:

- To get a detailed idea about the four DNS attacks types, different works of literature related to DNS security have been reviewed.
- All types of data have been collected from global datasets.
- After data preprocessing are performed on the collected dataset, four deep learning algorithm (LSTM, BiLSTM, GRU, and BiGRU) were used to train and evaluate their performance using classification accuracy, F-Measure, AUC, and confusion matrix as performance evaluation metrics.

1.4 Scope and Limitation

1.4.1 Scope of the Study

The scope of this work is restricted or limited to four DNS attacks types and four types' deep learning classification algorithms. The required data was gathered from global dataset sources.

1.4.2 Limitation of the Study

The following list describes the limitations faced while conducting this study.

- The main challenge was getting accurate information on the existing status of ethiotelecom DNS attacks from a security viewpoint.

- Another obstacle was a willingness to deliver the important information as it was related to security and confidentiality issues.
- There are few numbers of research are found related to DNS attacks classification using deep learning.
- There was also a shortage of global DNS attacks datasets.

1.5 Contribution of the Study

Besides the practical applicability for ethiotelecom, the research is expected to provide an awareness to the ethiotelecom security section and related division staffs and managers to increase their knowledge about the dynamism of DNS security issues that can lead to degraded internet related services due to mischievous activities on DNS. It reveals gaps in the current security system and alleviates these security issues. It also shows deep-learning techniques efficient to detect DNS attacks. This reduces false positive and False Negative rates, and other side it increases the true positive and true negative rates.

1.6 Literature Review

Some authors and researchers have put forward their ideas on an area of the number of attacks that have been carried out on DNS. Some are DOS/DDOS, cache poisoning, DNS Amplification, tunneling attack, and others. To detect or protect those attacks, the researchers proposed different solutions like DNSSEC, TSIG, DNS Firewall, machine learning, deep learning, and others [2] that are published in IEEE articles, books, journals, and researchers' work. In this section, some research papers that are related to this thesis are reviewed here as follow:

DNS is a complex distributed database on which most internet services rely on [1]. Since monitoring this traffic is critical, it is important to identify attacks, measure performance, and generate usage statistics. This aids to advance the DNS infrastructure and inform

detected issues to domain registrars to deliver a quality of internet and data services to customers [1].

Over the years, the DNS attackers have accomplished their attacking activities on DNS because DNS is an important part of internet infrastructure and it has several security gaps or is vulnerable to be attacked by attackers. There are various ways in which DNS can be affected. Some attacks are cache poisoning, DNS reflection/Amplification attack, DOS/DDOS, and others.

DNS amplification attack exploits vulnerabilities of DNS to consume target or victim resources faster. It uses the reflection method, where an attacker sends a DNS query with the deceiving IP address of the victim to a recursive DNS server [2]. Recursive DNS server replies large size responses to target victim and consumes its resources.

DNS Cache Poisoning attack is entering wrong information into the DNS cache, which makes the DNS server reply the attacker desires other than the correct reply [2]. Then customers are directed to an incorrect website. To protect against the attack, DNSSEC has been introduced as a more secure alternative, which implements a more advanced technique called cryptography. DNSSEC provides a more fruitful result to DNS cache poisoning attacks. This protects the DNS cache data forgery attack. It uses a public-key cryptographic signature, to verify and authenticate the data [4]. However, it has not been widely adopted yet.

Another one is the DNS tunneling attack that is often used for malicious activities. It is used for stealing information in a network, called data exfiltration. To accomplish this, an attacker first infects a DNS client with malware that creates a tunnel to the attacker's machine through a recursive DNS server located within the user's network, thus bypassing a firewall of the network [5, 8]. To classify the DNS tunneling, they used the machine learning method, a multilabel support vector machine. When detecting a normal

DNS connection, kernel SVM has 100% classification accuracy [3]. Kernel SVM has overtaken Bayesian classifier.

DOS/DDOS attacks are possible against any DNS server which is open for queries. In a DOS attack, attackers seek to make DNS server inaccessible to intended users or flood target DNS server by sending huge quantity of packets to it. DDOS attack aims to crash DNS by overwhelming it with a flood of UDP requests. The UDP requests are generated through a network of jeopardized systems that may be part of Botnet [2]. They typically aim to destroy DNS resources and network bandwidth, ranging from Network layer to Application layer [6].

The proposed DDOS detection is a deep learning approach is called DeepDefense. The DeepDefense approach identifies DDOS attacks using Recurrent Neural Network (RNN) (e.g., LSTM, GRU) [6]. The deep learning approach is suitable for large-scale network traffic and can rapidly extract high-level features from low-level ones and achieve powerful representation and interpretation. In a large dataset, we can reduce the error rate. This shows its capability of learning from historical network packets [6]. The detection approach uses a sequence of continuous network packets and learns subtle differences between attack traffic and legitimate ones. Historical information is fed into RNN models (LSTM, CNNLSTM, GRU, and 3LSTM) to identify DDoS attacks. It helps to find repetitive patterns representing DDoS attacks and locate them in long-term traffic sequences. The outcome is LSTM model shows a slightly higher accuracy (97.996%), recall (97.887%) score, and less error rate (2.004%) in Data14 [6]. 3LSTM is the best model for Data15 which is larger than Data14. However, LSTM and GRU are very close to 3LSTM [6]. The LSTM neural network can detect DDOS because LSTM networks are capable of solving the vanishing gradients problem in RNN and use a memory cell to offer the previous information. We notice that the growth of window size, performance of the

LSTM model develops [6]. LSTM with a window size of 100 attains the lowest error rate. This shows the ability of Recurrent Neural networks in long-term sequences.

We have reviewed the literature, some researchers have provided an IDS which is using signature-based, anomaly-based, and hybrid-based mechanisms to detect various DNS attacks. However, according to [16] such a simple and static access control mechanism could wrongly classify normal as an attack. Deep learning techniques have the hopeful capability in such regard by avoiding the rigidity of traditional rule and programmed tools by adapting their behavior using their inputs, so they can detect novel attacks or increase generalizability.

Deep Learning uses Supervised learning (e.g. DNN, CNN, and RNN) methods to identify attacks with high accuracy, due to the amount of data provided by manually labeled instances [9].

Since the output of CNN and DNN only considers the impact of present input without seeing information about future and previous time, they could attain substantial performance on classification tasks without time-varying characteristics [9]. Relating time-dependent data, RNN is suggested as a special type of neural network structure, which is designed with "memory" function to maintain previous content. Thus RNN is good at dealing with time-series information [9]. LSTM is a special type of RNN that solves the long-term dependency problem and overcomes the vanishing gradient drop during training, Kim et al. apply LSTM architecture for intrusion detection and classification [9, 7].

There are DNS attacks classification methods. Two of them are Binary classification and Multi-class classification. Binary classification denotes classification tasks that have two class labels, whereas Multi-class classification takes more than two class labels.

Finally, there are four key measurements to calculate performance of binary and multi-class classifiers. They are accuracy (ACC), precision (PR), recall (RE), and F1-score. We define measurements as follows:

ACC = (TN +TP) / (FN + TN + TP + FP),

PR = TP / (FP +TP),

RE = TP / (FN +TP) and

F1-score = (2 ∗ RE* PR) / (RE +PR).

1.7 Organization of Thesis Document

This thesis comprises six chapters and each chapter has been structured as follows:

- The first chapter provides the introduction part, which includes: statement of the problem, general and specific objectives of the study, scopes, and limitations of the research, methodology, contribution of the study, literature review which discuss the main points learned from each literature and at last, how the thesis is organized.
- The next chapter deals with the background regarding DNS and its security. It states four DNS attacks types that are DOS/DDOS, Cache Poisoning, tunneling, and domain generating algorithms (DGA) attacks. It also tries to discuss mitigation mechanisms for those attacks in DNS.
- The third chapter discusses deep learning techniques. It also states deep learning models, supervise and unsupervised deep learning approaches relate to multi-class DNS attacks classification.
- Chapter four points out all findings from the experimental analysis used in this research. It starts from the setup used for dataset collection, data preprocessing,

training supervised algorithms, and finally evaluate their performances using performance metrics.

- Chapter five explains the main outcomes from the experimental analysis and performance of the four supervised deep learning algorithms to identify DNS attacks.
- Finally, chapter six contains recommendations and conclusions on the outcomes from this research. It also points out some future works.

CHAPTER TWO

2. DNS and its Vulnerability to different types of attacks

2.1 Overview of the DNS

The process of DNS resolution includes translating a domain name (such as www. google. com) into an IP address (such as 8.8.8.8). An IP address is set to each host on the internet, and that address is required to find an appropriate internet host. When a user wants to load a webpage, a translation has to occur between the domain name (google.com) and the machine-friendly IP address that is required to find the google.com webpage.

2.2 DNS Architecture

2.2.1 Recursive DNS resolver

It is a DNS server that answers a request from a client and finds out the DNS record. A succession of requests has been made until it gets the authoritative DNS name server. If the record is found, it returns the address of the requested domain name, otherwise, it returns times out or an error. Fortunately, recursive DNS resolvers do not repeatedly need to make numerous requests to locate the records that are needed to return to a client; caching is a short-term memory to easy retrieval that helps to find quickly the requested record earlier DNS lookup.

2.2.2 Authoritative DNS server

An authoritative DNS server contains DNS resource records. It is the server at the end of the DNS lookup series that will reply queried resource records. The web browser request to get the actual IP address of a website.

2.2.3 DNS lookup

As mentioned before, the core objective of DNS is to translate a domain name into the correct IP address. To show how this method works, it benefits to follow the DNS lookup path from a web browser, through the DNS lookup process, and return as shown in Fig 2-1 below.

DNS lookup data will be cached inside the Recursive DNS resolver. When DNS data is cached, steps are avoided from the DNS lookup process that makes it faster. The sample below summarizes the resolution of domain names that are not cached.

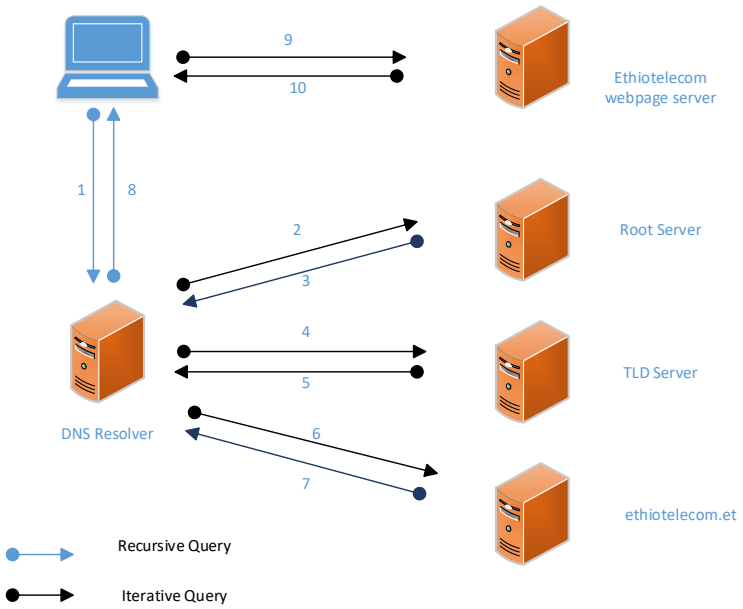


Fig 2-1. Complete DNS resolution and webpage query

- **Root name server** - It is the initial step in translating a domain name into an IP address.
- **TLD name server** – It is a top-level domain server (TLD) that is found after root servers.
- **Authoritative name server** - It is the last node in the domain name resolution process. If it has the requested domain name, it will reply IP address of the domain name to the DNS resolver.

2.2.4 DNS Queries

Normally, DNS lookup includes two kinds of queries.

1. **Iterative query** - the DNS client requests name resolution. If the asked DNS server has not requested a domain name, it will reply with a referral to a lower level authoritative DNS server. This method continues with other DNS servers until either timeout occurs or an error.
2. **Recursive query** - In a recursive query, a DNS client needs that a DNS recursive resolver to answer the request on behalf of the DNS client.

2.2.5 DNS caching

Cache stores data temporarily in a suitable place to improve the reliability and performance of the resolution process. DNS caching comprises storing data nearer to the DNS client for the resolution process to become swiftly, thus reducing load times and decreasing resource consumption. DNS information can be stored for a particular period of time or time-to-live (TTL).

2.3 Type of DNS attacks

There are different kinds of DNS attacks, some of them are as follow

2.3.1 DOS Flood Attacks

DNS flood attack occurs when attackers make enormous number of DNS requests to DNS servers. DNS servers like other Internet infrastructure are likely to be attacked by DoS Flood attacks. DoS attacks are also challenging to prevent because DNS uses UDP requests for domain name resolution. When several machines send a flood of swift DNS queries to the DNS server, the server has to handle more traffic than it can [18]. Resources of DNS servers can be consumed by a flood of DNS requests. So the response time for honest user become slower.

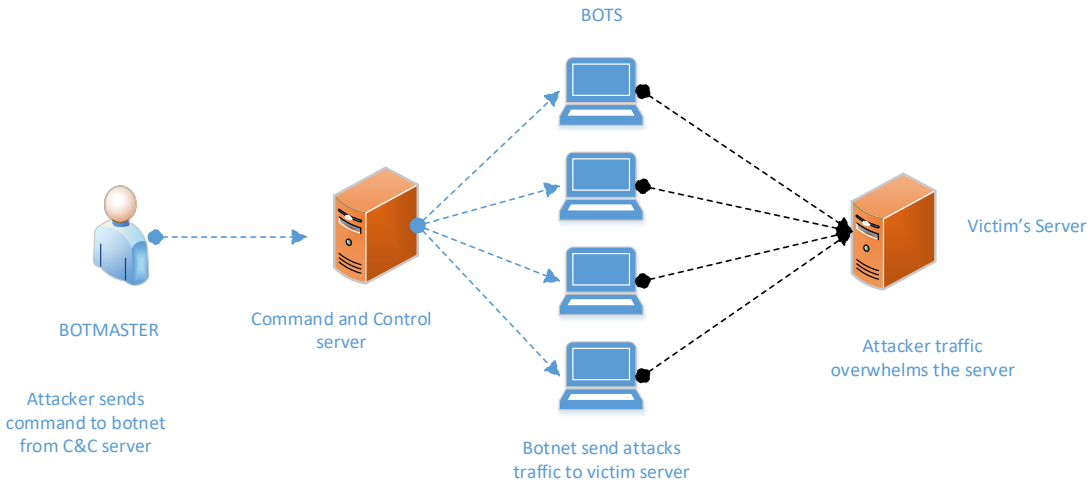


Fig 2-2. DOS flood attack

Fig 2-2 shows that the attacker sends commands to a botnet from the command control server to numerous infected hosts or bots which also send DNS queries to the targeted DNS servers. The attack's traffic overwhelms the DNS server and making it unavailable or unable to respond to legitimate users.

2.3.2 DNS Cache Poisoning

DNS cache poisoning is an injection of wrong information into a DNS cache, in order to DNS reply an incorrect one, and users are directed to the incorrect websites instead of the correct one as shown in Fig 2-3.

With UDP, the DNS accepts fake data entry without verification, due to this reason it is vulnerable to forging. Therefore, an attacker sends data via UDP and imitates its response from a genuine server by falsifying the header data.

Since there is no method for DNS resolvers to verify data in their caches, wrong DNS data remains in the cache up to the time to live expires.

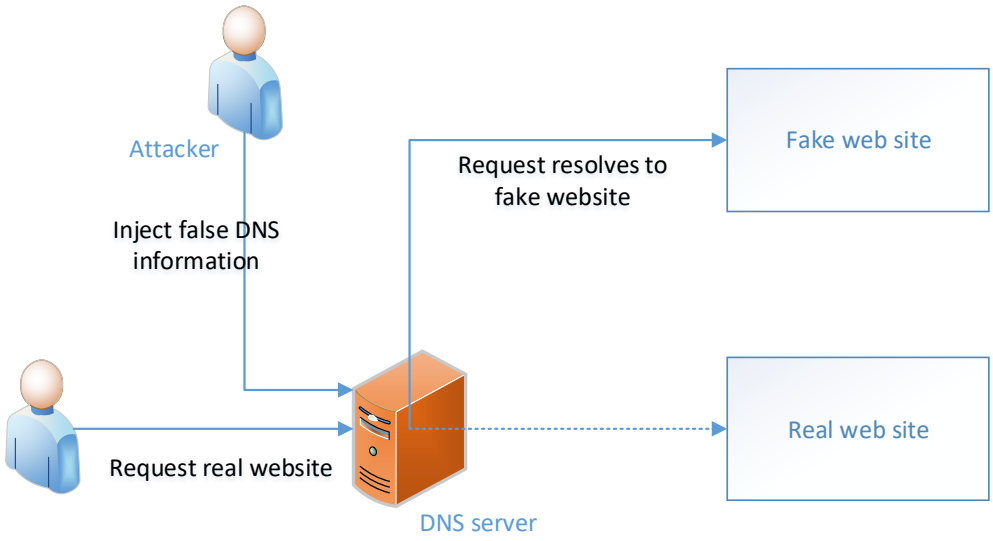


Fig 2-3. DNS cache poisoning

2.3.3 DNS Tunneling Attack

DNS tunneling is a method of DNS attack that exchanges data through DNS queries and responds in DNS protocol. It can be used to extract information by creating a communication channel with an external compromised machine.

It is often used for malicious activities, like data exfiltration. First, the attacker attacks a DNS client by malware that creates a tunnel to the attacker's device through a recursive DNS server located within the user's network, thus bypassing the firewall of the network. The malware sends data from attacked DNS client as a subdomain to the attacker's name server and then the attacker gets the data.

2.3.4 DNS Amplification Attack

Attacker deceives IP address of the targeted host and sends a small spoof DNS query to DNS resolver that produces a large reply that is directed to the targeted host. This assault is used to exhaust the resources of the target victim's servers faster. This type of attack is called DNS Amplification attack.

Furthermore, DNS amplification generally transmits DNS queries through one or more botnets that significantly growing the magnitude of traffic directed to the targeted servers and making it much challenging to find the attackers.

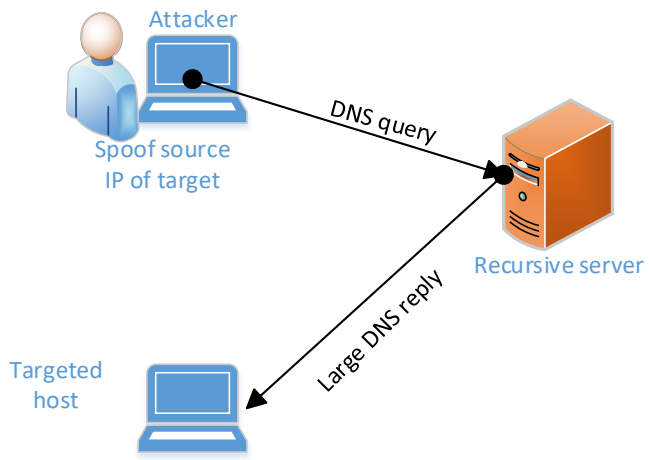


Fig 2-4. DNS amplification attack

2.3.5 Algorithmically Generated Domain-Flux Attacks

Domain-Flux is continuously changing the domain name of the C&C server through the execution of DGAs which are employed to prevent blacklisting of the C&C server domains.

The attacker first infects a DNS client by malware that generates Algorithmically Generated Domain names. Bots send generated domain names to DNS servers to communicate C&C server. A DGA continuously generates multiple C&C server domain names using a certain seed [14].

There are different types of malware attacks that have become more complicated by using DGA to find the C&C server that allows attackers to communicate with the infected systems, exfiltration data, and even remotely control the compromised device or server. Two of them are Ramnit and Qakbot.

2.3.5.1 Ramnit attack

Ramnit is a class of Trojan that allows attackers to control infected victims without direct contact, to steal banking and personal information, and open backdoors to download further malware.

Ramnit set up a communication channel with its C&C servers using a domain generation algorithm. The DGA is executed with a 32-bit input seed to create domain names that the Ramnit bot will attempt to communicate with.

The DGA code has two for-loops. The external for-loop creates a stream of characters from a PRNG using the seed. The internal for-loop chooses the length of domain that ranges between 8 and 17 characters and “cuts” the stream based on the length (as shown in Figure

2-5). For example: for the 32-bit input seed 1, the primary two “tmevtyqjvtnziv.com” (of length 14) and “mevtyqjvtnzi.com” (of length 12) are generated domain names.

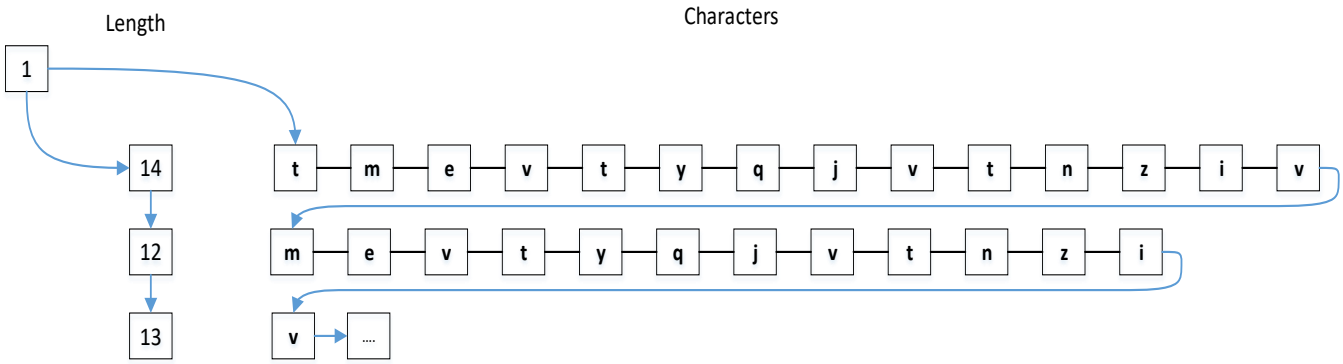


Figure 2-5: Ramnit DGA creates a stream of characters via a PRNG and selects their length based on the previous selection and its input seed.

The Ramnit worm can become a fast-spreading and very persistent worm that can affect its targets. Therefore, it’s essential to appropriately scan emails and attachments before viewing or opening their content. Aside from that, check the destination of a hyperlink before browsing it.

2.3.5.2 Qakbot attack

Qakbot depends on a DGA for C&C communication.

QakBot's DGA has the following general layout:

- Get the current date.
- Compute the CRC32 checksum of the date string.
- Provide the checksum into a Mersenne Twister random number generator.
- Generate a random amount of random alphabet characters, then add a TLD.

Current Date Acquisition

When Qakbot creates domain names using its DGA, the first step is to acquire the current date. And if a date is identified, the DGA moves on to the next step. Date strings would be of the general format "Date: Mon, 01 Jan 2018 00:00:00 GMT", and the first 6 characters are then stripped out.

Before the date is used, it is followed to a certain format, "%u. %s. %s. %08x". From left to right, the values placed into this formatted string are the day, month, year, and a special suffix value. But first, these inputs are modified slightly.

Days are transformed from 1-31 to 0-2, by subtracting 1 then dividing by 10, and months are made lowercase. The suffix value is either 0 or 1, depending on the version of QakBot, and this example uses a value of 1. Using the date "Mon, 01 Jan 2018 00:00:00 GMT", the changed date string format becomes "0.feb.2021.00000001".

CRC32 Checksum

The formatted date string and its length are then passed into a CRC32 checksum generating function, quickly recognizable due to its access of a table, whose values contain:

```
0x00000000, 0x1DB71064, 0x3B6E20C8, 0x26D930AC,
0x76DC4190, 0x6B6B51F4, 0x4DB26158, 0x5005713C,
0xEDB88320, 0xF00F9344, 0xCB61B38C, 0xD6D6A3E8,
0x86D3D2D4, 0x9B64C2B0, 0xBDBDF21C, 0xA00AE278
```

Mersenne Twister

An MT19937 Mersenne Twister random number generator is set up, recognizable from its use of the constant 0x6C078965, using the calculated CRC32 value as the seed:

Domain Generation

Qakbot is now prepared to start generating domain names. The main DGA is called with the following parameters shown below. The char **TLD pointer-to-strings array contains top-level domains that were stored in QakBot's memory, in this case, "com; net; org; info; biz; org" which is split by the ";" into an array of TLDs.

The first phase the DGA takes is to randomly generate a number between 0 and the number of TLDs to figure out which TLD to append at the end. Next, the DGA will randomly generate a number between 8 and 25, then loop that many times. In Each loop, it will generate a number between 0 and 25, corresponding to a letter of the alphabet to append to the domain name being generated.

After the domain name is generated, a "." and a randomly selected TLD are appended. This process is repeated in batches of 5 domains at a time, and the DGA may be called up to 5000 times.

This Qakbot DGA implementation written in C will generate the first five domains Qakbot would create if it was run on January 1st, 2018, using the date string "Date: Mon, 01 Jan 2018 00:00:00 GMT". For this date, the output is:

```
mdjjljhwtmefsumtdcpfnwp.org
aojpkrdnblwmocdob.org
svfamwehjqiht.com
zcevciwmsqbsgutvwhyp.com
bcrovlbako.biz
```

Qakbot Sample SHA256: 5B7A5A58E4AF312CD23E1F28597F2818953DD23ABDEEDB52ADB882958E2766CB

2.4 Attacks Detection Techniques

An **intrusion detection system (IDS)** is a software application or device that protects a network from attackers' policy violations. **I**ntrusion detection systems identify attacks with a low false alarm rate and a high detection rate. The current contributions in literature emphasis deep learning techniques, to build anomaly-based intrusion detection systems.

2.4.1 IDS Detection Types

An intrusion detection system (IDS) which is an important cyber security method, monitors the status of software and hardware running in the network [17]. Cyber security methods primarily consist of anti-virus software, firewalls, and virtual private network and intrusion detection systems [18]. Among them, an IDS is a type of detection system that plays a crucial role in protecting networks from internal and external attacks by controlling the states of software and hardware running in a network. There are several types of IDS, the most common are signature and anomaly-based detection [15, 20] as shown in Fig 2-5. Hybrid is the combination of both detection types to offer better prevention and detection capabilities [18].

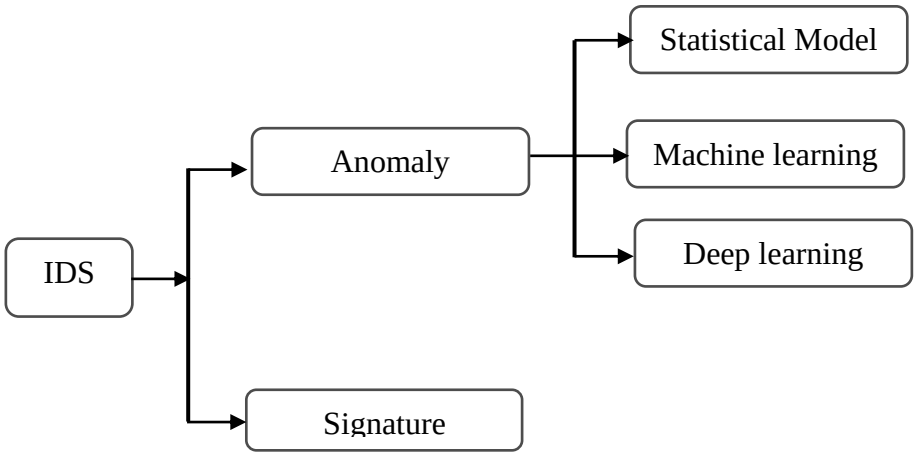


Fig 2-6. IDS detection types

2.4.1.1 Signature-based Detection

Signature-based IDS identifies possible attacks by looking for specific patterns, such as arrangements of bytes in network traffic. It is a static method and is limited to the detection of only known attack patterns. [16, 19]. That means signature-based IDS can simply identify known attacks, it is difficult to detect novel attacks, for which no pattern is existing.

Nowadays, a large number of new threats are created or developed every day, which decreases the efficiency of signature-based IDS because it is not a practical solution to update the signatures databases every few minutes [13].

2.4.1.2 Anomaly-based Detection

Anomaly detection is the identification of unexpected observations, events, or things that vary significantly from the norm [13]. Any type of anomaly detection has two basic assumptions. Those are they are very rare events and the features are significantly varying from typical instances. This approach like the rule-based systems technique capable to detect previously unknown attacks, it can suffer from false positives [15].

It is a technology aimed to identify and adapt to unknown attacks. There are different anomaly detection techniques. Some of them use artificial intelligence models consisting of machine learning or deep learning to train a model based on normal activity to detect any anomalies. They can detect new attacks [16]. Deep learning requires large datasets for training without facing the overfitting problem as it may have more capability to generalize than machine learning models [13, 15] as shown Fig 2-6.

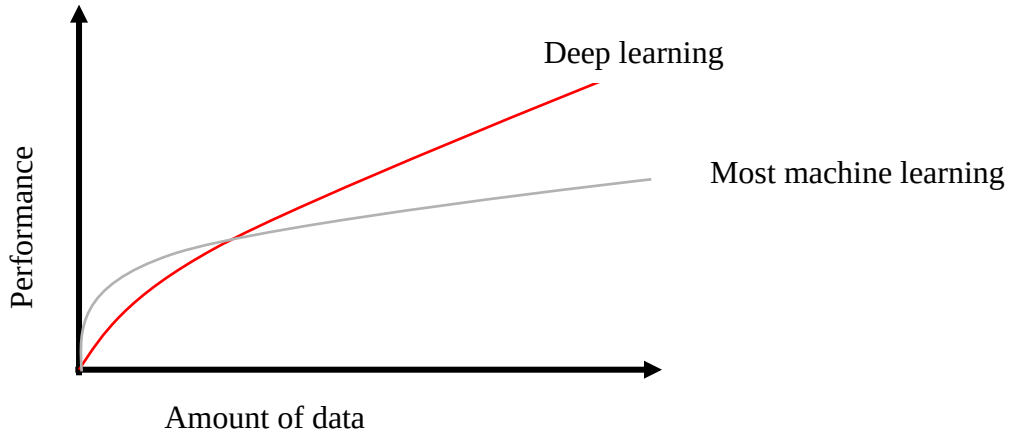


Fig 2-7. Deep learning performance-optimized when using a larger dataset

CHAPTER THREE

3 Deep Learning

3.1 Introduction

Deep learning is a subclass of machine learning techniques that teach computers to do human-like tasks: such as speech recognition, image identification, and others. Deep learning is a basic tool for driverless cars, supporting them to identify a traffic sign and others. It is also vital to voice control in phones, TVs and, tablets. Deep learning is attaining lots of attention recently and achieving good outcomes that were impossible before.

Deep learning models learn to implement classification directly from text, images, or sound. They can accomplish it with high accuracy, sometimes greater than human-level performance. To train the models, neural network architectures and large labeled datasets are used.

Deep learning accomplishes very good results because of accuracy. Deep learning does higher accuracy than ever before. It helps electronics have been made based on user anticipations, and it is also implemented in driverless cars.

Deep learning needs a large size of labeled datasets and high computing power**.** For example, the development of driverless cars needs large numbers of images and videos. The deep learning algorithm can learn without human control. It can be used in many activities like e-commerce, banking, healthcare, etc.

3.1.1 Application of Deep learning

Deep learning applications are used in various activities some of them are:

i. Text Analysis & Understanding: Text analysis contains sorting of documents, opinions or views, automatic translation, analysis, etc. Recurrent neural networks are the most appropriate deep learning algorithm for textual data because it is a sequential data type.
ii. Automated Driving: Driverless cars investigators are implementing deep learning to recognize objects automatically like traffic lights and others. Moreover, deep learning is used to recognize pedestrians, which contributes to the reduction of accidents.
iii. Medical Research: Cancer researchers are implementing deep learning to identify cancer cells.
iv. Aerospace and Defense: Deep learning is used to distinguish objects immediately from satellites.
v. Electronics: Automated hearing and speech translation have been done using deep learning. For instance, home aid devices that reply to your voice are supported by deep learning applications.

3.1.2 How Deep Learning Works

Deep learning models are taught by using large and labeled datasets. Neural networks extract features from the data without manual feature extraction. It also selects features to increase the performance of the model.

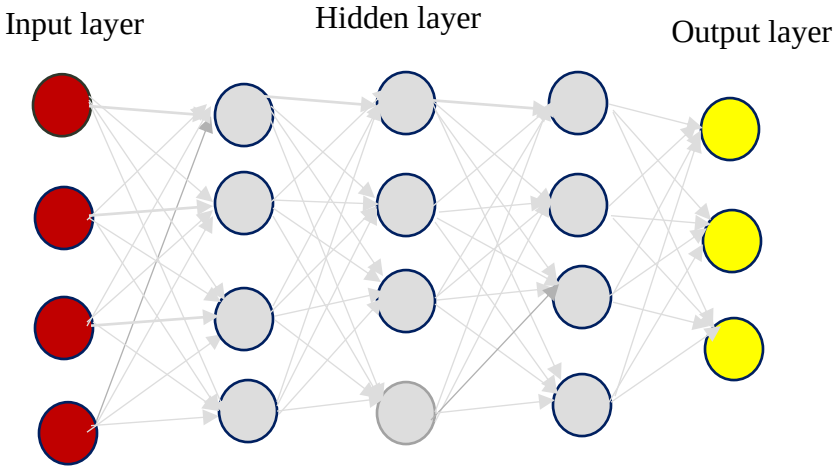


Fig 3-1. Neural networks are structured in layers consisting of interconnected nodes.

One of the common types of deep neural networks is known as convolutional neural networks (CNN). A CNN uses 2D convolutional layers that make it appropriate to processing 2D data, such as images. It extracts features from images.

The correctness of the model is evaluated using the cost (or loss) function. This is also called mean squared error (MSE). In the equation below,

$$Cost\ Function = \ MSE = 1/2m \sum_{i=1}^{m} (y' - y)^2$$

- i represents the index of the sample,
- m is the number of samples,
- y' is the predicted value, and
- y is the actual value.

The aim of minimizing the value of the cost function is to improve the accuracy of the model. When the model amends its weights and bias, it uses the cost function to touch the point of local minimum or convergence. The process in which the algorithm amends its weights is through gradient descent, allowing the model to find the direction to reduce errors. With each training sample, the parameters of the model are modified to converge at the minimum.

3.2 Difference between Machine Learning and Deep Learning

Deep learning is a subclass of machine learning. In deep learning, important features are extracted automatically from images and continue to improve as the size of datasets increases, whereas machine learning converges at a certain level of performance when we add extra instances to the network.

Thus, effective deep learning needs a very large quantity of data to train the model to get reliable outputs.

3.3 Supervised and Unsupervised Deep Learning Algorithms

Mainly we can categorize Deep learning algorithms into two types

1. Supervised deep learning algorithms
2. Unsupervised deep learning algorithms

3.3.1 Supervised Deep Learning Algorithms

Supervised deep learning algorithms are trained using labeled datasets.

$$Y = f(X)$$

The aim is to estimate the mapping function that predicts the output variables (Y) for input data (X). It is called supervised. Learning ends when the algorithm achieves an adequate level of accuracy or performance.

3.3.2 Unsupervised Deep Learning Algorithms

Unsupervised learning is a type of deep learning in which the algorithms only have input data (X) and no corresponding output variables.

3.4 Neural network

A neural network is a system of software models and/or hardware to behave like interconnected neurons in the human brain. Neural Networks are a variety of deep learning technology, which is also a subset of artificial intelligence, or AI.

Some types of neural networks in deep learning are:

1. Artificial Neural Network
2. Convolutional Neural Network
3. Recurrent Neural Network

3.4.1 Artificial Neural Network

An artificial Neural Network is a set of multiple perceptron /neurons at each layer. ANN is also known as Feed-forward neural networks. It is one of the simplest types of neural networks. They pass information in one direction, data enter through input nodes and exits through the output nodes. It has no recurrent connection to the hidden state**.** So it can't handle sequential data, considers only the present input, and cannot remember previous inputs. The solution to these problems is the RNN.

Advantages and Disadvantages of artificial neural networks

Artificial neural networks have advantages some of them are:

- Parallel processing capabilities (perform more than one job at a time).
- ability to learn and model nonlinear, complex relationships
- High fault tolerance or the damaged of one or more cells of the ANN will not halt classification or prediction.
- The ability to generalize or predict the output of unseen data.

The disadvantages of ANNs include:

- Appropriate or good artificial neural network architecture can only be set up through trial and error methods and experience.
- All problems have been translated into numerical values before they can be presented to the ANN.
- The lack of explanation behind searching solutions is one of the biggest disadvantages of ANNs.

3.4.2 Convolutional Neural Networks

CNN's are Neural Networks that can identify and categorize particular features from images and are commonly used for analyzing visual images. A CNN architecture is made of fully-connected layers, pooling layers, and convolutional layers. When these layers are piled, a CNN architecture will be built.

3.4.3 Recurrent Neural Networks

It is more complex. The principle of RNN is to feedback the output of processing nodes and feed the result back to input again. Each node in the RNN model acts as a memory cell, to hold some information as it goes to the next time step. The data to be used later,

the model will have to remember and work for next step will go on this process. If its prediction is wrong, then the system continues to correct prediction during backpropagation.

Recurrent Neural Networks allow us to model sequential data and time-dependent problems [17]. However, it is challenging to train because of the gradient problem. RNNs have the problem of vanishing gradients. This creates the learning of long data sequences is challenging. Long training time, bad accuracy, and poor performance are the main issues in gradient problems.

Some of the common and effective ways to deal with gradient problems are Long Short-Term Memory Network (LSTMs), gated recurrent unit (GRU), and Bi-RNN [17].

	ANN	RNN	CNN
Data	Tabular Data	Sequential Data	Image
Recurrent Connection	No	Yes	No
Parameter Sharing	No	Yes	Yes
Vanishing & Exploding Gradient	Yes	Yes	Yes

Table 3-1. Summarized some of the differences among different types of neural networks.

3.5 Deep Learning Models for Multi-class DNS attacks Detection

3.5.1 CNN-Based Detection Model.

It uses a different type of multilayer perceptron and contains one or more convolutional layers that can be either fully connected or pooled. It is mainly known in image recognition, natural language processing, facial recognition, image classification, and signal processing.

3.5.2 LSTM-Based Detection Model

Long Short-Term Memory (LSTM) networks are special types of Recurrent Neural Networks that learn order dependence in sequence.

LSTM networks were intended for long-term dependencies to remember information for a long period without learning, again and again. Because of this, the entire process becomes simpler and faster. It includes an inbuilt memory to store information that is used for processing, predicting, and classifying based on data.

There are five architectural elements in LSTM, the three different gates, and two states. The gates regulate information flow in an LSTM cell. The remaining two are hidden state and cell state. The purposes of gates as shown below

1. Forget gate: The information that is no longer useful in the cell state is removed with the forget gate.
2. Input gate: It adds Adding valuable information to the cell state.
3. Output gate: The task of pulling out valuable information from the current cell state to be offered as output is done by the output gate.

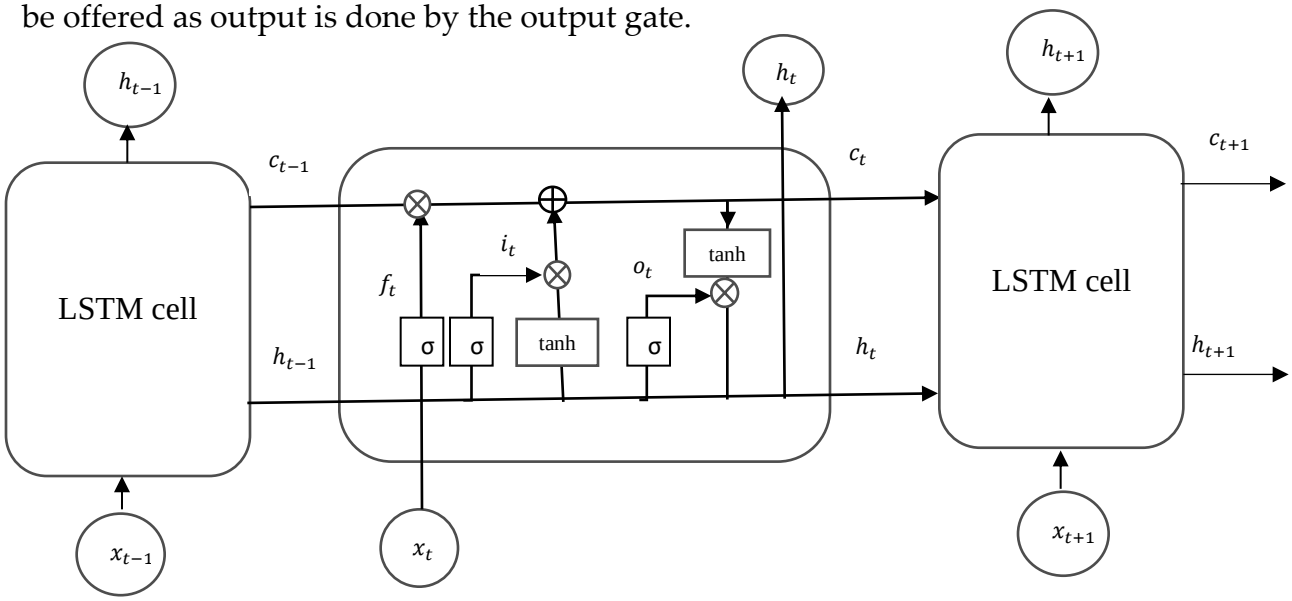


Fig 3-2. Internal structure of an LSTM model

LSTM is expressed by equations:

$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$, ... (1)

$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$, ... (2)

$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}ch_{t-1} + b_c)$, ... (3)

$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$, ... (4)

$h_t = o_t tanh(c_t)$ . ..... (5)

Where some types of neural networks in deep learning are:

f_t = forget gate, i_t = input gate, O_t = output gate , h_t =hidden state, c_t= cell state

First, the forget gate f decides which information to remove from the cell state [14]. Next, the input gate i, which stores the recent information, decides which information to update in the cell state among the new inputs. It generates a new candidate for transfer to the cell through the tanh layer. Then, the information produced through the forget and input gates is joined to update the cell state. Lastly, the output gate 0_t decides which part of the cell state to output, and the final result value h_t is determined through the cell state output and the output gate. Some of the famous applications of LSTM include: Language Modelling, Machine Translation, Image Captioning, and Handwriting generation.

3.5.3 Gated Recurrent Unit (GRU)

GRU is an advancement of the standard RNN. It is similar to Long Short Term Memory (LSTM). Just like LSTM, GRU uses gates to regulate the flow of information. They are relatively new as related to LSTM. This is the reason they offer some improvement over LSTM and have simpler architecture.

GRU is that, unlike LSTM, it does not have a separate cell state (C_t) and used the hidden state to transfer information. It only has two gates, an update gate, and a reset gate. It only has a hidden state (H_t). Due to the simpler architecture, GRUs are faster to train.

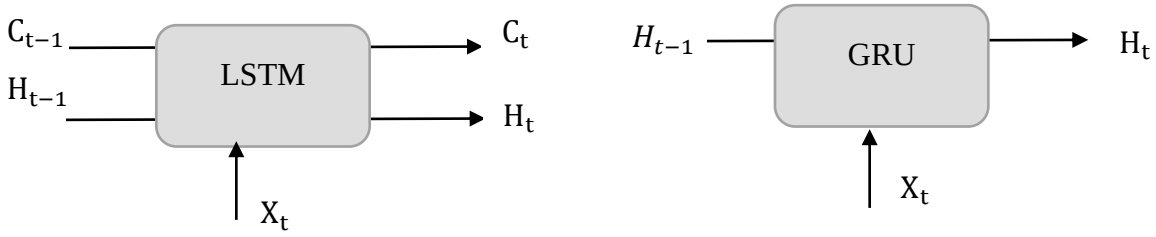


Fig 3-3. The architecture of the Gated Recurrent Unit and LSTM

Now let's understand how GRU works. Here we have a GRU cell which is more or less similar to an LSTM cell. At each timestamp t, it takes an input X_t and the hidden state H_{t-1} from the previous timestamp t-1. Later, it outputs a new hidden state H_t which is again passed to the next timestamp.

3.5.4 Bidirectional LSTM

A **Bidirectional LSTM**, or **BiLSTM**, is a sequence processing model that contains two LSTMs: one taking input to forwarding direction, and the other to the backward direction. BiLSTM accesses the information efficiently in both directions. Thus, BiLSTM learns bidirectional information whereas unidirectional information is learned by LSTM [14].

3.5.5 The difference between LSTM and BILSTM

LSTM stores information that has already passed through it from inputs using the hidden state. It only stores information of the past because the only inputs it has seen. While using bidirectional, it will access our inputs in two directions, one from past to future and the other from future to past. Because of this, BILSTMs show very good results.

CHAPTER FOUR

4. Experimental Analysis

In this chapter, detailed explanations linked to experimental processes have been described in this research. As shown in figure 4.1 below, the experimental process has generally five portions: data collection and cleaning, data preprocessing, Training model and evaluation, and lastly the model comparison. The following parts describe how these were accomplished.

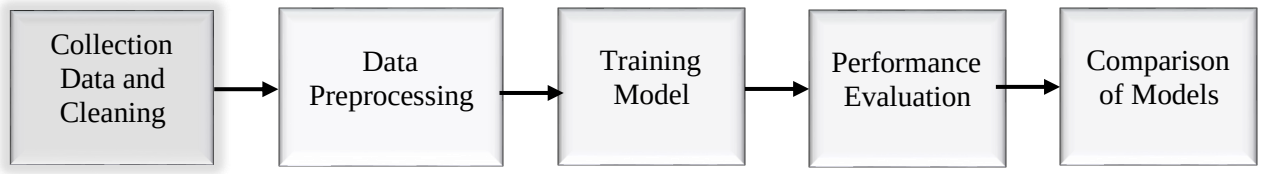


Figure 4-1 Experimental process showing data collection, preprocessing, training models, and Evaluation

4.1 Data Collection and Cleaning

This research needs datasets that characterizes both normal and anomaly type of DNS attacks. To gather DNS attacks datasets from ethiotelecom is challenging or they are unwilling to provide those datasets for this research, which might be to secure the systems. So we collected such types of datasets from global datasets like qakbot_dga, Flood Attack, ramnit_dga, and tunneling DNS attacks. The Flood attack dataset sample is shown in Tables 4-2 below. The source of tunneling, DNS Flood, domain generating

algorithms datasets are https://data.mendeley.com/datasets/ and https://research.unsw.edu.au/ projects /unsw-nb15-dataset and Bambenek Consulting OSINT [14] respectively.

Table 4-1 shows the type of dataset, type of instance, and numbers of instances. The datasets are collected from global datasets in form of CSV file format. Each dataset has normal and anomaly types of instance, for example, ramnit contains 22,499 normal and 22,145 anomaly instance as shown below.

The dataset cleaning is done by filling empty cells, removing redundancy, maintaining the format of features, and others.

Type of dataset (DNS Attack Type)	Type of instances	No of instances	Total
Tunneling	Normal	19,296	36,522
	Anomaly	17,226	
Ramnit	Normal	22,499	44,644
	Anomaly	22,145	
Qakbot	Normal	22,187	44,464
	Anomaly	22,277	
Flood Attack	Normal	17702	70,783
	Anomaly	53081	
Total	Normal	81,684	196,413
	Anomaly	114,729	

Table 4-1 Number of instances for four types of datasets and their respective number and type instances.

scrip	sport	dstip	dsport	proto	state	dur	sbytes	dbytes	. . .	label
59.166.0.0	6055	149.171.126.5	54145	tcp	FIN	0.072974	4238	60788	. . .	4
59.166.0.0	7832	149.171.126.3	5607	tcp	FIN	0.144951	5174	91072		4
59.166.0.8	11397	149.171.126.6	21	tcp	FIN	0.116107	2934	3742		4
59.166.0.0	3804	149.171.126.3	53	udp	CON	0.000986	146	178		4
59.166.0.8	14339	149.171.126.6	14724	tcp	FIN	0.03848	8928	320		4
59.166.0.8	39094	149.171.126.3	53	udp	CON	0.001026	130	162		4
59.166.0.0	10845	149.171.126.7	5190	tcp	FIN	0.005645	1064	2260		4
59.166.0.3	45642	149.171.126.5	80	tcp	FIN	0.020018	1036	824		4
59.166.0.4	1931	149.171.126.6	6881	tcp	FIN	3.27602	13766	548216		4
59.166.0.8	25724	149.171.126.6	5190	tcp	FIN	0.153293	1272	2572		4
59.166.0.3	49668	149.171.126.7	80	tcp	FIN	1.010939	1684	10168		4
59.166.0.2	14951	149.171.126.6	53	udp	CON	0.001025	146	178		4
59.166.0.4	27545	149.171.126.3	111	udp	CON	0.004571	568	320		4
59.166.0.4	56591	149.171.126.3	39730	udp	CON	0.001662	536	304		4

Where Scrip = Source ip address, Sport = Source port number, Dstip = destination ip, Dsport = destination port number, Proto = Transaction protocol, State = state and its dependent protocol, Dur = Record total duration, Sbytes = source to destination transaction bytes and Dbytes =destination to source transaction bytes.

Table 4-2 Sample of Flood attack dataset

4.2 Data preprocessing

Data preprocessing is the first and crucial phase in the building of a deep learning model. It is a process of making suitable data to increase the accuracy of classification. In this work, the required datasets were collected from global datasets as mentioned before. Tunneling and domain generating algorithms, like qakbot and ramnit were tokenized and represented each alphabet and others in the domain name by numeric. Our Flood DNS attacks datasets include 36 numeric features and four non-numeric (two IP-address and two categorical features).

In general, in this section data preprocessing involves numericalize and normalizing datasets that optimize the classification of deep learning models. Each portion will be discussed in detail in the next sub-sections.

4.2.1 Numericalization

Three of the collected datasets contain the domain name and label features. To train deep learning models, all dataset features value must be changed into numerals. So we converted the non-numeral features like characters in the domain name, source IP address, destination IP address, proto, and others into numerals. In this section, we show that domain name change into numerals as shown below in Table 4-3 and Table 4-4.

Domain Name	label
stsuodwqssexpb.cc	3
wjneysssgrcdjf.cc	3
ovyvnbkjnqpeqjrar.net	3
wknyvkmnvqirwe.su	3
hxgvbkmfylcukngdf.cc	3

Table 4-3. Tunneling Datasets

Domain Name	label
... 30.0 29.0 31.0 25.0 14.0 33.0 27.0 29.0 29.0 15.0 34.0 26.0 12.0 76.0 13.0 13.0	3
... 20.0 24.0 15.0 35.0 29.0 29.0 29.0 17.0 28.0 13.0 14.0 20.0 16.0 76.0 13.0 13.0	3
... 12.0 21.0 20.0 24.0 27.0 26.0 15.0 27.0 20.0 28.0 11.0 28.0 76.0 24.0 15.0 30.0	3
... 21.0 24.0 35.0 32.0 21.0 23.0 24.0 32.0 27.0 19.0 28.0 33.0 15.0 76.0 29.0 31.0	3
... 12.0 21.0 23.0 16.0 35.0 22.0 13.0 31.0 21.0 24.0 17.0 14.0 16.0 76.0 13.0 13.0	3

Table 4-4. Numericalization of tunneling dataset

4.2.2 Normalization

Some features in the datasets, such as dur, sbytes, and dbytes in Flood attack have a big variance between the minimum and maximum values. We applied a logarithmic scaling method to map to the range [0, 1] [11]. Normalization is not necessary for every feature or dataset, but a dataset like UNSW-NB15-dataset, where the features have different ranges of values, will require normalizing the values. Normalization changes the values of a dataset to a common scale. If the values of the features do not have a similar range,

then the gradient descents may take too long to converge [16]. We can assure that the gradient descents can converge more quickly by normalizing. Normalizing data also increase the accuracy of a classifier.

4.3 Training of Classification Algorithms

The following tasks, after preprocessing of the collected datasets, are training models using chosen deep learning algorithms and datasets to build classification models. This phase includes setup hyper parameters, selecting training algorithms and train models, and others for multi-class classification.

4.3.1 Setup hyper parameters

To build the deep learning model, first, we carefully chose hyper-parameters for multi-class classification. We determined the following hyper-parameters: batch size, the number of epochs, learning rate, dropout, and activation function.

To prevent overfitting in models, we introduced dropout during the training of the model.

Parameter Setting	Multi-class Classifiers
Epoch	5
Batch size	1500
Learning rate	0.01
Dropout	0.5
Optimizer	Adam
Activation Function	softmax
Loss function	categorical_crossentropy

Table 4-5: Hyper parameters of the deep learning models.

4.3.2 Training Models

Four supervised deep learning models have been selected for the DNS attacks multi-class classification. These are LSTM, BiLSTM, GRU and BiGRU.

While training these models, '0' label was used to denote normal data traffic flow instances while other labels are used to denote anomaly instances. We use the 10-fold cross-validation technique to train deep learning models.

4.4 Performance Evaluation

As mentioned before, to evaluate and compare the performance of deep learning models that are used to detect DNS attacks, we use confusion matrix, accuracy, F1-measure, and area under the ROC curves (also known as AUC) performance metrics.

4.4.1 Classification Accuracy

The classification accuracy is usually used to evaluate the classification performance, and it is defined as a ratio of correctly classified to the total number of input samples often presented as a percentage.

$$\text{Accuracy (\%)} = \frac{\text{number of correct classified} * 100}{\text{Total numbers of input samples}} = \frac{TP + TN}{TP + FP + TN + FN} * 100$$

Where;

TP represents the number of samples, the actual value was positive and the model classified as positive value.

TN represents the number of samples, the actual value was negative and the model classified as negative value.

FP represents the number of samples; the actual value was negative but the model classified as positive value.

FN represents the number of samples; the actual value was positive but the model classified as negative value.

4.4.2 Weighted- average and macro-average

Weighted-average and macro-average are also used to compare the performances of the deep learning models [14]. Weighted-average calculates the performance considering the proportion for each label in the dataset whereas macro average calculates without considering the proportion for each label in the dataset. We get the macro average by independently calculating and summing the performances of each class and dividing them by the total number of classes. Macro average treats all classes as having the same number of data; thus, we can measure the average performance per class. If the amount of data is unbalanced in multiclass classification, the weighted average is considered to be fairer than the macro average.

$$W = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}$$

Where

W = weighted average

W_i = weights applied to x values

X_i = data values to be averages

n = number of terms to be averaged

4.4.3 Confusion Matrix

In multi-class classification, the result is often presented as a two-dimensional confusion matrix with a row and column for each class. A confusion matrix is the summary of classification results. The numbers of accurate and inaccurate classifications are summarized with counts of true positive, true negatives, false positives, and false negatives which are defined as the previous subsection.

Classified as →	Confusion Matrix	Class A	Class B	Class C
Actual Class	**Class A**	**A**	**D**	**G**
	Class B	**B**	**E**	**H**
	Class C	**C**	**F**	**I**

Table 4-6 Confusion Matrix

The diagonal elements as shown in Table 4-6 denote the numbers of correct classification (A, E, and I) and other elements denote the numbers of incorrect classification (B, C, F, D, G, and H).

4.4.4 F1-score

The F1 score, which is also called as F1-measure is the harmonic mean of the recall and precision, it measures the performance of prediction when just as much importance is given to recall as to precision [14]. F1-score is popularly used as a general measure to accurately calculate the performance of a model when the data are unbalanced.

$$\text{F1}-\text{score} = \frac{2*\text{precision} * \text{Recall}}{\text{Precision}+\text{Recall}}$$

Where "Precision" is the number of accurate positive outcomes divided by the number of positive outcomes predicted by the classifier and "Recall" is the number of accurate

positive outcomes divided by the number of all instances that should have been identified as positive. Alternatively, it can be put as follows;

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

4.4.5 AUC-ROC Curves

AUC-ROC curve supports us to plot how well our deep learning model is performing. However, it uses for only binary classification, we can modify it to measure multi-class classification too by using the one vs rest technique.

It is a probability curve that draws the TPR against FPR at different threshold values. The AUC is the measure of the capability of a model to distinguish classes.

A. Sensitivity / Recall / True Positive Rate:

Sensitivity represents what percentage of the positive class is correctly categorized. A lower FNR and a higher TPR are needed to correctly classify the positive class.

$$\text{Sensitivity} = \frac{TP}{FN+TP}$$

B. Specificity / True Negative Rate:

Specificity informs us what ratio of the negative class is correctly classified.

False Positive Rate

FPR informs us what fraction of the negative class is incorrectly classified by the classifier.

$$\text{FPR} = \frac{FP}{FP+TN} = 1 - \text{Specificity}$$

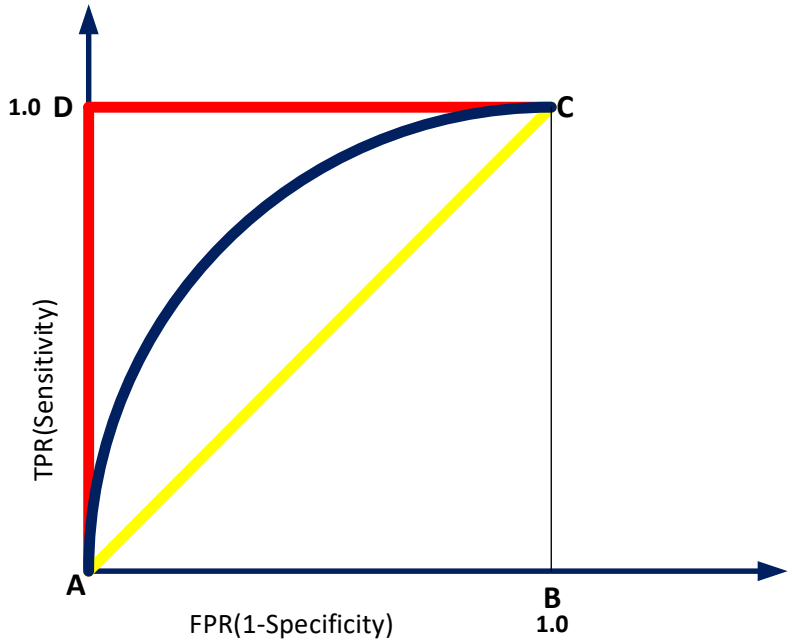


Fig 4-2. AUC-ROC Curves

When AUC = 1, then the classifier can perfectly classify all Negative and Positive classes. It is represented by a red line (ADC) as shown in Fig 4-2.

If AUC = 0, then the classifier would be classifying all Positives as Negatives, and all Negatives as Positives.

When 0.5<AUC<1, there is a high possibility the model will be able to differentiate the positive class from the negative class values. So the classifier can detect more True negatives and True positives than False Negatives and False Positives. It is represented blue-black curve (AC).

When AUC=0.5, then the classifier cannot distinguish between Positive and Negative classes that mean the model is classifying randomly for all the data points. It is represented in the yellow line (AC).

Hence, the better classifier has a higher AUC value and can differentiate positive and negative classes.

4.5 Comparison of different deep learning models

Select best multi-class classifier using performance metrics such as mean test accuracy, F1-score, AUC, and others.

4.6 Experimental environments

The experimental environment consisted of an Intel® Core™ i7-8665U CPU @ 1.90GHz 2.11 GHz, 8 GB RAM, and running Windows 10 operating system with 64-bit operating system, x64-based processor.

Python 3.7.7 64-bit with the scikit-learn, TensorFlow, and Keras packages were used to develop the programs along with other Python open-source libraries.

CHAPTER FIVE

5. Discussion

As discussed in the previous sections, 10-fold cross-validation technique has been used in these evaluations to fairly measure the models and increase accuracy. In 10-fold cross-validation, the data are divided into ten equal portions. Then, nine portions are left out for training, and validation is accomplished on the remaining one portion. This process is repeated until all parts are used for validation [16, 21]. Confusion matrix, mean accuracy, F1-score, and AUC-ROC curves were used to evaluate the performance of the deep learning classifiers.

In this section, we compare and select the best classifier that has the highest DNS attacks classification accuracy among four types of deep learning multi-class classifiers. And we compare our selected model with the best classifier model, Ensemble, which we reviewed in another paper [14].

5.1 Evaluation of multi-class Classifiers

The multi-class classification performances depend on the type of deep learning models, hyper parameters, datasets, and others. We computed the performance of classifiers and the results are shown in Table 5-1 and Figure 5-1.

In this section, we compared the classifiers using mean test accuracy, F1-score, and AUC-ROC.

Models	Number of Features	Mean test Accuracy (%)	Time Taken to Classify (avg)	Accuracy (%)	Weight Average		
					Precision	Recall	F1-score
LSTM	286	96.94% (+/- 0.63%)	72sec	0.9713	0.9717	0.9713	0.9714
BILSTM	286	96.98% (+/- 0.70%)	598sec	0.9729	0.9737	0.9729	0.9730
BiGRU	286	97.18% (+/- 0.72%)	145sec	0.9750	0.9753	0.9750	0.9750
GRU	286	97.00% (+/- 0.78%)	36sec	0.9710	0.9712	0.9710	0.9711

Table 5-1. Overall classification performance of four types of multi-class classifiers.

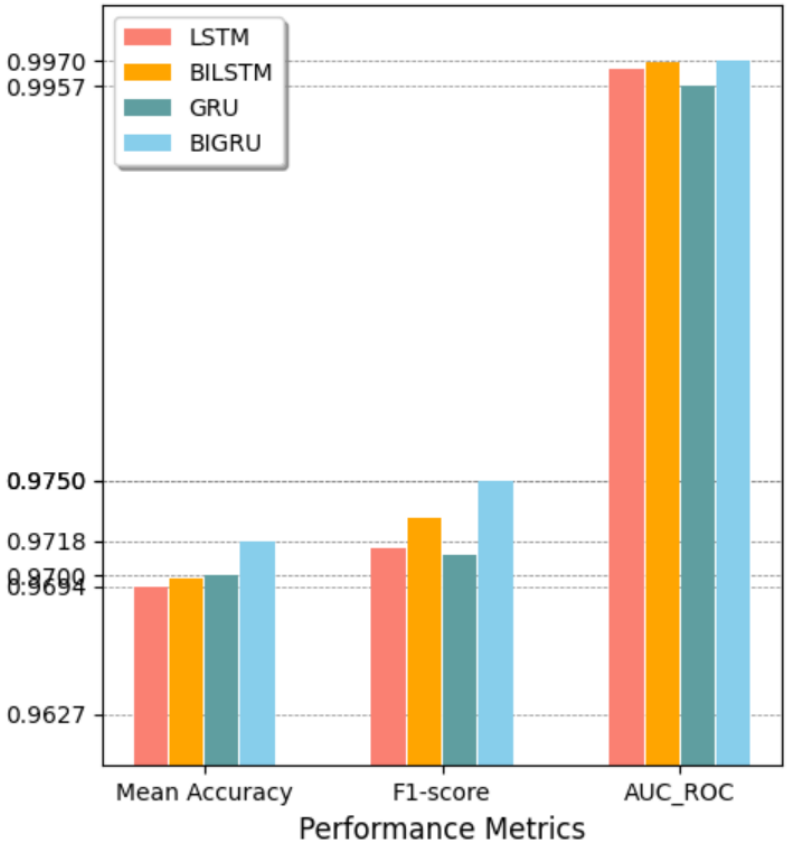


Figure 5-1 Overall classification performance comparison of deep learning models

Deep Learning Algorithms		Confusion Matrix	F1 Score	Support	Macro Average of AUC-ROC
BiLSTM	Normal : 0	[[7893 51 23 1 72]	0.9792	8040	0. 9969
	Ramnit : 1	[34 2181 0 0 0]	0.9442	2215	
	Qakbot: 2	[15 154 2058 0 0]	0.9508	2227	
	Tunnel : 3	[0 19 21 1683 0]	0.9880	1723	
	Flood : 4	[139 0 0 0 5169]]	0.9800	5308	
	Accuracy		0.9729	19513	
	Macro Average	0.9697 0.9682	0.9684	19513	
	Weighted Average	0.9737 0.9729	0.9730	19513	
LSTM	Normal	[[7908 47 13 0 72]	0.9781	8040	0.9966
	Ramnit	[51 2126 38 0 0]	0.9376	2215	
	Qakbot	[31 128 2068 0 0]	0.9471	2227	
	Tunnel	[1 19 21 1682 0]	0.9880	1723	
	Flood	[139 0 0 0 5169]]	0.9800	5308	
	Accuracy		0.9713	19513	
	Macro Average	0.9683 0.9644	0.9662	19513	
	Weighted Average	0.9717 0.9713	0.9714	19513	
GRU	Normal	[[7900 56 11 1 72]	0.9791	8040	0.9957
	Ramnit	[26 2093 96 0 0]	0.9365	2215	
	Qakbot	[32 93 2102 0 0]	0.9420	2227	
	Tunnel	[0 13 27 1683 0]	0.9880	1723	
	Flood	[139 0 0 0 5169]]	0.9800	5308	
	Accuracy		0.9710	19513	
	Macro Average	0.9659 0.9644	0.9651	19513	
	Weighted Average	0.9712 0.9710	0.9711	19513	
BiGRU	Normal	[[7935 21 12 0 72]	0.9798	8040	0.9970
	Ramnit	[52 2163 0 0 0]	0.9545	2215	
	Qakbot	[32 120 2075 0 0]	0.9560	2227	
	Tunnel	[0 13 27 1683 0]	0.9883	1723	
	Flood	[139 0 0 0 5169]]	0.9800	5308	
	Accuracy		0.9750	19513	
	Macro Average	0.9748 0.9692	0.9717	19513	
	Weighted Average	0.9753 0.9750	0.9750	19513	

Table 5-2. Performance evaluations of four types of deep learning multi-class classifiers.

We have observed from Table 5-1, BIGRU and BILSTM deliver a comparable performance and BIGRU is a little bit higher performance than BILSTM.

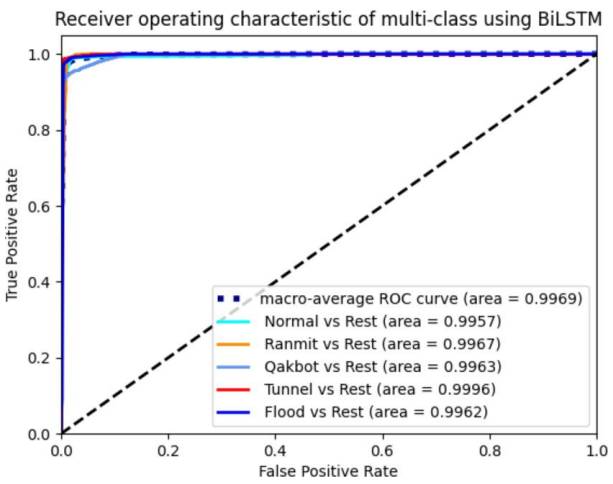
Experimental outcomes showed that the mean test accuracy of the BiGRU was 97.18%, which was greater than GRU. This showed that the bidirectional pattern of BiGRU is more efficient than the unidirectional pattern of GRU for dealing with sequential information. And the mean test accuracy of the BILSTM model was 96.98%, which was also greater than LSTM (96.94%). BIGRU model had the highest mean test accuracy among others models.

Considering the other performance metrics, the F1-score of the BiLSTM was 0.9730, which was higher than LSTM (0.9714). And F1-score of the BIGRU model (0.9750) was greater than BILSTM. This metric also shows that the BIGRU model has the highest performance among other classifiers as shown in Table 5-1.

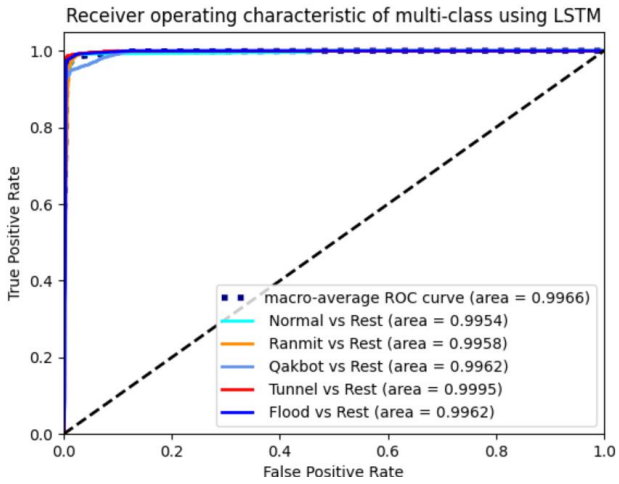
On other hand, BiLSTM needed 598 secs which was the highest to classify these multi-class attacks. The GRU required 36 secs, the shortest time. BIGRU required less time to classify compared to BILSTM, but it needed more time than GRU and LSTM.

Using both F1-score and mean test accuracy results, BIGRU was the best multi-class classifier for our works and effective for sequential information. However, the time taken to classify attacks was high.

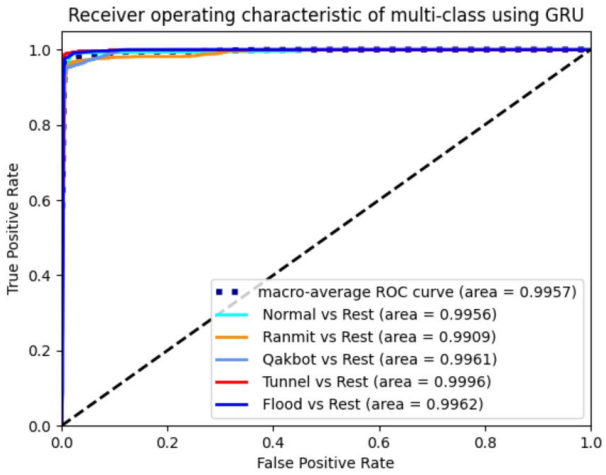
Considering the other performance metric, the Macro average of AUC-ROC, it showed an almost similar performance pattern to both mean test accuracy and F1-measure. BIGRU AUC was 0.9970 which also indicates the highest classification performance than others and BILSTM AUC was 0.9969 which was the second-best as shown Fig 5-2(a, b, c and d).



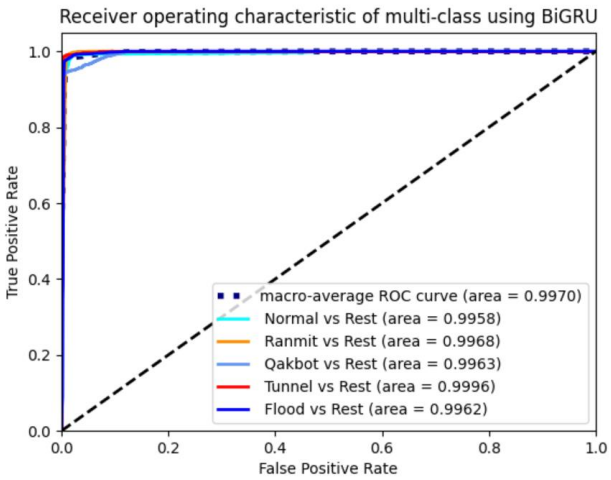
(a)



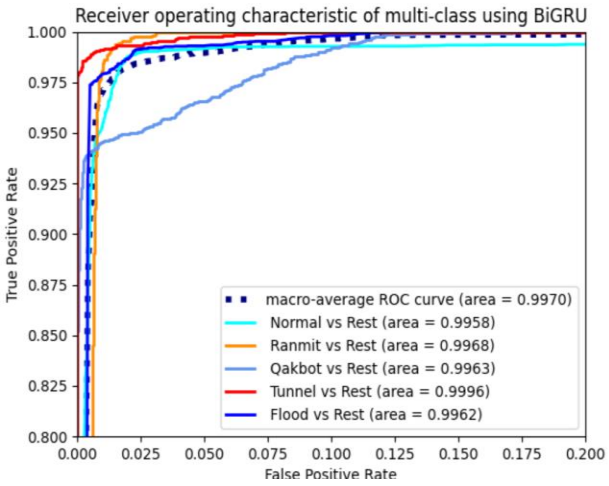
(b)



(c)



(d)



(e)

Fig 5-2. ROC curves of multi-class DNS attacks using different deep learning classifiers

The ROC curves in Figure 5-2 above show that the macro average ROC curve of BIGRU most closely to the top left corner or highest AUC value than the other three classification models that indicate the best classification performance is provided by BIGRU (Fig5-2(d)). And then BILSTM (Fig5-2(a)) was the second-best and GRU become the last one. Fig5-2(e) depicts the magnified BIGRU AUC-ROC curve.

Using three performance metrics, BIGRU high preferable classification model to classify our multi-class DNS attack compared to the other three deep learning algorithms. The values of mean test accuracy, F1-score, and AUC are 97.18%, 0.9750, and 0.9970 respectively.

5.2 Per class classification performance of models which we reviewed [14]

An ensemble model is the best multi-class classifier to sort ramnit and qakbot class in reviewed work [14] as shown in table 5-3 below.

Class Name	CNN			LSTM_Att			BILSTM_Att			Ensemble		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Ramnit	0.7637	0.8408	0.8004	0.8287	0.8964	0.8612	0.8321	0.8951	0.8625	**0.8486**	**0.9023**	**0.8747**
Qakbot	0.7164	0.6620	0.6882	0.7491	0.7995	0.7735	0.7676	0.7798	0.7737	**0.7820**	**0.7970**	**0.7894**

*P: precision, R: recall, F1: F1-score

Table 5-3. Precision, recall, and F1-score value for four types of multi-class classifiers.

5.3 Per class classification performance comparison between BIGRU and Ensemble classification models

We compared class classification performance above two classifiers, BIGRU and Ensemble.

CLASS NAME	MULTI-CLASS CLASSIFICATION			MULTI-CLASS CLASSIFICATION :OTHER WORK[14]		
	BIGRU			Ensemble		
	P	R	F1	P	R	F1
RAMNIT	**0.9335**	**0.9765**	**0.9545**	**0.8486**	**0.9023**	**0.8747**
QAKBOT	**0.9816**	**0.9317**	**0.9560**	**0.7820**	**0.7970**	**0.7894**

*P: precision, R: recall, F1: F1-score

Table 5-4. Precision, recall, and F1-score value of ramnit and qakbot class classification.

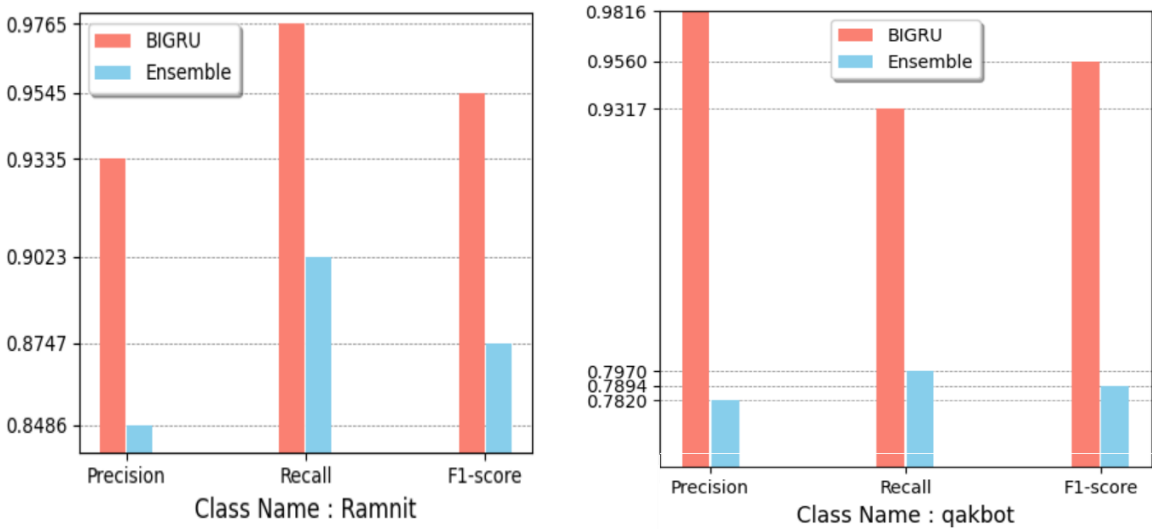


Fig 5-3. Comparison between BiGRU and Ensemble models on class ramnit and qakbot classification

F1-score of the Ensemble model to classify the class ramnit was 0.8747, while our multiclass classifier, BIGRU, was 0.9545. Thus, BIGRU has better classification performance than the Ensemble model to classify ramnit.

In the same way, the performance of classifying the class qakbot attack, the F1-score of the Ensemble model was 0.7894, while multiclass classifier, BIGRU, was 0.9560. The performance of per class classification by Ensemble model is less than BIGRU for this type of class. F1-score of BIGRU to classify qakbot and ramnit classes are greater than Ensemble model by 0.1666 and 0.0798 respectively as shown Fig 5-3.

CHAPTER SIX

6. Conclusion and Future Work

6.1 Conclusion

As mentioned before, the growth of internet service in the last recent decades was increasing exponentially worldwide. One of the key components for providing internet services for clients is Domain Name System.

Attackers have focused on this platform to exploit for different purposes and the attack types are also dynamic and increasing. Currently, in ethiotelecom, the existing security systems identify potential threats based on ruled base access control mechanisms and others. It also takes time to set policies, challenges to manage and update patches. Those types of preventing mechanisms are not dynamic, challenge to detect unknown attacks, and have high false-positive rates.

Recently, deep learning methods have emerged as a successful approach to detecting DNS attacks instead of traditional models because it has more ability to generalize.

From the results of performance measured shown in the previous section, BIGRU recorded a relatively better performance than the other deep learning classifier. BILSTM delivers a comparable result with BIGRU, it becomes the second-best classifier.

Even though the proposed experimental procedures are specifically planned for this research only, they could also implement practically too if some additional preconditions are fulfilled.

6.2 Future Work and Recommendations

The main future work from this study is that classify the DNS attacks using real DNS attacks datasets that are collected from the ethiotelecom security department instead of using global datasets. This also has to be supported by a fully privileged laboratory.

7. References

[1] Luca Deri, Lorenzo Luconi Trombacchi, Maurizio Martinelli, and Daniele Vannozzi, "Distributed DNS Traffic Monitoring System"

[2] Georgios Kambourakis, Tassos Moschos, Dimitris Geneiatakis, and Stefanos Gritzalis, "Detecting DNS Amplification Attacks"

[3] Ahmed Almusawi, and Haleh Amintoosi," DNS Tunneling Detection Method Based on Multilabel Support Vector Machine"

[4] M. M. Dissanayake, "DNS Cache Poisoning: A Review on its Technique and Counter measures"

[5] Sanjay, Balaji Rajendran, and Pushparaj Shetty D, "DNS Amplification & DNS Tunneling Attacks Simulation, Detection, and Mitigation Approaches"

[6] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li, "DeepDefense: Identifying DDOS Attack via Deep Learning"

[7] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel, "A comparison of LSTM and GRU networks for learning symbolic sequences"

[8] Richard Preston, "DNS Tunneling Detection with Supervised Learning"

[9] Yirui Wu, Dabao Wei, and Jun Feng, "Network Attacks Detection Methods Based on Deep Learning Techniques: A Survey"

[10] Keiron O'Shea and Ryan Nash, "An Introduction to Convolutional Neural Networks"22

[11] Pramita Sree Muhuri, Prosenjit Chatterjee, Xiaohong Yuan, Kaushik Roy and Albert Esterline, "Using a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to Classify Network Attacks"

[12] Anestis Karasaridis, Kathleen Meier-Hellstern, David Hoeflin, "Detection of DNS Anomalies using Flow Data Analysis"

[13] Khloud Al Jallad, Mohamad Aljnidi, and Mohammad Said Desouki, "Anomaly detection optimization using big data and deep learning to reduce false-positive"

[14] Juhong Namgung, Siwoon Son, and Yang-Sae Moon,'' Efficient Deep Learning Models for DGA Domain Detection"

[15] Gozde Karatas. Onder Demir, Ozgur Koray Sahingoz," Deep Learning in Intrusion Detection Systems"

[16] Niraj Thapa, Zhipeng Liu, Dukka B. KC, Balakrishna Gokaraju and Kaushik Roy," Comparison of Machine Learning and Deep Learning Models for Network Intrusion Detection Systems"

[17] Hongyu Liu and Bo Lang," Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey"

[18] Xingguo Li, Junfeng Wang, and Xiaosong Zhang," Botnet Detection Technology Based on DNS"

8. Appendix

Publishable manuscript

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Telecommunication Engineering Graduate Program

Multi-Class DNS Attacks Classification using Deep Learning

By: Teshome Assefa

Supervised by: Fistum Assamnew (Ph.D.)

A manuscript submitted to Addis Ababa University, Addis Ababa Institute of Technology School of Electrical and Computer Engineering, in partial fulfillment of the requirements for the degree of Master of Science in Telecommunications Engineering (Telecommunication Information system truck).

January 2021

Addis Ababa, Ethiopia